

A solution to the exact match on rare item searches^{*}

Introducing the lost sheep algorithm.

Morten Goodwin
Tingtun AS
PO Box 48
N-4791 Lillesand, Norway.
morten.goodwin@tingtun.no

ABSTRACT

This paper proposes an approach for finding a single web page in a large web site or a cloud of web pages. We formalize this problem and map it to the exact match on rare item searches (EMRIS). The EMRIS is not much addressed in the literature, but many closely related problems exist. This paper presents a state-of-the-art survey on related problems in the fields of information retrieval, web page classification and directed search.

As a solution to the EMRIS, this paper presents an innovative algorithm called the lost sheep. The lost sheep is specifically designed to work in web sites with of links, link texts and web pages. It works as a pre-classifier on link texts to decide if a web page is candidate for further evaluation.

This paper also defines sound metrics to evaluate the EMRIS. The lost sheep outperforms all comparable algorithms both when it comes to maximizing accuracy and minimizing the number of downloaded pages.

Categories and Subject Descriptors

H.4.0 [Information Systems Applications]: General; I.5.m [Pattern Recognition]: Miscellaneous

General Terms

Algorithms, Performance, Experimentation

Keywords

Crawling, Classification, Search

^{*}Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. WIMS'11, May 25-27, 2011 Sogndal, Norway Copyright ©2011 ACM 978-1-4503-0148-0/11/05... \$10.00

1. INTRODUCTION

The amount of information available online on public web sites is increasing drastically. Many public web sites host huge amount of information on diverse topics, often in dynamic environments. Furthermore, the concept of web sites is becoming less important as it is neither clear nor important for most users where one web site starts and another ends. In these highly dynamic web sites, it is not possible to keep all information indexed at all times, which is required for traditional crawlers and search engines. Additionally, for copyright or privacy reasons, many web site owners deliberately restrict crawlers from visiting part of their web site using the robot exclusion protocol [1].

The consequence of these emerging properties of the public web is that large parts of published information falls into the deep web and unindexable dynamic clouds and cannot be found by crawlers nor indexed by major search engines. This leads to navigation challenges for users even when the users know the area they are looking for, and critical web pages could be very difficult find.

This paper is on finding a single web page in a web site when only one page is relevant. This is a specifically challenging area of information retrieval named the exact match on rare item searches [2], referred to in this paper as **EMRIS**.

In web page classification algorithms, web pages are often treated as single isolated entities. However, web pages have more information available apart from what is available in the page itself. Each page is part of a structured web site. Literature shows that web pages which link to each other have a semantic correlation. For example, it is more likely that a web page about sport will link to another web page about sport, than linking to a web page about the financial market [3]. Similarly, web pages within the same topics tend to link to each other [4]. Hence for web page classification it is valuable information that one web page links to another. This information is seldom utilized in web classification algorithms.

The paper is organized as following. The EMRIS is not much addressed in literature, but a survey introducing EMRIS and related algorithms are presented in section 2. Section 3 presents the lost sheep a novel approach to the EMRIS. No sound metric to evaluate EMRIS is available in the literature and section 4 presents metrics from related fields and based on these defines metrics for EMRIS. The empirical results

are presented in 5. Finally the conclusion and future work are presented in 6 and 7.

2. RELATED WORK

This section presents the state-of-the-art on how web sites are modeled, on finding single web pages in web sites, and relevant algorithms within each field. The algorithms presented here are also implemented and the corresponding results presented in section 5.1.4.

The terms used in the following algorithm descriptions in some cases differ from the original cited paper. This is intentionally done to make a consistent description throughout the paper.

2.1 Website / directed graph

Web sites are often modeled as graphs of nodes connected with directed edges [5, 6, 7]. This is done to enable analytic analysis and controlled empirical experiments on the web sites and algorithms interacting with the web sites. In these models, each web page is represented as a node, and the links between the web pages are represented as directed edges between the nodes. In addition, web pages have link texts that intend to informatively describe corresponding web pages with short texts [5]. Link texts are not addressed in the directed graph models. Because of this, this paper extends the existing models with labels containing natural language text connected to each edge.

In this paper, a web site is modeled as a directed graph of nodes connected with directed edges and labels $S_i = \{\mathbf{P}_i, \mathbf{E}_i, \mathbf{L}_i\}$. Each web page is represented as a node p_j . N is the number of nodes in the directed graph S_i so that $\mathbf{P}_i = \{p_0 \dots p_N\}$.

The nodes are connected together with directed edges so that edge $e_{j,k} \in \mathbf{E}_i$ is a link from node p_j to p_k . Each edge has a corresponding label so that $l_{j,k} \in \mathbf{L}_i$ is a natural language label connected to $e_{j,k}$ and observable from node p_j .

This is in line with existing models of web sites in the literature, but is extended with labels [5, 6, 7] and is empirically shown to represent real web sites (see section 5.1). Table 1 presents the comparisons between the directed graph model and read web sites.

