

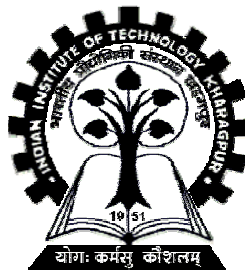
# **Focused Web Crawling for E-Learning Content**

Thesis to be submitted in Partial Fulfillment  
of the Requirements for the Award of the Degree of

**Master of Technology**

**In**

**Computer Science and Engineering**



*Submitted by:*

**Udit Sajjanhar (03CS3011)**

*Under the supervision of*

**Prof. Pabitra Mitra**

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

May 2008

# Certificate

This is to certify that the thesis titled *Focused Web Crawling for E-Learning Content* , submitted by **Udit Sajjanhar**, to the Department of Computer Science and Engineering, in partial fulfillment for the award of the degree of **Master of Technology** is a bonafide record of work carried out by him under our supervision and guidance. The thesis has fulfilled all the requirements as per the regulations of the institute and, in our opinion, has reached the standard needed for submission.

**Prof. Pabitra Mitra**

Dept. of Computer Science and Engineering  
Indian Institute of Technology  
Kharagpur 721302, INDIA

**Udit Sajjanhar**

Dept. of Computer Science and Engineering  
Indian Institute of Technology  
Kharagpur 721302, INDIA

## **Abstract**

*A focused crawler is designed to traverse the Web to gather documents on a specific topic. It can be used to build domain-specific Web search portals and online personalized search tools. To estimate the relevance of a newly seen URL, it must use information gleaned from previously crawled page sequences.*

*The work describes the design of the focused crawler for Intinno, an intelligent web based content management system. Intinno system aims to circumvent the drawbacks of existing learning management systems in terms of scarcity of content which often leads to the cold start problem. The scarcity problem is solved by using a focused crawler to mine educational content from the web. Educational content is mined from University websites in the form of course pages.*

*We present a survey of various probabilistic models such as Hidden Markov Models(HMMs) and Conditional Random Fields(CRFs) and other techniques like using context graphs for building a focused crawler and finally we describe the design of the system by applying CRFs. The system consists of the following components: learning the user browsing pattern while he or she shows how to collect relevant information and using the learnt model to predict a utility score for links that prioritizes their download.*

*It is often observed that University websites are structurally similar to each other. Humans are good at navigating websites to reach specific information within large domain-specific websites. Thus the user browsing behavior is mapped to a problem of sequential learning where states in the sequence are pages of the same nature. The model parameters of this learning problem are estimated with the help of a Conditional Random Field.*

*During the crawling phase, the focused crawler is required to prioritize the order in which the links will be downloaded so that the ratio of relevant pages visited to the total number of pages visited is maximized. This prioritization is done using Reinforcement Learning which estimates the ability of a partially observed sequence to end up in the goal state by using the probabilistic model parameters learnt during the training.*

## Table of Contents

Chapter 1: Introduction .....	5
1.1 Background .....	5
1.2 Motivation .....	8
Chapter 2: Previous work in Focused Crawling .....	10
Chapter 3: Problem Definition.....	14
3.1 Strategies for crawling .....	15
3.2 Content from University Courses.....	17
3.3 Problem Formulation.....	18
3.4 Possible Approaches .....	20
Chapter 4: Realtional Learning using Graphical Models .....	22
4.1 Introduction .....	22
4.2 Graphical Models .....	22
4.3 Conditional Random Fields.....	27
Chapter 5: Our Approach.....	30
5.1 Model Training.....	30
5.2 Path Foraging .....	31
5.3 Expremental Results.....	34
Bibliography .....	38

# Chapter 1. Introduction

## 1.1 Background

A Learning Management System (or LMS) is a software tool designed to manage user learning processes [1]. LMSs go far beyond conventional training records management and reporting. The value-add for LMSs is the extensive range of complementary functionality they offer. Learner self-service (e.g. self-registration on instructor-led training), learning workflow (e.g. user notification, teacher approval, waitlist management), the provision of on-line learning, on-line assessment, management of continuous professional education, collaborative learning (e.g. application sharing, discussion threads), and training resource management (e.g. instructors, facilities, equipment), are some of the additional dimensions to leading learning management systems [2].

In addition to managing the administrative functions of online learning, some systems also provide tools to deliver and manage instructor-led synchronous and asynchronous online teaching based on learning object methodology. These systems are called Learning content management systems or LCMSs. An LCMS provides tools for authoring and re-using or re-purposing content as well as virtual spaces for learner interaction (such as discussion forums and live chat rooms). The focus of an LCMS is on learning content. It gives authors, instructional designers, and subject matter experts the means to create and re-use e-learning content more efficiently [3].

The current learning management systems have a number of drawbacks which hinder their wide acceptance among teachers and students. One of them is the non availability of free content. LMS's assume that the content will be put up by users i.e. teachers and students. This leads to the cold start problem. Instructors who begin to make up a course don't have the material to start up. Materials presented may lack coverage of the subject area and thus fail to cater information needs of all students in a class. On the other hand, students while studying or reading a lecture have to waste a lot of their time in searching for relevant resources from the web.

We aim to build a system which solves the above problem to a large extent. The system uses the power of Web to solve the cold start problem. While putting up new course, assignment or a lecture, similar resources would be available from the digital library either by search or by recommendations. Also when a student reads a lecture he will have to his disposal:

1. *Recommended Material*: This will save his time in searching for relevant material from the web.
2. *Learning Material Abstractions*: This would help the student to efficiently grasp the learning content.

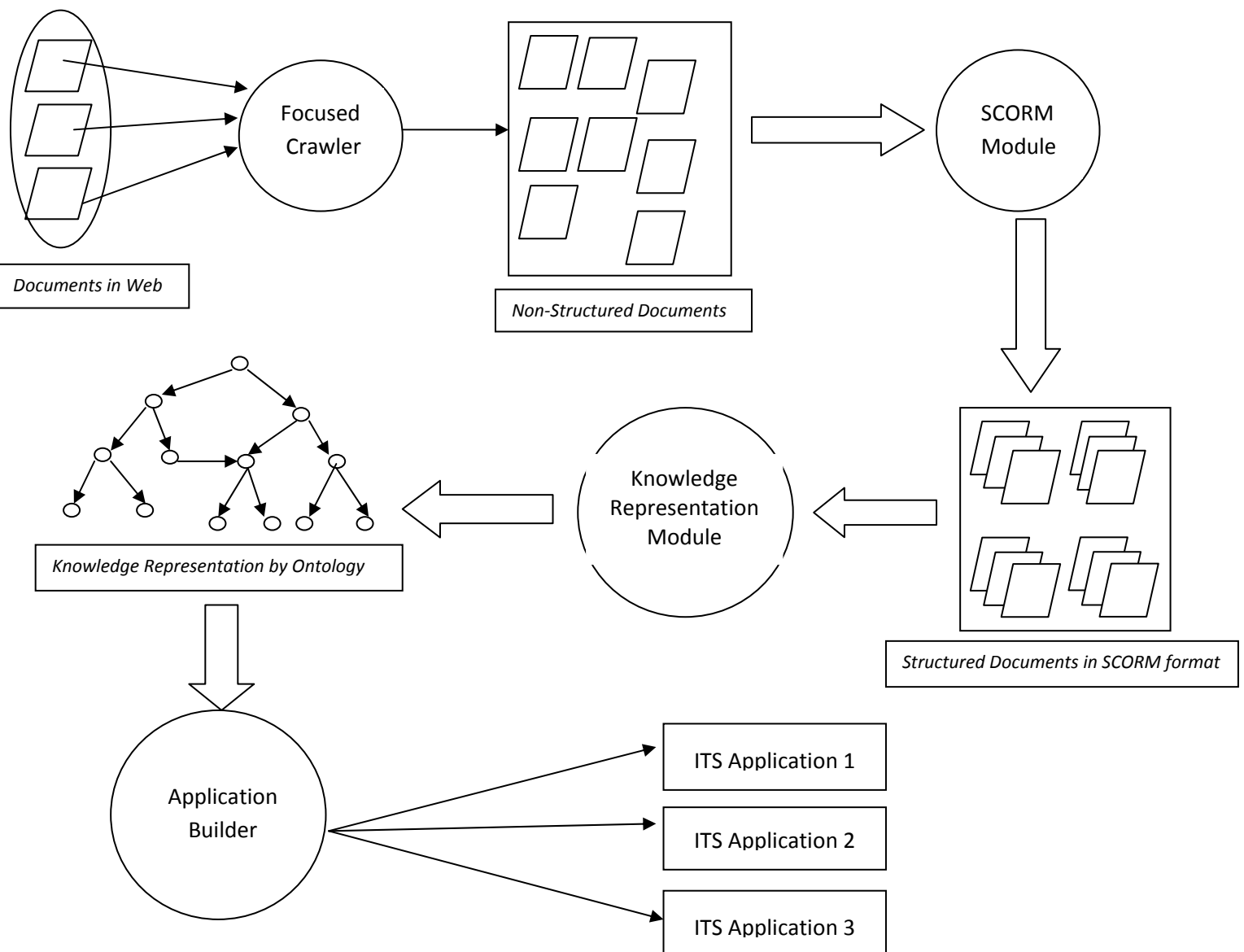


Figure 1. Components of the Intinno System

The major components of the system are:

1. *Focused Crawler:*

This exploits the structural similarity among University websites by learning human browsing pattern. From the learned parameters, it downloads educational resources with the goal of maximizing the ratio of relevant pages to the number of pages downloaded.