Web site model	Notation	Real web sites
Node	p_j	Web page
Edge	$e_{j,k}$	Link
Label	$l_{j,k}$	Link text
Directed graph	S_i	Web site

Table 1: Mapping between information retrieval and web sites.

Figure 1 shows an example of a simple web site s_1 modeled as a directed graph. s_1 consists of four web pages p_s, p_t, p_1, p_2 . The web page links consist of the directed edges $e_{s,t}, e_{s,1}, e_{s,2}$ and the corresponding labels $l_{s,t}, l_{s,1}, l_{s,2}$. These labels are observable from the starting node, p_s , with out following the any of the edges.

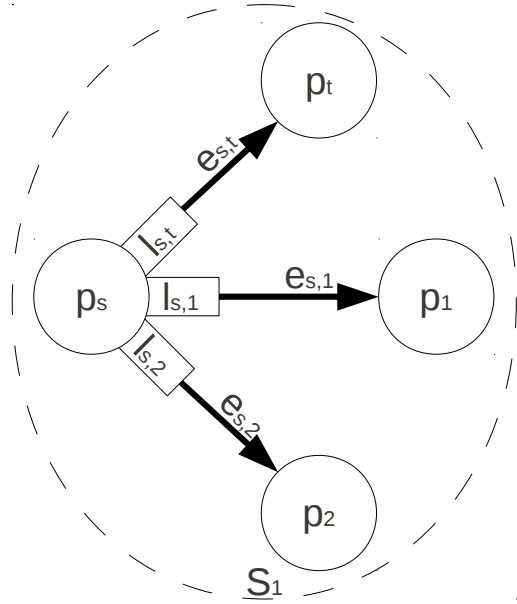


Figure 1: Web site example.

2.2 Exact match on rare item searches (EMRIS)

The EMRIS is a problem introduced by [2] focusing on finding a single web page in a large set of web pages, such as a web site or cloud of web pages. It was introduced for the following reasons; It is particularly challenging to find a single node in a large directed graph, and the objectivity of approaches that classify large amounts of data were questioned. The EMRIS has not previously been formally defined.

When a classification approach is developed with the intention to classify large amount of data, it is not practically possible to first manually classify all data to perform controlled experiments. The common approach is therefore that the classifier is run with a relatively small set of manually classified training data. Subsequently, the classification results are manually verified. Unfortunately, certifying the results is not a deterministic approach. People tend to (sub-consciously) give a unfair advantage to their own algorithms. In contrast, with the EMRIS, there is only one relevant node and this can easily be defined prior to conducting experiments. Thus, as no manual verification of the results are needed, the objectivity of the algorithms can no longer be questioned.

A practical example of the EMRIS could be to find a target web page, p_t , with contact information, such as physical address and phone number. Neither the location nor properties of p_t are known, only that the content of p_t is related to contact information. To identify contact information let q be the input query $q = [\text{contact}, \text{about}, \text{phone}, \text{address}]$.

Let a be a hypothetical algorithm with the query (q), directed graph (s_1 in figure 1) and starting node (p_s) as input, and the expected target node as output: $a(q, s_1, p_s) \in \mathbf{P}$. Using the directed graph s_1 , a first downloads and parses

node p_s . a observes that p_s has the edges $e_{s,t}, e_{s,1}, e_{s,2}$ and corresponding labels $l_{s,t}, l_{s,1}, l_{s,2}$.

a knows, from examining the edges $e_{s,t}, e_{s,1}, e_{s,2}$, that the nodes p_t, p_1, p_2 exists. However, the content of p_t, p_2 and p_3 are unknown. The goal is therefore to find which of p_t, p_1 and p_3 have contact information and is the target node.

Since p_s has been observed, the following could be known and conclusions drawn:

p_t : A node with the edge $e_{s,t}$ (from p_s to p_t) is represented with the label $l_{s,t} = \text{"About us"}$. Since $\text{"about"} \in q$, a may conclude that p_t is likely to contain to contact information.

p_1 : A node with the edge $e_{s,1}$ (from p_s to p_1) is represented with the label $l_{s,1} = \text{"Contact us"}$. $\text{"contact"} \in q$, a may conclude that p_1 is likely to contain contact information.

p_2 : A node with the edge $e_{s,2}$ (from p_s to p_2) is represented with the label $l_{s,2} = \text{"Image Gallery"}$. Since neither $\text{"image"} \in q$ nor $\text{"gallery"} \in q$, a may conclude that p_2 is not likely to have contact information.

Most web page classification algorithms use equal amount of resources on p_t , p_2 and p_3 . However, keeping in mind that the labels describe the corresponding nodes [8], it is more likely that either p_t or p_1 is the target node. Thus, since the aim is to find the target node and it is significantly more costly to download a node than doing a memory operation [9], it makes sense to focus most of the available resources on what is likely to be the target nodes, namely p_t and p_1 , instead of p_2 .

To continue, let a download p_t and p_1 . By comparing the content of p_t and p_1 with q , a may conclude that p_t matches q best. a then returns p_t as the node most probable to be the target node.

[2] introduced three algorithms to solve the EMRIS: similarity search (SIM), degree based search (DEG) and Similarity*Degree (SIMDEG).

2.2.1 Similarity based search (SIM)

The Similarity Based Search (SIM) compares the input query q to the content of neighboring nodes, and follows the node which is most similar to q , SIM was used in [2] for EMRIS.

For each node p_j the SIM follows all edges available from p_j , $\{e_{j,0}, \dots, e_{j,n}\}$. It calculates a similarity score between q and all nodes available from $\{e_{j,0}, \dots, e_{j,n}\}$. The algorithm pursues the node with the largest calculated similarity score.

For SIM to work, the algorithm needs to be pulled towards the target node p_t . To pull SIM towards p_t the nodes which lead towards p_t should have a higher similarity score than remaining nodes. Since p_t should be similar to q , it will have a high similarity score. Similarly, nodes close to p_t (parent nodes) will be similar to p_t and thus also have a high similarity score. However, grandparent nodes (nodes

linked to parent nodes of p_t) have been shown not to be similar to p_t [3, 4]. Thus we can expect high similarity score of parent nodes, but not of grandparent or any other nodes.

This means, until the SIM reaches one of the parent nodes, the nodes will not be similar to q and the similarity score will not help the SIM towards p_t . As a result, when the algorithm is at a node far away from the target node, which is expected in the beginning of a crawl, it will walk at random. This problem is not addressed [2].

2.2.2 Degree based search (DEG)

A Degree Based Search (DEG) uses the number of edges from neighboring nodes to guide the crawler. DEG was used in [2] for the EMRIS.

The DEG approach is similar to the SIM. However, instead of using the similarity score between q and the nodes, DEG pursues the neighboring node with the largest number of edges. Thus, it assumes that p_t is more likely to be linked from a node with many edges compared to a node with few.

2.2.3 Similarity*Degree Search (SIMDEG)

The Similarity*Degree Search (SIMDEG) combines SIM and DEG. The SIMDEG was used in [2] for the EMRIS.

[2] showed some advantages and disadvantages of the DEG and the SIM; The DEG assumes that p_t is more likely to be linked from a node with many edges, and SIM only uses the content of the nodes and q to calculate which are most similar without using the edges.

To utilize the advantages of both the SIM and the DEG it was combined as the SIMDEG. For each neighboring node, the SIMDEG calculates the similarity score as well as a frequency of edges. The algorithm pursues the node with the highest similarity score times frequency of edges.

In practice this means that in the beginning a crawl, when the algorithm is far away from p_t , the frequency of edges will be larger than the similarity score and the SIMDEG will be guided towards the nodes with the largest number of edges. When a node p_j with an edge towards p_t is reachable, the similarity score of p_j will increase and be larger than the frequency of edges. In this situation the SIMDEG will crawl towards p_j which is close to p_t .

It is therefore expected that the SIMDEG outperforms both the SIM and the DEG, which was shown in [2].

2.3 Web page classification and surrounding pages

The state-of-the-art classification algorithms for web pages originates from text classification algorithms. Since web pages are mostly text, it is natural that most web page classification algorithms are based on text classification algorithm.

Some algorithms utilize the tagged semantic information available in web pages such as title, heading information etc. Others use information available outside of the actual web page in a classification approach.

[10] used a portion of the parent web page (web page linking to the page to be classified). This approach works so that for each page p_j to be classified, a set of neighboring pages (pages linked to and from p_j) should be downloaded. The approach distinguishes between trusted and not trusted links whereas a trusted link is a link similar to the target page.

Similarly, several approaches have been attempted in using the link information for classification of web pages [11, 12]. The approaches still centers around a web page and try to find the corresponding in-links, as following. For every web page p_j , search other pages that link to p_j and use the descriptive link text.

Others have used the link structure and link text for analyzing web sites [9, 13]. These approaches focus on classifying complete web sites, not individual web page.

The approaches further requires that significant parts of the web sites, if not the entire web sites, to be downloaded and processed, which is impractical and often impossible [14].

The above shows that in-links are powerful for web mining. However, the approaches assumes knowledge of in-links to web pages. Knowing all in-links to a page within a web site requires significant resources [14]. Knowing all in-links from other web sites is in practice impossible. A practical solution would be to use search engines such as Google for finding in-links.

[8] introduces an approach which utilizes the topical correlation between linked web pages and takes advantage of the links to other pages for describing the current page. Hence, the approach shows that even though the links on a page p_j are to other pages, it can still be used to describe the content of the page p_j .

Hence, existing web page classification approaches do not consider that web pages which are structured together can be classified in a sequential manner, such as is common for focused crawling where the link descriptions are used to decide if web pages should be downloaded. Similarly, no web page classification approach has been implemented with a pre-classification on the link texts to decide if a page should be downloaded.

Despite evidence that given a web page p_j , the pages linking to p_j (parent pages) can be useful for classification of p_j , pages which link to parent pages of p_j (grandparent pages) are not useful. This is because they are most often not topically related to p_j [3, 4].