2. *SCORM Module:*

The data downloaded by the Focused crawler is unstructured. Educational Data is represented in SCORM (*Sharable Content Object Reference Model*) [28] format to enable easier exchange and reuse. This module organizes the unstructured data downloaded by the crawler and converts it into SCORM format so that it can be used by the Application builder in an efficient manner.

3. *Knowledge Representation Module:*

SCORM format only enables the easier exchange and reuse of Learning Objects [29]. However to build intelligent ITS applications we have to capture the semantic relations between the documents. This is done by the Knowledge Representation Module which tries captures the Semantic relations by automatic construction of Ontologies. This is done by recognition of keywords from documents and then extraction of relationships between keywords.

4. *ITS applications:*

These are the end products of the system with which the user interacts. For a student, the ITS applications recommend similar material so that the time spent in searching for relevant resources on the web is minimized. Also the ITS applications present to the student an abstraction over the material that he/she is currently reading. This helps in the easier grasping of concepts making the learning process efficient.

In this work we describe the focused crawler component of the Intinno System.

## 1.2 Motivation

The size of the publicly index able world-wide-web has provably surpassed one billion documents [30] and as yet growth shows no sign of leveling off. Dynamic content on the web is also growing as time-sensitive materials, such as news, financial data, entertainment and schedules become widely disseminated via the web. Search engines are therefore increasingly challenged when trying to maintain current indices using exhaustive crawling. Even using state of the art systems such as Google, which reportedly crawls millions of pages per day, an exhaustive crawl of the web can take weeks. Exhaustive crawls also consume vast storage and bandwidth resources, some of which are not under the control of the search engine.

Focused crawlers [3,4] aim to search and retrieve only the subset of the world-wide web that pertains to a specific topic of relevance. The ideal focused crawler retrieves the maximal set of relevant pages while simultaneously traversing the minimal number of irrelevant documents on the web. Focused crawlers therefore offer a potential solution to the currency problem by allowing for standard exhaustive crawls to be supplemented by focused crawls for categories where content changes quickly. Focused crawlers are also well suited to efficiently generate indices for niche search engines maintained by portals and user groups [31], where limited bandwidth and storage space are the norm. Finally, due to the limited resources used by a good focused crawler, users are already using personal PC based implementations. Ultimately simple focused crawlers could become the method of choice for users to perform comprehensive searches of web-related materials.

The major open problem in focused crawling is that of properly assigning credit to all pages along a crawl route that yields a highly relevant document. In the absence of a reliable credit assignment strategy, focused crawlers suffer from a limited ability to sacrifice short term document retrieval gains in the interest of better overall crawl performance. In particular, existing crawlers still fall short in learning strategies where topically relevant documents are found by following off-topic pages.



The credit assignment for focused crawlers can be significantly improved by equipping the crawler with the capability of modeling the context within which the topical materials is usually found on the web [32]. Such a context model has to capture typical link hierarchies within which valuable pages occur, as well as describe off-topic content that co-occurs in documents that are frequently closely associated with relevant pages.

The focused crawler for the Intinno system tries to collect the course pages which are rich source of authenticated educational content. Crawling the whole university and then separating out the course pages with the help of a classifier is the simplest solution. However such a solution is highly in-efficient both in terms of Space and Time required. In another scheme crawler learns the user browsing pattern as the user starts from the University homepage and follows a path which consists of many non-topical pages to reach the pages hosting course content. Hence a context driven crawling scheme which learns the link hierarchies among the pages leading to the relevant page is required to train the crawler.

## Chapter 2: Prior Work in Focused Crawling

The first generation of crawlers [33] on which most of the web search engines are based rely heavily on traditional graph algorithms, such as breadth-first or depth-first traversal, to index the web. A core set of URLs are used as a seed set, and the algorithm recursively follows hyper links down to other documents. Document content is paid little heed, since the ultimate goal of the crawl is to cover the whole web.

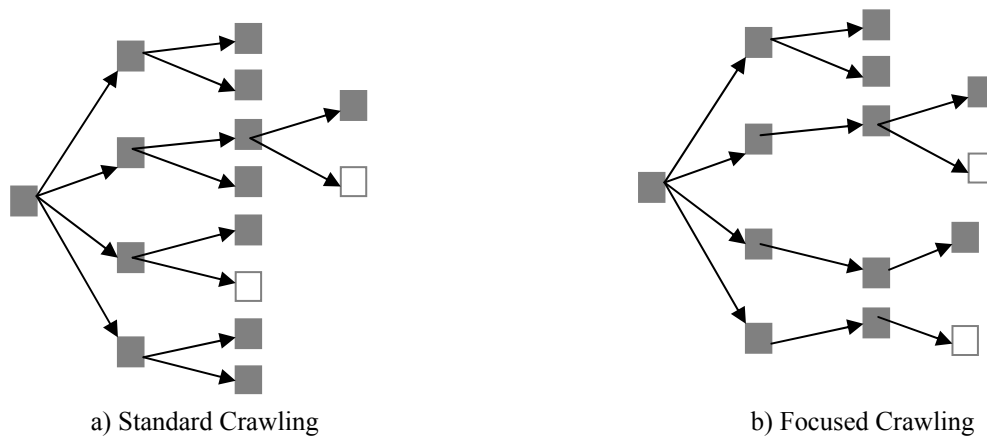


Figure 2:

- a) A standard crawler follows each link, typically applying a breadth first strategy. If the crawler starts from a document which is  $i$  steps from a target document, all the documents that are up to  $i-1$  steps from the starting document must be downloaded before the crawler hits the target.
- b) A focused crawler tries to identify the most promising links, and ignores off-topic documents. If the crawler starts from a document which is  $i$  steps from a target document, it downloads a small subset of all the documents that are up to  $i-1$  steps from the starting document. If the search strategy is optimal the crawler takes only  $i$  steps to discover the target.

A focused crawler efficiently seeks out documents about a specific topic and guides the search based on both the content and link structure of the web [4]. Figure 2 graphically illustrates the difference between an exhaustive breadth first crawler and a typical focused crawler. A focused crawler implements a strategy that associates a score with each link in the pages it has downloaded [4, 15, 20]. The links are sorted according to the scores and inserted in a queue. A best first search is performed by popping the next page to analyze from the head of the queue. This strategy ensures that the crawler preferentially pursues promising crawl paths.

A variety of methods for focused crawling have been developed. The term focused crawler was first coined by Chakrabarti in [4], however, the concept of prioritizing unvisited URLs on the crawl frontier for specific searching goals is not new, and Fish-Search [34] by De Bra et al. (1994) and Shark-Search [35] by Hersovici et al. (1998) were some of the earliest algorithms for crawling for pages with keywords specified in the query. In Fish-Search, the Web is crawled by a team of crawlers, which are viewed as a school of fish. If the “fish” finds a relevant page based on keywords specified in the query, it continues looking by following more links from that page. If the page is not relevant, its child links receive a low preferential value. Shark-Search is a modification of Fish-search which differs in two ways: a child inherits a discounted value of the score of its parent, and this score is combined with a value based on the anchor text that occurs around the link in the Web page.

The focused crawler introduced in [4] uses canonical topic taxonomy. The user specifies a number of starting points, and browses from these pages. The user matches pages to the best categories while browsing. This categorization is used to train a classifier which makes relevance judgements on a document to the topic. A distiller then determines visit priorities based on hub-authority connectivity analysis. Intelligent crawling with arbitrary predicates is described in [22]. The method involves looking for specific features in a page to rank the candidate links. These features include page content, URL names of referred Web page, and the nature of the parent and sibling pages. It is a generic framework in that it allows the user to specify the relevant criteria. Also, the system has the ability of self-learning, i.e. to collect statistical information during the crawl and adjust the weight of these features to capture the dominant individual factor at that moment.

Similar methods can be found in [20, 36, 37]. These methods have in common that they use a baseline best-first focused crawling strategy combined with different heuristics based on local features extracted from the parent page, such as similarity to a query, in-degree, PageRank, and relevance feedback. All of the methods mentioned above are based on the assumption that on the Web a relevant page would link to other relevant pages. Thus if the crawler has found a relevant page then the links extracted out of this page will have a greater probability of being downloaded as compared to the links extracted from a less relevant page.

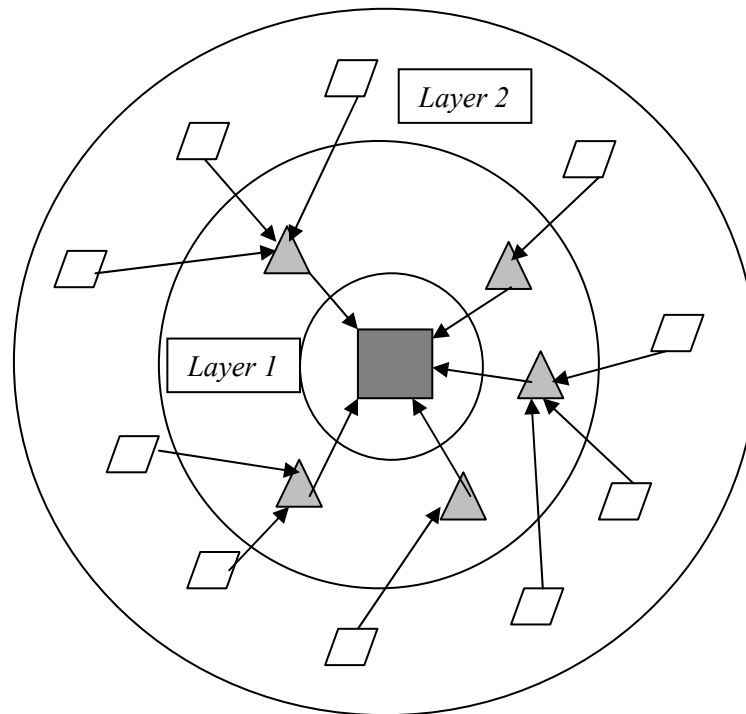
Other methods used to capture path information leading to targets include reinforcement learning [16], genetic programming [38], and Context Graph algorithms [32]. In [16], an algorithm was presented for learning a mapping performed by Naive Bayes text classifiers from the text surrounding a hyperlink to a value function of sum of rewards, the estimate of the number of relevant pages that can be found as the result of following that hyperlink.