Traditional classification algorithms are not created to work in directed graphs. Thus, they do not have functionality to prioritize which nodes to visit.

Note that main purpose of this paper is not to find which classification algorithms work best, but find a practical solution to the EMRIS. Because of this, only a few algorithms are presented in this section.

2.3.1 Hidden Markov Model Classifier (HMM)

This section presents a specific type of the HMM called the Learning Automata (LA). More specifically LA presented here is a two action multiple state Tsetlin Learning Automata. This is an LA with no absorbing states, and is accordingly ergodic supporting dynamic environments, such as web sites [15, 16].

This LA could be used as a binary classifier with the two actions $A = \{\beta_1, \beta_2\}$ and S states. The model has a beginning state $B \in S$ and a quit state $Q \in S$ as can be seen in figure 2.

Each LA is connected to a node and has the label text and node text available. Initially, stemming is performed on the label and node text. The state transitions follow a reward penalty system. Whenever a word matches q $s_j(t+1) := s_j(t) - 1$, $s_j(t+1) := s_j(t) + 1$ otherwise. Thus, whenever the classifier detects a word which was not in q , the model moves one state closer to the quit state Q . Similarly, whenever the classifier detects a word which was in the query, the model moves one state closer to the state B . Thus, the LA closest to the Q state is most probably connected to the target node and the LA closest to the B is least likely to be the target node.

The transitions are done by comparing the detected words with q and number of comparisons is counted by t . Note that t is not used by the LA, but by the HMM Twofold (section 2.3.2) and the LS (section 3).

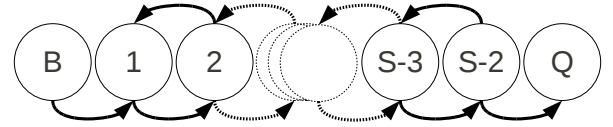


Figure 2: Hidden Markov Model

2.3.2 Hidden Markov Model Classifier Two Fold (HMM2F)

By using the HMM it is often the choice between accuracy and speed. By choosing many hidden states, the classifier will be able to reach high accuracy but will use a many transactions to do so. In contrast, by using few hidden states, it is possible to quickly converge but will not reach a high accuracy. [17].

Instead of letting the HMM move in a linear space, it has been shown that by letting the number of states moved in each transaction vary, quick convergence speed and high accuracy can be reached. This is done so that if $t = 0$, it moves $S/2$ states, if $t = 1$ it moves $S/4$ states and so on [18].

2.4 Focused crawling

A focused crawler is a crawler that aims at downloading web pages of a specific predefined topic. The aim is to harvest as much as possible of relevant information, maximizing the harvest rate and minimizing the total recall (see section 4.3).

There are some clear similarities between focused crawl and an EMRIS, namely to find a subset of nodes in a directed graph based on relevance.

However, in contrast to the EMRIS, it is of no interest for

a focused crawler to only find one node with relevant information as this would not maximize the harvest rate nor minimize the loss rate [19, 20].

2.4.1 Content based focused crawler using query input

There are many different strategies used for focused crawling. Of these, the content based crawlers have been shown most efficient [21].

Most focused crawlers use training data specifically designed for focused crawling or user input. Unfortunately, this is not available in the directed graph (see section 2.1).

[22] presented a content based focused crawler using query keywords as input. The crawler basis itself around a queue of nodes prioritized according to a satisfactory number. The satisfactory number for a node p_j is calculated as the number of relevant nodes with edges linking towards p_j . A node is relevant if the content has at least one word in common with the input query q .

For this approach to work, there has to be knowledge of the nodes linking to p_j . This is not possible from the beginning of a crawl. Thus, the approach assumes knowledge of the directed graph which is not available. This is even acknowledged by [22] which assume that nodes have been downloaded prior to the crawl.

3. THE LOST SHEEP ALGORITHM (LS)

This section proposes a new innovative algorithm for the EMRIS, named the lost sheep (LS) after story which it resembles. The LS is not a classifier, but a classification framework for the EMRIS. It can be seen as a pre-classifier on labels to decide if a node is candidate for further evaluation and should be downloaded.

The algorithm works as following. A herder has a number of sheep which runs out on a web site. The herder releases all sheep at the same time. Each sheep runs a for while and then returns back to the herder. A sheep which discovers something interesting will spend longer time than a sheep which did not discover anything interesting. The last sheep to return to the herder, the lost sheep, has discovered the most interesting information. The herder, which has perfect knowledge of which nodes the sheep found, can then choose to either accept the node the lost sheep found as the most relevant node, or move to the node the lost sheep found and repeat the process. The LS is formally described in section 3.1.

Figure 3 presents an example of the lost sheep algorithm. The figure shows a shepherd releasing 5 sheep $\{x_1, \dots, x_5\}$. x_2, x_3 and x_5 examines the labels and choose not to pursue the node further. However, x_1 and x_4 choose to following their corresponding edge further, which means that either x_1 or x_4 finds the most interesting information and will be the lost sheep.

Note that the duration is typically not duration in time, as this could be biased, but a number indicating the level of interesting information found.

This has the advantage that the algorithm does not spend resources on classification of nodes which can, within a reasonable certainty, say is not correct. Furthermore, the resources are spent on the parts of the directed graph which has the potential to be the target node. By nature, the algorithm spends resources on parts of the web site which has the potential to be correct. Or, in other words, the parts of the web site closest to the correct solution.

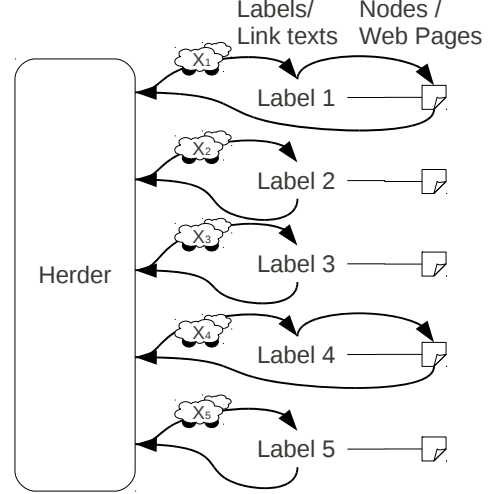


Figure 3: Lost sheep algorithm

The authors are of course aware of the “no free lunch” theorem which states that classifiers cannot learn what is not available in the training data and that there is no algorithm that works best for all cases [23]. This means that which algorithm that will work best depends on the properties of the training data and available features. Because of the “no free lunch” theorem, the LS can be generalized to use any classification algorithm which can interact with web sites. For practical reasons this paper only presents a specific version of the lost sheep using a hidden Markov model classifier (described in section 2.3.1). This description can be used directly as an implementation specification.

3.1 Lost sheep description

This section presented the specific LS using a hidden Markov model classifier. (For details on the hidden Markov model classifier see section 2.3.1.) This works on a single directed graph S_i .

- Let the input query be a list of words: q .
- Let a threshold indicate when the algorithm should stop be set to: $\text{threshold} \in [0, 1]$.
- Let a max depth to prevent infinite loops be set to $\text{maxdepth} > 0$.
- Let $f(x_j) = t$ at all times, and let $t := 0$ before LS starts.

Lost sheep with a hidden Markov model classifier

- 1 Let p_s be the start node with the edges $E = \{e_{s,0} \dots e_{s,n}\}$ and corresponding labels $L = \{l_{s,0} \dots l_{s,n}\}$ directly observable, where n is the number of edges from p_s .
 - 2 The herder releases n sheep $X = \{x_0, \dots, x_n\}$.
 - 3 for each $x_j \in X$
 - 3.1 Extract the edge label $l_{s,j}$.
 - 3.2 Apply stemming to $l_{s,j}$.
 - 3.3 Remove stop words from $l_{s,j}$.
 - 3.4 Split $l_{s,j}$ into a list of words: $W_{s,j}$
 - 3.5 for each word $w_{s,j,k} \in W_{s,j}$
 - 3.5.1 $t := t + 1$
 - 3.5.2 if $w_{s,j,k} \in q$, $x_j(t+1) := x_j(t) + 1$, otherwise $x_j(t+1) := x_j(t) - 1$.
 - 3.5.3 if $x_j(t) = Q$, $f(x_j) := t$, return.
 - 3.6 Extract the web page text p_j .
 - 3.7 Apply stemming to p_j .
 - 3.8 Remove stop words from p_j .
 - 3.9 Split p_j into a list of words: W_j
 - 3.10 for each word $w_{j,k} \in W_{j,k}$
 - 3.10.1 $t := t + 1$
 - 3.10.2 if $w_{j,k} \in q$, $x_j(t+1) := x_j(t) + 1$, otherwise $x_j(t+1) := x_j(t) - 1$.
 - 3.10.3 if $x_j(t) = Q$, $f(x_j) := t$, return.
 - 4 numruns := numruns + 1
 - 5 Find $x_{\text{best}} \in X$ where $f(x_{\text{best}}) = \arg \max_{x_j} f(x_j)$
 - 6 If $\frac{f(x_{\text{best}})}{\sum_0^Q f(x_j)} < \text{threshold}$ and numruns < maxdepth, then $p_s := x_{\text{best}}$ and goto 1.
-

x_{best} is chosen as the sheep with the most likely to have the correct solution.

4. PERFORMANCE METRICS

No satisfactory metrics for EMRIS exists. This section defined metrics for EMRIS based using both existing EMRIS metrics [2] and widely accepted metrics from relevant fields, namely: web page classification, information retrieval and focused crawling.

4.1 Classification

Web classification algorithms aim at maximizing the number of pages classified correctly (True Positives TP, and True Negative TN) as well as minimizing the number of web pages wrongly classified (False Positives FP, and False Negative FN).