A genetic programming algorithm is used in [38] to explore the space of potential strategies and evolve good strategies based on the text and link structure of the referring pages. The strategies produce a rank function which is a weighted sum of several scores such as hub, authority and SVM scores of parent pages going back k generations.

A problem with the approaches mentioned above is that they cannot exploit patterns of pages along a route leading to a goal page. The only pattern they exploit is the relevance to a specific topic. However in some cases often pages on unrelated topics might consistently lead to topics of interest. To explicitly address this problem, Rennie and McCallum [16] used reinforcement learning to train a crawler on specified example web sites containing target documents. The web site or server on which the document appears is repeatedly crawled to learn how to construct optimized paths to the target documents. However, this approach places a burden on the user to specify representative web sites. Initialization can be slow since the search could result in the crawling of a substantial fraction of the host web site. Furthermore, this approach could face difficulty when a hierarchy is distributed across a number of sites.

The Context Graph method, proposed by Diligenti et al. [32] uses backlinks to estimate the link distance from a page to target pages. Their method starts from a set of seed documents, follows backlinks to a certain layer, and then builds up a context graph for the seed pages. A classifier is constructed for each layer using the Naive Bayes algorithm. As a new document is found, it is classified into a layer. Documents classified into layers closer to the target are crawled first. The experiments showed that this approach maintained a higher level of relevance in the retrieved Web pages. However, the assumption that all pages in a certain layer from a target document belong to the same topic described by a set of terms does not always hold. Also this does not take into account the case in which multiple pages of similar kind may have to be passed before

transiting to a page of another kind. Graphical models such as HMMs and CRFs are able to include such cases as well, thereby making the model stronger. Also relying on Search engines for back links is not a efficient strategy. A complete framework to evaluate different crawling strategies is described in [39 - 41]. An application of extending digital libraries with a focused crawler can be found in [42].






 Documents in Layer 2     
  Documents in Layer 1     
  Target Document Representation

Figure 3: A context graph represents how a target document can be accessed from the web. In each node a web document representation is stored. The graph is organized into layers: each node of layer  $i$  is connected to one (and only one) node of the layer  $i-1$  (except the single node in layer 0). There are no connections between nodes at the same level. The seed document is stored in layer 0. A document is in layer  $i$  if at least  $i$  steps (link followings) are needed to reach the target page starting from that document.

### Chapter 3: Problem Definition

Web being a rich repository of learning content, we attempt to collect high volume of learning material from web using a web miner [3]. The type of content required for the digital library would include:

- (a) Courses
- (b) Assignments
- (c) Lectures & Tutorials
- (d) Animations & Videos
- (e) Quizzes & Questions.
- (f) Discussion Forums.
- (g) Information of relevant technologies from the industry.

This content can be mined from the following sources:

1. Websites hosting standardized, reviewed and open source course material like MIT Open Courseware, NPTEL India.
2. Course websites of large international universities.
3. Discussion Forums - Google Groups, Yahoo Answers
4. Websites for animations/videos - Youtube, Google Video and metacafe
5. Websites for general content - Wikipedia, Mathworld
6. Company Websites for product related info and case studies
7. Domain specific websites for questions, tutorials etc.

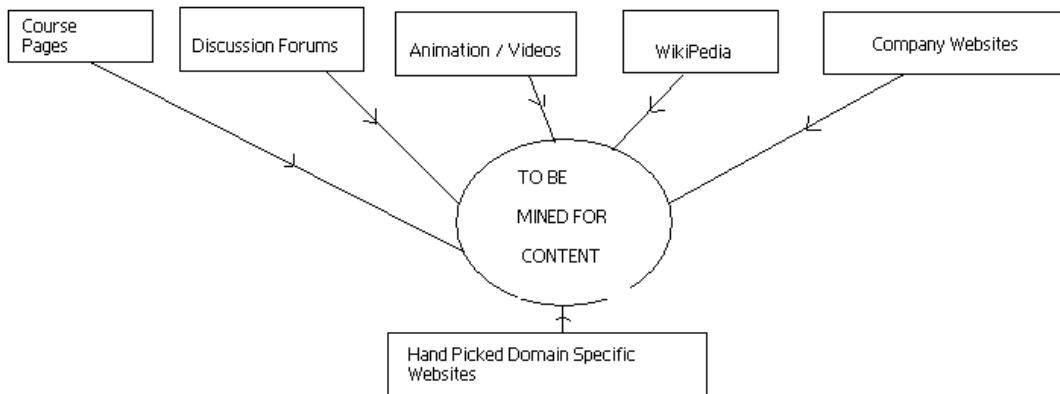


Figure 4: Pictorial description of sources to be mined for content

### 3.1 Strategies for crawling the above resources

#### 3.1.1 Open source and reviewed course pages

A general purpose crawler to crawl all the courses from MIT OCW and NPTEL is employed. Content is structured and thus is easier to crawl. Also it provides us a list of basic courses to include in the digital library. Courses from MIT OCW can be downloaded directly and the download data is arranged into folders. The content from NPTEL is ad hoc and cannot be downloaded directly. Hence, data downloaded from NPTEL will have to be catalogued.

#### 3.1.2 Content from University course pages

Full crawl is not possible in this case and we opt for focused crawling [4]. Focused crawling is possible due to the following observations in most universities page structures.

- Every University page has a page listing all its schools and departments
- Every Dept will have a page listing all its faculty members
- Every faculty member will have links of the courses on his home page.

The above structure is utilized to tune the focused crawler. The focused first learns this structural similarity by observing the browsing behavior of a human who intends to reach a course page starting from the University homepage. This structural similarity is modeled in terms of features from the pages and the features from the links. Using probabilistic models the focused crawler learns the features and while crawling it uses the learned features to prioritize the download of links. The exact scheme is described in Chapter 5.

We also attempted direct search on Google for university course material. Using Google search keywords of the form: <name of course> course page syllabus etc.. often returns many course pages. However this approach has the problem of manually listing the names of all the courses in order to reach them. The problem lies in the fact that we don't have a exhaustive list of courses. Also since we don't have the list of courses from a particular university, quantifying the recall

for this method would be difficult. Also the number of automatic queries that can be sent to a search engine are limited and thus this method is not efficient.

Another issue involved for such course pages is that of extraction of learning content from courses on the web. The data downloaded from a course on the web, may be arranged in various ways and needs to be processed to extract the relevant information. Here we propose a simplistic algorithm for doing it in each of the following two cases:

**Case 1:** All the data of a particular course lies on one page. In this case different kinds of data will be under corresponding headings. For example all the assignments of a particular course will be under the assignments headings and all the questions will be under the questions heading. To extract data from such a course, we detect the headings on a particular page and we hypothesize that all the data under a heading is of that type. The algorithm is described in section 5.3.2 and has about 65% accuracy.

**Case 2:** The course page has separate links for separate kind of data i.e. the assignments are on one page and the questions on another. We assume that these separate pages have such an anchor text that indicates the type of content on the page. For example the main course has links to Assignments / Lectures and Quizzes. To extract data from such a course we assume that type of content on each page to be given by the anchor text on the hyperlink.

### 3.1.3 Unstructured data: Discussion forums and animation videos

Full Crawl is irrelevant and is also not possible. Focused crawling is the approach adopted. From the courses already stored in the digital library now extract a set of keywords, including, (i) Terms from the name of the course, (ii) Terms from the syllabus of the course, and (iii) Terms from assignment heading/name, Lecture heading/name.

Next we search for discussions/Animations/Videos from the web which match the above list of keywords and index the results obtained above with the keywords with which they were found and the content of the entity obtained.



### 3.1.4 General purpose collections like WikiPedia

Full Crawl of Wikipedia is possible and can be obtained as a single XML document. However, full crawl/download may not be necessary and may in fact weaken precision of the search on digital library. We use a keyword based focused approach described above to limit the pages being indexed in Wikipedia. Each Wikipedia article can be characterized a lectures or tutorials. While indexing the articles of Wiki more importance should is given to the headings and the sub headings on the page.

### 3.1.5 Websites of industrial corporations

Websites in this category will have to be handpicked and will be few in number. Examples of company websites includes whitepapers, manuals, tutorials obtained from research lab of companies like IBM, Google, Microsoft, GE. Handpicked websites of popular corporate training resources like those offering questions/quizzes on C and those offering tutorials like How Stuff Works.

## **3.2 Content from university course pages**

Out of the above mentioned sources, course websites of different Universities are the richest source of learning content. The advantages of this content are:

1. Since this content is hosted on the University site, under the professor/teacher taking this course, the content is deemed to be authenticated and correct.
2. Also this type of content is used in a real scenario the teach the students and hence is most relevant to the students.