Classification algorithms are typically evaluated according to macro-precision, $P_m(S_i)$ and macro-recall $R_m(S_i)$ for the binary classification problems as following [24].

$$P_m(S_i) = \frac{1}{2} \left(\sum_{i=1}^2 \frac{TP_i}{(TP_i + FP_i)} \right) \quad (1)$$

$$R_m(S_i) = \frac{1}{2} \left(\sum_{i=1}^2 \frac{TP_i}{(TP_i + FN_i)} \right) \quad (2)$$

For $P_m(S_i)$ and $R_m(S_i)$ to be applicable, the EMRIS can written as a binary classification problem

1. Classify the target node p_t as relevant (P_m^1 and R_m^1).
2. Classify all $N - 1$ non-relevant nodes ($\{p_j \in \mathbf{P} : p_j \neq p_t\}$) as non-relevant (P_m^2 and R_m^2).

Note that i in equation 1 and 2 refers to the different classification problems. For EMRIS, one of the classification problems is to classify the relevant node as relevant (Problem 1: $P_m^1, R_m^1, TP_1, TN_1, FP_1$ and FN_1) the other is to classify all not relevant nodes as not relevant (Problem 2: $P_m^2, R_m^2, TP_2, TN_2, FP_2$ and FN_2). To be able to simplify the algorithms, $P_m(S_i)$ is split into $P_m^1(S_i), P_m^2(S_i)$ and $R_m(S_i)$ into $R_m^1(S_i), R_m^2(S_i)$.

$$\begin{aligned} P_m(S_i) &= \frac{1}{2} \left(\frac{TP_1}{(TP_1 + FP_1)} + \frac{TP_2}{(TP_2 + FP_2)} \right) \\ &= \frac{1}{2} \left(P_m^1(S_i) + P_m^2(S_i) \right) \end{aligned} \quad (3)$$

$$\begin{aligned} R_m(S_i) &= \frac{1}{2} \left(\frac{TP_1}{(TP_1 + FN_1)} + \frac{TP_2}{(TP_2 + FN_2)} \right) \\ &= \frac{1}{2} \left(R_m^1(S_i) + R_m^2(S_i) \right) \end{aligned} \quad (4)$$

Since the EMRIS is on finding only one node, there is, in contrast to most other binary classification problems, only two possible scenarios for EMRIS; The correct node has been found (section 4.1.1) or the correct node has not been found (section 4.1.2). This means that even though both metrics $P_m(S_i)$ and $R_m(S_i)$ are relevant for binary classification problems, it could be simplified for EMRIS.

Section 4.1.1 and 4.1.2 presents all possible outcomes of $P_m(S_i)$ and $R_m(S_i)$ for EMRIS. Using all possible outcomes, section 4.1.3 presents simplified $P_m(S_i)$ and $R_m(S_i)$.

4.1.1 The correct node has been found

When a correct node has been found, it means that p_t is correctly classified as relevant. This gives $TP_1 = 1$, $FP_1 = 0$ and $FN_1 = 0$.

Keeping in mind that there is only one relevant node in EMRIS, when the correct node has been found it means that that all not relevant $N - 1$ nodes have also been correctly classified as not relevant. This gives that $TP_2 = N - 1$, $FP_2 = 0$ and $FN_2 = 0$.

Using equations 3 and 4 gives:

$$\begin{aligned} P_m^1(S_i) &= \frac{TP_1}{(TP_1 + FP_1)} = \frac{1}{(1 + 0)} = 1, \text{when } TP_1 = 1 \\ R_m^1(S_i) &= \frac{TP_1}{(TP_1 + FN_1)} = \frac{1}{(1 + 0)} = 1, \text{when } TP_1 = 1 \end{aligned}$$

and

$$\begin{aligned} P_m^2(S_i) &= \frac{TP_2}{(TP_2 + FP_2)} = \frac{N - 1}{(N - 1 + 0)} = 1, \text{when } TP_1 = 1 \\ R_m^2(S_i) &= \frac{TP_2}{(TP_2 + FN_2)} = \frac{N - 1}{(N - 1 + 0)} = 1, \text{when } TP_1 = 1 \end{aligned}$$

4.1.2 The correct node has not been found

When a correct node has not been found, it means that p_t is wrongly classified as not relevant, which gives $TP_1 = 0$.

When p_t is wrongly classified as not relevant it must be an FN. Still all but one of the not relevant nodes are still correctly classified as not relevant. Similarly, another node must be wrongly classified as relevant which means that the target node must be a FP. This gives $TP_2 = N - 2$, $FP_2 = 1$ and $FN_2 = 1$

Using equations 3 and 4, if the selected node is not p_t , $TP_1 = 0$. This gives:

$$P_m^1(S_i) = \frac{TP_1}{(TP_1 + FP_1)} = 0, \text{ when } TP_1 = 0$$

$$R_m^1(S_i) = \frac{TP_1}{(TP_1 + FN_1)} = 0, \text{ when } TP_1 = 0$$

and:

$$P_m^2(S_i) = \frac{TP_2}{(TP_2 + FP_2)} = \frac{N - 2}{(N - 2 + 1)}, \text{ when } TP_1 = 0$$

$$R_m^2(S_i) = \frac{TP_2}{(TP_2 + FN_2)} = \frac{N - 2}{(N - 2 + 1)}, \text{ when } TP_1 = 0$$

4.1.3 $P_m(S_i)$ and $R_m(S_i)$

Following the logic in section 4.1.1 and 4.1.2 as well as equation 3 and 4, $P_m(S_i)$ and $R_m(S_i)$ must, if the algorithm correctly classifies p_t as relevant, be:

$$P_m(S_i) = \frac{1}{2} (P_m^1(s_i) + P_m^2(s_i))$$

$$= \frac{1}{2} (1 + 1) = 1, \text{ when } TP_1 = 1$$

$$R_m(S_i) = \frac{1}{2} (R_m^1(s_i) + R_m^2(s_i))$$

$$= \frac{1}{2} (1 + 1) = 1, \text{ when } TP_1 = 1$$

Following the same logic if the algorithm wrongly classifies p_t as not relevant. $P_m(S_i)$ and $R_m(S_i)$ must be:

$$P_m(S_i) = \frac{1}{2} (P_m^1(s_i) + P_m^2(s_i))$$

$$= \frac{1}{2} \left(0 + \frac{N - 2}{(N - 2 + 1)} \right)$$

$$= \frac{N - 2}{2N - 2}, \text{ when } TP_1 = 0$$

$$R_m(S_i) = \frac{1}{2} (R_m^1(s_i) + R_m^2(s_i))$$

$$= \frac{1}{2} \left(0 + \frac{N - 2}{(N - 2 + 1)} \right)$$

$$= P_m(S_i), \text{ when } TP_1 = 0$$

This shows, for EMRIS, $P_m(S_i)$ and $R_m(S_i)$ are always identical. Furthermore, $P_m(S_i)$ and $R_m(S_i)$ can be derived directly from only knowing if TP_1 exists and the number of number nodes in the directed graph, N .

Thus, equation 1 and 2 can for the EMRIS be simplified as following. If TP_1 , then $R_m(S_i) = TP$, otherwise $R_m(S_i) =$

$\frac{(|TP - 1|)(N - 2)}{2N - 2}$, or written as following:

$$R_m(S_i) = TP_m(S_i) = TP + \frac{(|TP - 1|)(N - 2)}{2N - 2} \quad (5)$$

4.1.4 Usefulness of $P_m(S_i)$ and $R_m(S_i)$ for EMRIS

Despite the common usage in classification problems and used for EMRIS in, [2] it is noteworthy that $R_m(S_i)$ is not a good metric for the EMRIS. This can be shown by defining an algorithm which deliberately wrongly classify p_t as not relevant. This is the worst algorithm possible as TP will always be 0. Even for such an algorithm $R_m(S_i)$ will correctly classify $N - 2$ nodes as non-relevant, which is highly rewarded $R_m(S_i)$.

Thus, even for the worst possible algorithm large directed graphs will receive a score of $\frac{1}{2}$ for $R_m(S_i)$.

$$R_m(S_i) \lim_{N \rightarrow \infty} = TP + \frac{(|TP - 1|)(N - 2)}{2N - 2}$$

$$= \frac{1}{2}, \text{ when } TP = 0. \quad (6)$$

Also for small directed graphs, e.g. with a directed graph of 102 nodes, $R_m(S_i) \approx \frac{1}{2}$ when $TP = 0$.

4.1.5 Accuracy

Another often used, but simpler, metric in the classification field is accuracy. It is defined as the percentage of TP over the total number of tests applied. Accuracy will for the EMRIS be defined as the average $TP(S_i)$:

$$Accuracy = \sum_{i=0}^{i=T} TP(S_i) / T \quad (7)$$

where T is the number of tests applied.

Note that this is for multiple directed graphs. In EMRIS, for one $TP(S_i) \in \{0, 1\}$, and, for all tests T , Accuracy $\in [0, 1]$.

Keep in mind that $P_m(S_i)$, $R_m(S_i)$, or averages of these over number of T , are considered more valuable metrics for classification problems as these also include FN , FP , and TN [24], but are not good metrics for EMRIS (see section 4.1.4).

4.2 Information Retrieval

[2] presented evaluation metrics retrieved from various sources for the EMRIS:

[2] defines effectiveness as $F_1(S_i) = \frac{2P_m(S_i)R_m(S_i)}{P_m(S_i) + R_m(S_i)}$, but failed to notice that $P_m(S_i)$ and $R_m(S_i)$ are identical for the EMRIS (see section 4.1.3) which gives the following:

$$F_1(S_i) = \frac{2P_m(S_i)R_m(S_i)}{P_m(S_i) + R_m(S_i)}$$

$$= \frac{2(R_m(S_i))^2}{2R_m(S_i)}$$

$$= R_m(S_i) = P_m(S_i) \quad (8)$$

Thus for EMRIS $F_1(S_i) = R_m(S_i) = P_m(S_i)$

Additionally, [2] defined efficiency as the path length (number of downloaded nodes) and duration time.