However, along with being the richest source of valid educational content this type of content is most difficult to mine. This is due to the fact that this content is non-structured in nature. There are following difficulties in mining this content:

- Every teacher has his/her own way of hosting the content. Some might be putting up the whole content in a single page while others might be having a more structured representation of content with different sections for assignments, lectures, etc.
- Every University has their own sitemap. A set of rules to reach the course pages starting from University homepage, if designed for a particular university, might not work for every case.

One of the solutions to get the course pages from a particular university would be to crawl the whole university and separate out the course pages from the set of all crawled pages. The separation of course pages will be done by a binary classifier that will be trained on the prior set of course pages that can be obtained with the help of a search engine. However crawling the whole university for course pages would be inefficient both in terms of Time and Space required. Hence we need a focused crawling [4] technique to efficiently mine relevant course pages starting from the university homepage.

### **3.3 Problem Formulation**

It is often observed that University websites are structurally similar to each other. Humans are good at navigating websites to reach specific information within large domain-specific websites. Our system tries to learn the navigation path by observing the user's clicks on as few example searches as possible and then use the learnt model to automatically find the desired pages using as few redundant page fetches as possible. Unlike in focused crawling [4], our goal is not to locate the websites to start with. These are collected from web directories [5] and similar resource websites. We start from a listing of University homepages and after watching the user find the specific information from a few websites in the list, we automate the search in the remaining.

There are two phases to this task:

1. Training phase: where the user teaches the system by clicking through pages and labeling a subset with a dynamically defined set of classes, one of them being the Goal class. The

classes assigned on intermittent pages along the path can be thought of as “milestones” that capture the structural similarity across websites. At the end of this process, we have a set of classes  $C$  and a set of training paths where a subset of the pages in the path are labeled with a class from  $C$ . All unlabeled pages before a labeled page are represented with a special prefix state for that label. The system trains a model using the example paths, modeling each class in  $C$  as a milestone state.

2. Crawling phase: where the given list of websites is automatically navigated to find all goal pages. The system uses the model parameters learnt in the training phase to prioritize the download of links trying to optimize the ratio of the number of relevant pages downloaded to the total number of pages downloaded.

Formally, we are given a website as a graph  $W(V,E)$  consisting of vertex set  $V$  and edge set  $E$ , where a vertex is a webpage and an edge  $e = \langle u, v \rangle$  is a hyperlink pointing from a webpage  $u$  to a webpage  $v$ . The goal pages  $P_G$  constitute a subset of pages in  $W$  reachable from starting seed page  $P_S$ . We have to navigate to them starting from  $P_S$  visiting fewest possible additional pages. Let  $P : P_1, P_2, \dots, P_n$  be one such path through  $W$  from the start page  $P_1 = P_S$  to a goal page  $P_n \in P_G$ . The ratio of relevant pages visited to the total number of pages visited during the execution is called the **harvest rate**. The objective function is to maximize the harvest rate.

This problem requires two solutions.

1. Recognizing a page as the goal page. This is a classification problem where given a webpage we have to classify it as being a goal page or not. Often the page alone may not hold enough information to help identify it as the goal page. We will need to consider text around the entire path leading to the goal page in order to decide if it is relevant or not. For example, if we want to get all course pages starting from a university root page, then it is necessary to follow a path through departments’ homepages and then through professors’ homepage. A course page on its own might be hard to classify.

2. *Foraging for goal pages.* This can be thought as a crawling exercise where, starting from the entry point, we want to visit as few pages as possible in finding the goal pages. This problem is different from the previous work on focused crawling[4] where the goal is to find all web pages relevant to a particular broad topic from the entire web. In our case, we are interested in finding course pages starting from a University homepage. We exploit the regularity in the structures of University websites to build more powerful models than is possible in the case of general-purpose focused crawlers.

### 3.4 Possible Approaches

One possible method of solving the problem is to train a classifier that can discriminate the goal pages from the non-goal pages. Then, extract from the classifier the set of prominent features to serve as keywords to a search engine that indexes all the websites of interest. By restricting the domain to each given starting URL in turn, we issue a keyword search to get a set of candidate pages. We further classify these pages to identify if these are goal pages or not. However this method cannot provide high accuracy for the simple reason that the goal page itself may not hold enough information to correctly identify it as the goal page. The path leading to the goal page is important too.

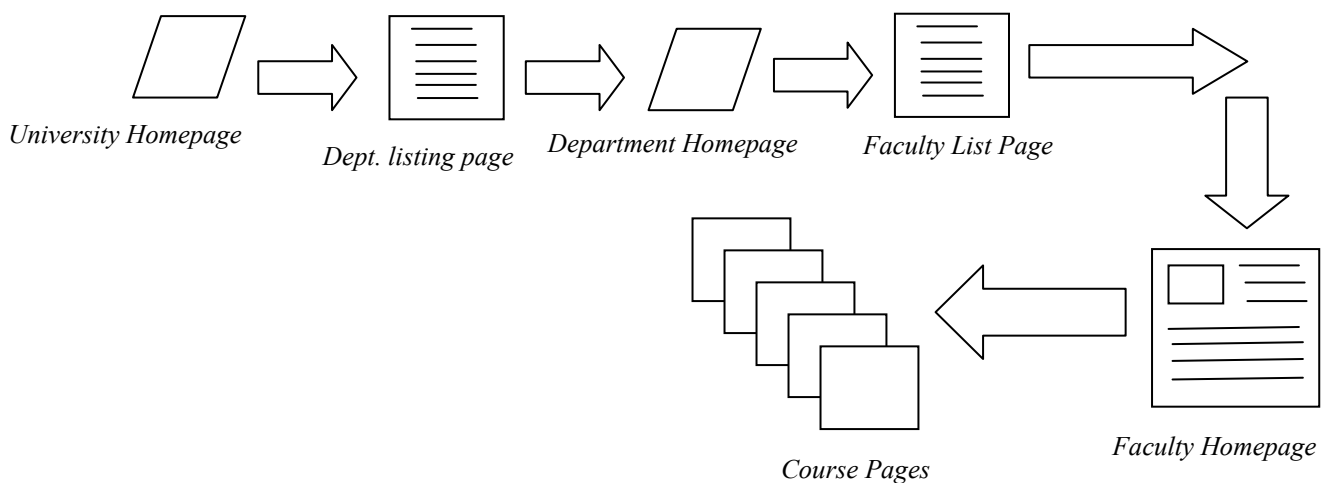


Figure 5: Pictorial description of the Sequential labeling problem for reaching the course pages

A Focused crawler must use information gleaned from previously crawled page sequences to estimate the relevance of a newly seen URL. Therefore, good performance depends on powerful modeling of context as well as the current observations. Probabilistic models, such as Hidden Markov Models( HMMs)[6] and Conditional Random Fields(CRFs)[7], can potentially capture both formatting and context.

Thus a second approach and the one that we use is to treat this as a sequential labeling problem where we use Hidden Markov Models (HMMs) and the Conditional Random Fields to learn to recognize paths that lead to goal states. We then superimpose ideas from Reinforcement Learning [8] to prioritize the order in which pages should be fetched to reach the goal page. This provides an elegant and unified mechanism of modeling the path learning and foraging problem.

## Chapter 4: Relational learning using Graphical Models

### 4.1 Introduction

Relational data has two characteristics: first, statistical dependencies exist between the entities we wish to model, and second, each entity often has a rich set of features that can aid classification. For example, when classifying Web documents, the page's text provides much information about the class label, but hyperlinks define a relationship between pages that can improve classification [43]. Graphical models are a natural formalism for exploiting the dependence structure among entities. Traditionally, graphical models have been used to represent the joint probability distribution  $p(y, x)$ , where the variables  $y$  represent the attributes of the entities that we wish to predict, and the input variables  $x$  represent our observed knowledge about the entities. But modeling the joint distribution can lead to difficulties when using the rich local features that can occur in relational data, because it requires modeling the distribution  $p(x)$ , which can include complex dependencies. Modeling these dependencies among inputs can lead to intractable models, but ignoring them can lead to reduced performance.

A solution to this problem is to directly model the conditional distribution  $p(y|x)$ , which is sufficient for classification. This is the approach taken by conditional random fields [7]. A conditional random field is simply a conditional distribution  $p(y|x)$  with an associated graphical structure. Because the model is conditional, dependencies among the input variables  $x$  do not need to be explicitly represented, affording the use of rich, global features of the input.

### 4.2 Graphical Models

#### 4.2.1 Definitions

We consider probability distributions over sets of random variables  $\mathbf{V} = \mathbf{X} \cup \mathbf{Y}$ , where  $\mathbf{X}$  is a set of input variables that we assume are observed, and  $\mathbf{Y}$  is a set of output variables that we wish to predict. Every variable  $v \in \mathbf{V}$  takes outcomes from a set  $\mathcal{V}$ , which can be either continuous or discrete. We denote an assignment to  $\mathbf{X}$  by  $x$ , and we denote an assignment to a set  $A \subset \mathbf{X}$  by  $x_A$ , and similarly for  $\mathbf{Y}$ .

A graphical model is a family of probability distributions that factorize according to an underlying graph. The main idea is to represent a distribution over a large number of random variables by a product of local functions that each depend on only a small number of variables. Given a collection of subsets  $\mathbf{A} \subset \mathbf{V}$ , we define an undirected graphical model as the set of all distributions that can be written in the form

$$p(x, y) = \frac{1}{Z} \prod_A \Psi_A(x_A, y_A)$$

for any choice of factors  $\mathbf{F} = \{\Psi_A\}$ , where  $\Psi_A: \mathcal{V}^n \rightarrow \mathcal{R}^+$ . (These functions are also called *local functions* or *compatibility functions*). The term random field to refer to a particular distribution among those defined by an undirected model. To reiterate, the term model is used to refer to a family of distributions, and random field (or more commonly, distribution) to refer to a single one. The constant  $Z$  is a normalization factor defined as

$$Z = \sum_{x, y} \prod_A \Psi_A(x_A, y_A)$$

which ensures that the distribution sums to 1. The quantity  $Z$ , considered as a function of the set  $\mathbf{F}$  of factors, is called the partition function in the statistical physics and graphical models communities. In this chapter we assume that local functions have the form:

$$\Psi_A(x_A, y_A) = \exp\left\{\sum_k \theta_{Ak} f_{Ak}(x_A, y_A)\right\}$$

for some real-valued parameter vector  $\theta_A$ , and for some set of feature functions or sufficient statistics  $\{f_{Ak}\}$ . This form ensures that the family of distributions over  $\mathbf{V}$  parameterized by  $\theta$  is an exponential family.

A directed graphical model, also known as a Bayesian network, is based on a directed graph  $G = (\mathbf{V}, \mathbf{E})$ . A directed model is a family of distributions that factorize as:

$$p(x, y) = \prod_{v \in \mathbf{V}} p(v | \pi(v))$$

where  $\pi(v)$  are the parents of  $v$  in  $\mathbf{G}$ . An example of directed model is shown in figure 6. The term *generative model* to refer to a directed graphical model in which the outputs topologically precede the inputs, that is, no  $x \in \mathbf{X}$  can be a parent of an output  $y \in \mathbf{Y}$ . Essentially, a generative model is one that directly describes how the outputs probabilistically “generate” the inputs.

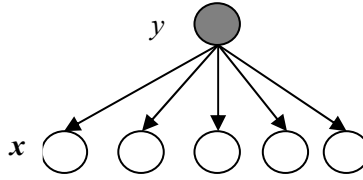


Figure 6: The naïve Bayes Classifier, as a directed graphical model.

#### 4.2.2 Sequence Models

The true power of graphical models lies in their ability to model many variables that are interdependent. In this section, we discuss perhaps the simplest form of dependency, in which the output variables are arranged in a sequence. To motivate this kind of model, we discuss an application from natural language processing, the task of named-entity recognition (NER). NER is the problem of identifying and classifying proper names in text, including locations, such as China; people, such as George Bush; and organizations, such as the United Nations. The named-entity recognition task is, given a sentence, first to segment which words are part of entities, and then to classify each entity by type (person, organization, location, and so on). The challenge of this problem is that many named entities are too rare to appear even in a large training set, and therefore the system must identify them based only on context. One approach to NER is to classify each word independently as one of either: Person, Location, Organization, or Other (meaning not an entity). The problem with this approach is that it assumes that given the input, all of the named-entity labels are independent. In fact, the named-entity labels of neighboring words are dependent; for example, while New York is a location, New York Times is an organization.

This independence assumption can be relaxed by arranging the output variables in a linear chain. This is the approach taken by the Hidden Markov model (HMM) [6]. An HMM models a sequence of observations  $X = \{x_t\}_{t=1}^T$  by assuming that there is an underlying *sequence of states*  $Y = \{y_t\}_{t=1}^T$  drawn from a finite state set  $S$ . In the named-entity example, each observation  $x_t$  is the identity of the word at position  $t$ , and each state  $y_t$  is the named-entity label, that is, one of the entity types Person, Location, Organization, and Other.



To model the joint distribution  $p(\mathbf{y}, \mathbf{x})$  tractably, an HMM makes two independence assumptions. First, it assumes that each state depends only on its immediate predecessor, that is, each state  $y_t$  is independent of all its ancestors  $y_1, y_2, \dots, y_{t-2}$  given its previous state  $y_{t-1}$ . Second, an HMM assumes that each observation variable  $x_t$  depends only on the current state  $y_t$ . With these assumptions, we can specify an HMM using three probability distributions: first, the distribution over initial states  $p(y_1)$ ; second, the transition distribution  $p(y_t | y_{t-1})$ ; and finally, the observation distribution  $p(x_t | y_t)$ . That is, the joint probability of a state sequence  $y$  and an observation sequence  $x$  factorizes as

$$p(y, x) = \prod_{t=1}^T p(y_t | y_{t-1}) p(x_t | y_t)$$

In natural language processing, HMMs have been used for sequence labeling tasks such as part-of-speech tagging, named-entity recognition, and information extraction.

### 4.2.3 Discriminative Generative Models

An important difference between naive Bayes and logistic regression is that naïve Bayes is generative, meaning that it is based on a model of the joint distribution  $p(y, x)$ , while logistic regression is discriminative, meaning that it is based on a model of the conditional distribution  $p(y|x)$ . In this section, we discuss the differences between generative and discriminative modeling, and the advantages of discriminative modeling for many tasks. For concreteness, we focus on the examples of naive Bayes and logistic regression, but the discussion in this section actually applies in general to the differences between generative models and conditional random fields.

The main difference is that a conditional distribution  $p(y|x)$  does not include a model of  $p(x)$ , which is not needed for classification anyway. The difficulty in modeling  $p(x)$  is that it often contains many highly dependent features, which are difficult to model. For example, in named-entity recognition, an HMM relies on only one feature, the word's identity. But many words,

especially proper names, will not have occurred in the training set, so the word-identity feature is uninformative. To label unseen words, we would like to exploit other features of a word, such as its capitalization, its neighboring words, its prefixes and suffixes, its membership in predetermined lists of people and locations, and so on.

To include interdependent features in a generative model, we have two choices: enhance the model to represent dependencies among the inputs, or make simplifying independence assumptions, such as the naive Bayes assumption. The first approach, enhancing the model, is often difficult to do while retaining tractability. For example, it is hard to imagine how to model the dependence between the capitalization of a word and its suffixes, nor do we particularly wish to do so, since we always observe the test sentences anyway. The second approach, adding independence assumptions among the inputs, is problematic because it can hurt performance. For example, although the naive Bayes classifier performs surprisingly well in document classification, it performs worse on average across a range of applications than logistic regression.

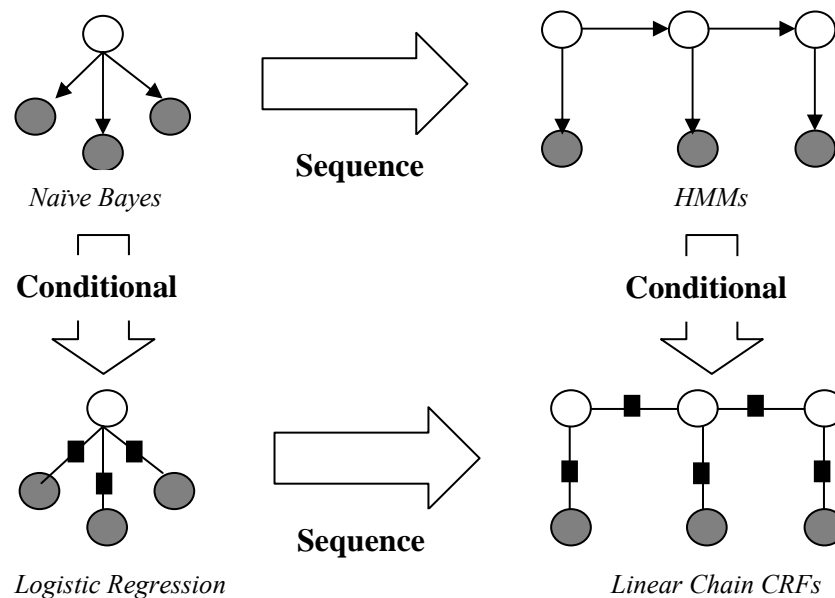


Figure 7: Diagram of relationship between Naïve Bayes, Logistic Regression, HMMs and Linear Chain CRFs

Furthermore, even when naive Bayes has good classification accuracy, its probability estimates tend to be poor. To understand why, imagine training naïve Bayes on a data set in which all the features are repeated, that is,  $\mathbf{x} = (x_1, x_1, x_2, x_2, \dots, x_K, x_K)$ . This will increase the confidence of the naive Bayes probability estimates, even though no new information has been added to the data. Assumptions like naive Bayes can be especially problematic when we generalize to sequence models, because inference essentially combines evidence from different parts of the model. If probability estimates at a local level are overconfident, it might be difficult to combine them sensibly. Actually, the difference in performance between naive Bayes and logistic regression is due only to the fact that the first is generative and the second discriminative; the two classifiers are, for discrete input, identical in all other respects. Naive Bayes and logistic regression consider the same hypothesis space, in the sense that any logistic regression classifier can be converted into a naive Bayes classifier with the same decision boundary, and vice versa.

The principal advantage of discriminative modeling is that it is better suited to including rich, overlapping features. To understand this, consider the family of naïve Bayes distributions. This is a family of joint distributions whose conditionals all take the “logistic regression form”. But there are many other joint models, some with complex dependencies among  $\mathbf{x}$ . By modeling the conditional distribution directly, we can remain agnostic about the form of  $p(\mathbf{x})$ . This may explain why it has been observed that conditional random fields tend to be more robust than generative models to violations of their independence assumptions. Simply put, CRFs make independence assumptions among  $\mathbf{y}$ , but not among  $\mathbf{x}$ .

### 4.3 Conditional Random Fields

Assume a vector  $\mathbf{f}$  of *local feature functions*  $\mathbf{f} = \langle f_1, \dots, f_K \rangle$ , each of which maps a pair  $(\mathbf{x}, \mathbf{y})$  and a position  $i$  in the vector  $\mathbf{x}$  to a measurement  $f_k(i, \mathbf{x}, \mathbf{y}) \in \mathbb{R}$ . Let  $\mathbf{f}(i, \mathbf{x}, \mathbf{y})$  be the vector of these measurements and let  $\mathbf{F}(\mathbf{x}, \mathbf{y}) = \sum_i^{|\mathbf{x}|} \mathbf{f}(i, \mathbf{x}, \mathbf{y})$ . For the case of NER, the components of  $\mathbf{f}$  might include the measurement  $f^{13}(i, \mathbf{x}, \mathbf{y}) = [[xi \text{ is capitalized}]] \cdot [yi = I]$ , where the indicator function  $[[c]] = 1$  if  $c$  is true and 0 otherwise; this implies that  $F^{13}(\mathbf{x}, \mathbf{y})$  would be the number of capitalized words paired with the label  $I$ . For the sake of efficiency, we restrict any feature  $f_k(i, \mathbf{x}, \mathbf{y})$  to be **local** in the sense that the feature at a position  $i$  will depend only on the previous labels.

With a slight abuse of notation, we claim that a local feature  $f_k(i, \mathbf{x}, \mathbf{y})$  can be expressed as  $f_k(y_i, y_{i-1}, \mathbf{x}, i)$ . Some subset of these features can be simplified further to depend only on the current state and are independent of the previous state. We will refer to these as **state features** and denote these by  $f_k(y_i, \mathbf{x}, i)$  when we want to make the distinction explicit. The term **transition features** refers to the remaining features that are not independent of the previous state.

A Conditional Random Field (CRF) is an estimator of the form

$$\Pr(\mathbf{y}|\mathbf{x}, \mathbf{W}) = \frac{1}{Z(\mathbf{x})} e^{\mathbf{W} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y})}$$

where  $\mathbf{W}$  is a weight vector over the components of  $\mathbf{F}$  and the normalizing term is:

$$Z(\mathbf{x}) = \sum_{\mathbf{y}'} e^{\mathbf{W} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}')}$$

#### 4.3.1 An Efficient Inference Algorithm

The *inference problem* for a CRF is defined as follows: Given  $\mathbf{W}$  and  $\mathbf{x}$ , find the best label sequence,  $\arg \max_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}, \mathbf{W})$

$$\begin{aligned} \arg \max_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}, \mathbf{W}) &= \arg \max_{\mathbf{y}} \mathbf{W} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}) \\ \arg \max_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}, \mathbf{W}) &= \arg \max_{\mathbf{y}} \mathbf{W} \cdot \sum_j f(y_j, y_{j-1}, \mathbf{x}, \mathbf{j}) \end{aligned}$$

An efficient inference algorithm is possible because all features are assumed to be local. Let  $\mathbf{y}_{1:i}$  denote the set of all partial labels starting from 1 (the first index of the sequence) to  $i$ , such that the  $i$ -th label is  $y$ . Let  $\delta(i, y)$  denote the largest value of  $\mathbf{W} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}')$  for any  $\mathbf{y}' \in \mathbf{y}_{1:i}$ . The following recursive calculation implements the usual Viterbi algorithm[44]:

$$\delta(i, y) = \begin{cases} \max_{y'} \delta(i-1, y') + \mathbf{W} \cdot f(y', y, \mathbf{x}, i), & \text{if } i > 0 \\ 1, & \text{if } i = 0 \end{cases}$$

The best label then corresponds to the path traced by  $\max_{\mathbf{y}} \delta(|\mathbf{x}|, y)$ .

### 4.3.2 Training Algorithm

Learning is performed by setting parameters to maximize the likelihood of a set of a training set  $T = \{(\mathbf{x}_l, \mathbf{y}_l)\}_{l=1}^N$  expressed in logarithmic terms as

$$L(W) = \sum_l \log \Pr(\mathbf{y}_l | \mathbf{x}_l, W) = \sum_l (W \cdot F(\mathbf{y}_l, \mathbf{x}_l) - \log Z_w(\mathbf{x}_l))$$

We wish to find a  $\mathbf{W}$  that maximizes  $L(\mathbf{W})$ . The above equation is convex and can thus be maximized by gradient ascent or one of many related methods. The gradient of  $L(\mathbf{W})$  is the following:

$$\nabla L(W) = \sum_l F(\mathbf{y}_l, \mathbf{x}_l) - E_{\Pr(\mathbf{y}' | W)} F(\mathbf{x}_l, \mathbf{y}')$$

The first set of terms are easy to compute. However, we must use the Markov property of  $\mathbf{F}$  and a dynamic programming step to compute the normalizer  $Z\mathbf{W}(\mathbf{x}_l)$ , and the expected value of the features under the current weight vector,  $E_{\Pr(\mathbf{y} | \mathbf{W})} \mathbf{F}(\mathbf{x}_l, \mathbf{y}_l)$ .

## Chapter 5: Our Approach

We first describe the model training phase where the user provided example positive and negative paths from a few websites are used to train a CRF model. We then describe how this trained model is used to locate goal pages starting from root pages of other websites.

### 5.1 Model Training

During training, we are given examples of several paths of labeled pages where some of the paths end in goal pages and others end with a special “fail” label. We can treat each path as a sequence of pages denoted by the vector  $\mathbf{x}$  and their corresponding milestone labels denoted by  $\mathbf{y}$ . Each  $x_i$  is a webpage represented suitably in terms of features derived from the words in the page, its URL, and anchor text in the link pointing to  $x_i$ .

A number of design decisions about the label space and feature space need to be made in constructing a CRF to recognize characteristics of valid paths. One option is to assign a state to each possible label in the set  $L$  which consists of the milestone labels and two special labels “goal” and “fail”. State features are defined on the words or other properties comprising a page. For example, state features derived from words are of the form

$$fk(i, \mathbf{x}, y_i) = [[x_i \text{ is “computer” and } y_i = \text{faculty}]]$$

The URL of a page also yields valuable features. For example, a “tilde” in the URL is strongly associated with a personal home page and a link name with word “contact” is strongly associated with an address page. We tokenize each URL on delimiters and add a feature corresponding to each token. Transition features capture the soft precedence order among labels. One set of transition features are of the form:

$$fk(i, \mathbf{x}, y_i, y_{i-1}) = [[y_i \text{ is “faculty” and } y_{i-1} \text{ is “department”}]].$$

They are independent of  $x_i$  and are called **edge features** since they capture dependency among adjacent labels. In this model transition features are also derived from the words in and around the anchor text surrounding the link leading to the next state. Thus, a transition feature could be of the form

$fk(i, \mathbf{x}, y_i, y_{i-1}) = [[xi \text{ is an anchor word “advisor”, } y_i \text{ is “faculty”, and } y_{i-1} \text{ is “student”}]]$ .

A second option is to model each given label as a dual-state —one for the characteristics of the page itself (**page-states**) and the other for the information around links that lead to such a page (**link-states**). Hence, every path alternates between a page-state and a link-state. There are two advantages of this labeling. First, it reduces the sparsity of parameters by making the anchor word features be independent of the label of the source page. In practice, it is often found that the anchor text pointing to the same page are highly similar and this is captured by allowing multiple source labels to point to the same link state of label. Second for the foraging phase, it allows one to easily reason about intermediate probability of a path prefix where only the link is known and the page leading to it has not been fetched. In this model, the state-features of the page states are the same as in the previous model, the state features of the link states are derived from the anchor text. Thus, the anchortext transition features of the previous model, become state features of the link state. Thus the only transition features in this model are the edge features that capture the precedence order between labels.

## 5.2 Path Foraging

Given the trained sequential model  $M$  and a list of starting pages of websites, our goal is to find all paths from the list that lead to the Goal state in  $M$  while fetching as few unrelated pages. The key technical issue in solving this is to be able to score from the prefix of a path already fetched, all the outgoing links with a value that is inversely proportional to the expected work involved in reaching the goal pages. Consider a path prefix of the form  $P_1L_2P_3 \dots L_i$  where  $L_{i-1}$  is a link to page  $P_i$  in the path. We need to find for link  $L_i$  a score value that would indicate the desirability of fetching the page pointed to by  $L_i$ . This score is computed in two parts. First in section 5.2.1, we estimate for each state  $y$ , the proximity of the state to the Goal state. We call this the “reward” associated with the state. Then in section 5.2.2, we show for the link  $L_i$ , the probability of its being in state  $y$ .

### 5.2.1 Reward of a State

We apply techniques from Reinforcement Learning to compute the reward score using the CRF model learnt during path classification phase. Reinforcement Learning is a machine learning paradigm that helps in choosing the optimal action at each state to reach the Goal states. The Goal states are associated with rewards that start to depreciate as the Goal states get farther from the current state. The actions are chosen so as to maximize the cumulative discounted reward.

We apply Reinforcement Learning to compute the probability of a partially-observed sequence to end-up in a Goal state. Since we cannot predict the state sequence that would be followed by the unseen observation subsequence, we cannot compute the actual probability of the sequence ending in a Goal state. Instead, we estimate this probability based on the training data by learning a reward function  $R$  for each state. For each position  $i$  of a given sequence  $\mathbf{x}$  we estimate the expected proximity to the Goal state from a state  $y$   $R_i^{\mathbf{x}}(y)$  recursively as follows:

$$R_i^{\mathbf{x}} = \begin{cases} \frac{\sum_{y'} R_{i+1}^{\mathbf{x}}(y') e^{Wf(y', y, \mathbf{x}, i)}}{\sum_{y'} e^{Wf(y', y, \mathbf{x}, i)}}, & 1 < i < n \\ [[y = goal]], & i = n \end{cases}$$

When  $i = n$ , the reward is 1 for the Goal state and 0 for every other label. Otherwise, the values are computed recursively from the proximity of the next state and the probability of transition to the next state from the current state. We then compute a weighted sum of these positioned reward values to get position independent reward values. The weight are controlled via  $\gamma$ , a discount factor that captures the desirability of preferring states that are closer to the Goal state, as follows:

$$R^{\mathbf{x}}(y) = \frac{\sum_{k=0}^{n-1} \gamma^k R_{n-k}^{\mathbf{x}}(y)}{\sum_{k=0}^{n-1} \gamma^k}$$

where  $n$  is the length of the sequence. The final reward value of a state is computed by averaging over all training sequences  $\mathbf{x}_1 \dots \mathbf{x}_N$  as

$$R(y) = \frac{\sum_{l=1}^N R_l^{\mathbf{x}}(y)}{N}$$



### 5.2.3 Probability of being in a state

Consider a path prefix of the form  $P1L2P3 \dots Li$  where  $Li-1$  is a link to page  $Pi$  in the path. We need to find for link  $Li$ , the probability of its being in any one of the link-states. We provide a method for computing this. Let  $\alpha_i(y)$  denote the total weight of ending in state  $y$  after  $i$  states. For  $i > 0$ , this can be expressed recursively as with base case defined as  $\alpha_0(y) = 1$ .

$$\alpha_i(y) = \sum_{y' \in Y} \alpha_{i-1}(y') e^{W \cdot f(y', y, x, i)}$$

The probability  $L_i$  of being in a link state  $y$  is then defined as :

$$\frac{\alpha_i(y)}{\sum_{y' \in Y} \alpha_i(y')}$$

### 5.2.4 Score of Link

Finally, the score of a link  $Li$  after  $i$  steps is calculated as the sum of the product of reaching a state  $y$  and the static reward at state  $y$ .

$$Score(L_i) = \sum_y R(y) \frac{\alpha_i(y)}{\sum_{y' \in Y} \alpha_i(y')}$$

### 5.2.5 Algorithm to prioritize links

1. During training, for all training instances, compute  $R^x(y)$  for all  $y$  during the backward pass.
2. Average the  $R(y)$ -values computed in step 1 over all training instances
3. During Training,
  - a. Maintain a priority queue of links that lead from pages fetched. Since computation of score requires the  $\alpha$ -values, those are also maintained along with the link information.
  - b. In addition to the score and the  $\alpha$ -values, the  $\delta$ -values used to compute the label are also maintained.

- c. Initially, the queue contains the URL of seed page with score 0, and the  $\alpha$  and  $\delta$ -values are set to 1 and 0 respectively.
  4. For each seed URL in the priority queue,
    - a. Crawl the highest priority link to fetch the target page P.
    - b. Compute  $\alpha$ -values and  $\delta$ -values for the page P.
    - c. Label the state P with the maximizing label.
    - d. For every outlink from page P,
      - i. Calculate  $\alpha$ -values to compute the score
      - ii. Calculate  $\delta$ -values to label the link
      - iii. Enter the link in the priority queue
    - e. If more URLs are to be crawled then go back to step 4(a).

### 5.3 Experimental Results

The experiments were planned to be conducted in two phases: We first tested the accuracy of the CRF-based sequential classifier in distinguishing between positive and negative paths and segmenting a path. In the second phase we plan to use the trained model to fetch the pages in foraging mode.

#### 5.3.1 Dataset Description

Dataset was generated with labeled sequences for the course page extraction. This data set was used to train the CRF to recognize the path leading to course pages starting from the university homepages. The dataset was built manually by starting from the university homepage and then for each page visited by the user, recording the label for that page, the page link and the content of the page itself. Sequences were classified into two categories: positive, the ones that led to the goal page and negative: the ones that led to irrelevant pages. Training was performed on 122 sequences from 7 university domains. The training dataset included 78 positive and 44 negative sequences and the model was tested on 68 sequences. The test data included some sequences from domains that were not included in the training data. The results are shown in Table 2.

<u>Parameters</u>	<u>Dataset Description</u>
<u>#sites</u>	<u>7</u>
<u>#training Examples</u>	<u>122</u>
<u>#training Positives</u>	<u>78</u>
<u>#test Examples</u>	<u>68</u>
<u>#test Positives</u>	<u>43</u>
<u>#labels</u>	<u>26</u>
<u>#Features Learnt</u>	<u>12198</u>

Table 1: Dataset Description

	<u>Precision</u>	<u>Recall</u>
<u>Only Goal State</u>	<u>89.6%</u>	<u>92.4%</u>
<u>All States</u>	<u>79.2%</u>	<u>82.5%</u>

Table 2: Results for classification using CRFs. Classification accuracy was measured for Goal states and for all states separately.


### 5.3.2 Feature Extraction

When an HTML page is fetched, the page is represented in DOM structure format (<http://www.w3.org/DOM/>) using the Hypertext Parsing suite. The text content from the page is split into tokens on white-space delimiters (space, tab, etc.). The page-state tokens are extracted from the head and body fields of the HTML page, while the link-state tokens are collected from the URL anchor text and the neighbourhood text around the link. To capture the text around the anchor text of a hyperlink, we extracted tokens from a fixed-sized window before and after the link. In our experiments, we kept the window size constant at 10. In addition to these, the words from the relative part of the target URL and the target file extension are also included as tokens. As an illustration, for the URL “<http://www-2.cs.cmu.edu/~svc/papers/view-publications-ckl2004.html>”, the tokens extracted from the URL were ~svc, papers, view, publications,

ckl2004, and html. Since ‘~’ is a key feature that usually distinguishes homepages from departmental pages, we also add ‘~’ as a separate token.

Two other important features are extracted from the page:

1. Nested patterns are discovered by using Suffix tree matching. This is a helpful feature to point out pages which have a list structure in them. For example consider figure 9 which shows a faculty listing page. Such a page is often generated by database querying and thus the HTML used to represent each of the recurring elements is the same except the data. Our Suffix tree matching algorithm captures this repetition in the page and fires a feature if the page is found to contain such repetitive HTML elements. This feature is also found to fire in the Department listing page.
2. Headings in a page are also an indication of the type of page. We found that if a page is a faculty homepage then the headings in that page are generally from the set {Teaching, courses, Research Interest, Publications, Contact, etc.}. Our heading extraction algorithm parses the HTML structure of the page and extracts out the headings. If the extracted headings belong to the predefined set then the heading feature is fired. Figure 8 shows an example of heading extraction.



The image shows a screenshot of a faculty homepage with a light blue background and a dark blue vertical bar on the left. The page content is organized into sections, each with a heading in a red-bordered box. The 'Conferences' section lists several events and a journal. The 'Address' section provides contact information. The 'Telephone' section includes a phone icon and contact numbers. The 'Research Interests' section describes the author's focus and provides links to publications and a research group. The 'Courses' section is partially visible at the bottom.

**Conferences**


Some conferences and journals I am currently involved with.

- [ACM CCS '07](#)
- [Usenix Woot '07](#)
- [Usenix Security '07](#)
- [NDSS '07](#)
- [Usable Security Workshop \(USEC'07\)](#)
- Editor of [Journal of Cryptology](#).

---

**Address**

- Mail: Computer Science Dept., Gates 475, Stanford, CA, 94305-9045
- Office: CS Building, Gates 475.
- [Directions](#) to the Gates building.

**Telephone** 

- Office: (650) 725-3897
- Fax: (650) 725-4671

My [PGP](#) key.

---

**Research Interests**

My main research focus is on applied cryptography, and network security.

Here is a list of my [publications](#) and current [students and research group](#).

Take a look at our [Security Lab](#). We are also running a biweekly [security seminar](#).

**Courses**

Figure 8: An example of heading extraction from a faculty homepage

The screenshot shows a web browser window displaying the Stanford Computer Science Faculty page. The page has a red header with the Stanford logo and navigation links. Below the header, there is a section for 'Faculty' with links to 'Regular Faculty', 'Courtesy Faculty', 'Consulting Faculty', 'Visiting and Acting Faculty', and 'Lecturers'. The 'Regular Faculty' section contains a table with the following data:

Name	Phone	Office	Email (add at cs.stanford.edu to the end)
<a href="#">Alex Aiken</a>	5-3359	GATES 411	
<a href="#">Serafim Batzoglou</a>	3-3334	Clark S266	
<a href="#">Gill Bejerano</a>	650 723-7666	Beckman B321	<a href="#">Click here</a>
<a href="#">Dan Boneh</a>	5-3897	GATES 475	
<a href="#">David Cheriton</a>	3-1131	GATES 439	
<a href="#">Bill Dally</a>	5-8945	GATES 301	
<a href="#">David Dill</a>	5-3642	GATES 344	
<a href="#">Dawson Engler</a>	3-0762	GATES 314	
<a href="#">Ron Fedkiw</a>		GATES 207	
<a href="#">Edward Feigenbaum</a>	3-4878	GATES 237	
<a href="#">Richard Fikes</a>	5-3860	GATES 246	
<a href="#">Hector Garcia-Molina</a>	3-0685	GATES 434	
<a href="#">Mike Genesereth</a>	3-0934	GATES 220	
<a href="#">Gene Golub</a>	3-3124	GATES 280	
<a href="#">Leonidas Guibas</a>	3-0304	GATES 375	
<a href="#">Patrick Hanrahan</a>	3-8530	GATES 370	
<a href="#">John Hennessy</a>	3-2481	BLDG 10	
<a href="#">Mark Horowitz</a>	5-3707	GATES 306	
<a href="#">Oussama Khatib</a>	3-9753	GATES 144	
<a href="#">Scott Klemmer</a>	3-3692	Gates 384	
<a href="#">Don Knuth</a>	723-4367	GATES 477	
<a href="#">Daphne Koller</a>	3-6598	GATES 142	
<a href="#">Vladlen Koltun</a>	723-6690	Gates 464	
<a href="#">Christos Kozyrakis</a>	5-3716	GATES 304	

Figure 9: A page having a nested HTML structure in the form of faculty information

### 5.3.3 Experimental platform

We performed our experiments on the MALLET [14] toolkit. The basic implementation of CRF is provided in the default package. We modified the basic implementation to calculate the reinforcement learning values for each state. We also modified the forward backward algorithm to calculate the  $\alpha$  values required for calculating the link score. The focused crawler has been implemented from scratch in Perl. The implementation manages the queue of urls by using MYSQL database integration with the perl code.

## References

[1] *Wikipedia* : <http://www.wikipedia.com>

[2] J. Cole and H. Foster, *Using Moodle: Teaching with the Popular Open Source Course Management System* (O'Reilly Media Inc., 2007).

[3] S. Chakrabarti, *Mining the Web: Discovering Knowledge from Hypertext Data* (Morgan-Kauffman, 2002).

[4] S. Chakrabarti, M.H. Van den Berg, and B.E. Dom, "Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery," *Computer Networks*, vol. 31, nos. 11–16, pp. 1623–1640.

[5] Yahoo Directory:

[http://dir.yahoo.com/Education/Higher\\_Education/Colleges\\_and\\_Universities/](http://dir.yahoo.com/Education/Higher_Education/Colleges_and_Universities/)

[6] Lawrence R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77(2), pages 257–286, February 1989.

[7] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001

[8] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[9] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *International Conference on Machine Learning*, 2000.

- [10] D. Pinto, A. McCallum, X. Wei, and W. B. Croft. Table extraction using conditional random fields. Proceedings of the ACM SIGIR, 2003.
- [11] F. Sha and F. Pereira. Shallow parsing with conditional random fields. Proceedings of Human Language Technology, NAACL 2003, 2003.
- [12] J. Wang and F.H. Lochovsky, "Wrapper Induction Based on Nested Pattern Discovery," Technical Report HKUST-CS-27-02, Dept. of Computer Science, Hong Kong, Univ. of Science & Technology, 2002.
- [13] V.G.Vinod Vydiswaran and Sunita Sarawagi, *Learning to extract information from large websites using sequential models*(In COMAD, 2005. SIGKDD Explorations. Volume 6, Issue 2 - Page 66)
- [14] McCallum, Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit." <http://mallet.cs.umass.edu>. 2002.
- [15] Hongyu Liu , Evangelos Milios , Jeannette Janssen, Probabilistic models for focused web crawling, Proceedings of the 6th annual ACM international workshop on Web information and data management, November 12-13, 2004, Washington DC, USA
- [16] J. Rennie and A. McCallum, "Using Reinforcement Learning to Spider the Web Efficiently," In proceedings of the 16th International Conference on Machine Learning(ICML-99), pp. 335-343, 1999.
- [17] G. Pant and P. Srinivasan, Learning to crawl: Comparing classification schemes, ACM Transactions on Information Systems 23(4) (2005) 430-462.
- [18] Hongyu Liu, Jeannette C. M. Janssen, Evangelos E. Milios: Using HMM to learn user browsing patterns for focused Web crawling. Data Knowl. Eng. 59(2): 270-291 (2006)

- [19] Junghoo Cho , Hector Garcia-Molina , Taher Haveliwala , Wang Lam , Andreas Paepcke , Sriram Raghavan , Gary Wesley, Stanford WebBase components and applications, ACM Transactions on Internet Technology (TOIT), v.6 n.2, p.153-186, May 2006
- [20] S. Chakrabarti, K. Punera, and M. Subramanyam, "Accelerated Focused Crawling through Online Relevance Feedback," Proc. 11th Int'l World Wide Web Conf. (WWW 02), ACM Press, 2002, pp. 148–159.
- [21] Marc Ehrig , Alexander Maedche, Ontology-focused crawling of Web documents, Proceedings of the 2003 ACM symposium on Applied computing, March 09-12, 2003, Melbourne, Florida.
- [22] C.C. Aggarwal, F. Al-Garawi, and P. Yu, "Intelligent Crawling on the World Wide Web with Arbitrary Predicates," Proc. 10th Int'l World Wide Web Conf. (WWW 01), ACM Press, 2001, pp. 96–105.
- [23] Srinivasan P, Mitchell JA, Bodenreider O, Pant G, Menczer F. Web Crawling Agents for Retrieving Biomedical Information. Proc. of the International Workshop on Bioinformatics and Multi-Agent Systems (BIXMAS). 2002 Jul.
- [24] Wallach, H.M.: Conditional random fields: An introduction. Technical Report MS-CIS-04-21, University of Pennsylvania (2004)
- [25] Sutton, C., McCallum, A.: An Introduction to Conditional Random Fields for Relational Learning. In "Introduction to Statistical Relational Learning". Edited by Lise Getoor and Ben Taskar. MIT Press. (2006)
- [26] McCallum, A.: Efficiently inducing features of conditional random fields. In: Proc. 19th Conference on Uncertainty in Artificial Intelligence. (2003)



[27] Andrew McCallum , Dayne Freitag , Fernando C. N. Pereira, Maximum Entropy Markov Models for Information Extraction and Segmentation, Proceedings of the Seventeenth International Conference on Machine Learning, p.591-598, June 29-July 02, 2000

[28] SCROM: <http://en.wikipedia.org/wiki/SCORM>

[29] Learning Technology Standards Committee (2002), [\*Draft Standard for Learning Object Metadata. IEEE Standard 1484.12.1\*](#), New York: Institute of Electrical and Electronics Engineers.

[30] “Web surpasses one billion documents: Inktomi/NEC press release.” available at <http://www.inktomi.com>, Jan 18 2000.

[31] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, “Building domain-specific search engines with machine learning techniques,” in *Proc. AAAI Spring Symposium on Intelligent Agents in Cyberspace*, 1999.

[32] M. Diligenti et al., "Focused Crawling Using Context Graphs," *Proc. 26th Int'l Conf. Very Large Data Bases (VLDB 2000)*, Morgan Kaufmann, 2000, pp. 527–534.

[33] O. Heinonen, K. Hatonen, and K. Klemettinen, “WWW robots and search engines.” Seminar on Mobile Code, Report TKO-C79, Helsinki University of Technology, Department of Computer Science, 1996.

[34] P.D. Bra, R. Post, Information retrieval in the World Wide Web: making client-base searching feasible, in: Proceedings of the 1<sup>st</sup> International WWW Conference, Geneva, Switzerland, 1994.

[35] M. Hersovici, M. Jacovi, Y. Maarek, D. Pelleg, M. Shtalhim, S. Ur, The Shark-search algorithm—an application: tailored Web site mapping, in: Proceedings of the 7th International WWW Conference, Brisbane, Australia, 1998.

- [36] J. Cho, H. Garcia-Molina, L. Page, Efficient crawling through URL ordering, in: Proceedings of the 7th World Wide Web Conference, Brisbane, Australia, 1998.
- [37] K. Stamatakis, V. Karkaletsis, G. Paliouras, J. Horlock, et al., Domain-specific Web site identification: the CROSSMARC focused Web crawler, in: Proceedings of the 2nd International Workshop on Web Document Analysis (WDA2003), Edinburgh, UK, 2003.
- [38] J. Johnson, K. Tsioutsoulis, C.L. Giles, Evolving strategies for focused Web crawling, in: Proceedings of the 20th International Conference on Machine Learning (ICML-2003), Washington, DC, USA, 2003.
- [39] F. Menczer, G. Pant, P. Srinivasan, M. Ruiz, Evaluating topic-driven Web crawlers, in: Proceedings of the 24th Annual International ACM/SIGIR Conference, New Orleans, USA, 2001.
- [40] F. Menczer, G. Pant, P. Srinivasan, Topical Web crawlers: evaluating adaptive algorithms, ACM TOIT 4 (4) (2004) 378–419.
- [41] P. Srinivasan, F. Menczer, G. Pant, A general evaluation framework for topical crawlers, Information Retrieval 8 (3) (2005) 417–447.
- [42] G. Pant, K. Tsioutsoulis, J. Johnson, C. Giles, Panorama: extending digital libraries with topical crawlers, in: Proceedings of ACM/IEEE Joint Conference on Digital Libraries (JCDL 2004), Tucson, Arizona, June 2004, pp. 142–150.
- [43] Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02), 2002.

[44] Lawrence R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77(2), pages 257–286, February 1989.

[45] Intinno: <http://www.intinno.com>