4.3 Focused crawling

There are two main metrics in focused crawling [25]:

Target Recall: The fraction of relevant pages on the web downloaded by the crawler.

Harvest Rate: The fraction of crawled pages over those that are relevant.

Keeping in mind that the number of relevant nodes in a directed graph is deliberately set to one for the EMRIS, the metrics can be simplified.

Let $D(S_i)$ be the number of downloaded nodes and $R_m(S_i)$ be the number of relevant nodes from directed graph S_i . This gives:

$$\text{Harvest Rate} = \frac{D(S_i)}{R(S_i)} = \frac{D(S_i)}{1} = D(S_i) \quad (9)$$

$$\text{Total Recall} = \frac{R(S_i)}{D(S_i)} = \frac{1}{D(S_i)} \quad (10)$$

Thus, from focused crawling, only the number of downloaded nodes ($D(S_i)$) is a relevant metric for the EMRIS.

4.4 Relevant metrics

As discussed in sections 4.1, 4.2 and 4.3, the relevant functions are $TP(S_i)$ (the number of true positives, accuracy) and $D(S_i)$ (the number of downloaded nodes).

Based on these metrics and the directed graph presented in section 2.1, we can formally define EMRIS as:

EMRIS: For directed graph S_i , maximize $TP(S_i)$ and minimize $D(S_i)$

5. EXPERIMENTS

This section presents the experimental set up and results. To give the best possible picture, two different types of experiments were carried out.

Generated directed graphs:. Each web site is modeled as a graph which are automatically generated. They are applied in the simulated environment and verified. This allows for many experiments with various setup.

Real web sites:. Selected algorithms are applied to real web sites solving actual problems. This shows to what extent the algorithms work in practice.

5.1 Generated directed graphs

This section presents the experimental setups and results using generated directed graphs. For each iteration of the experiment, a directed graph is generated as described in section 2.1. This graph represents a web site $S_i = \{\mathbf{P}_i, \mathbf{E}_i, \mathbf{L}_i\}$ where $\mathbf{P}_i = \{p_0 \dots p_N\}$ and N is the number of nodes in the site.

The directed graphs are generated as random controlled directed graphs using the evolving copy model [6] with the number of out-degrees (d-value) and in-degrees following an inverse power law (Zipf) [26, 27]. [6] showed with empirical evidence that the evolving copy model realistically represents the actual web.

It should be noted that part of the work in [6] shows how the web evolves over time. The EMRIS focuses on finding one node within a directed graph at a given time instance. How results may change over time is not within scope of this paper. Thus, the evolving part of the graphs is not relevant and is not implemented.

The content of the nodes are randomly generated using defined governmental keywords from the Norwegian LOS structure [28].

The copy model is extended with edge labels representing link texts. The edge labels are randomly chosen of words within the corresponding node text with a probability α and words not within the node with probability $1 - \alpha$. This means that some labels are describing the corresponding node while others are not. This resembles real web pages which contain links with texts intended to describe the corresponding page, [5] but are often times are not [29].

5.1.1 Experimental setup

All algorithms described in section 2 and 3 are implemented and tested. For each iteration, the algorithms are given a directed graphs. The starting node, p_s , is set to the start of the directed graph. The target node, p_t , is randomly picked from $\mathbf{P}_i$. The input query, q , is a random selection of words within p_t with probability α and a words within $\mathbf{P}_i$ but outside p_t with probability $1 - \alpha$. α is set to 0.75. N is chosen, in line with [2] as $N \in \{10^2, 10^3, 10^4\}$.¹

All results are average of 150 iterations.

5.1.2 Algorithms

Many algorithms exist in the field of information retrieval, focused crawling and classification. For practical reasons those that seemed most viable were selected in these experiments. However, other algorithms from these fields may yield different, and even better results. Nevertheless, the algorithms and set up were chosen to give as fair and objective results as possible, and provide a fair comparison with the LS.

¹[2] also included experiments for graph sizes of $N = 10$. Experiments for $N = 10$ were carried out for this paper, but have been deliberately been omitted. This was done to present only the most relevant results and because it is not useful to apply any EMRIS algorithms on a small graph of only 10 nodes since downloading all 10 nodes should be possible even with the limited capacity.

The algorithms have been implemented with as described in section 2 and 3. As many of these have not been applied to EMRIS previously, some assumption on the implementation were required. This section presents the implementation specific details. Additional algorithms are implemented to show upper and lower bounds.

Further note that the LS is used with the exact same algorithms and input parameters as HMM. Thus, the results can empirically show the impact in accuracy and number of downloaded nodes ($D(S_i)$) by introducing LS.

Random Walk (RW). The Random Walk (RW) is an algorithm which randomly crawls the directed graph. Despite its simplicity it is an often used crawling strategy which has many properties considered valuable when it comes to web crawling [30].

The algorithm works as following. For each node p_j the RW selects a random edge $e_{j,k}$ available from p_j . Subsequently, it follows $e_{j,k}$ to node p_k and continues selecting random edges.

Since it is unlikely that the RW reaches p_t the performance can be seen as a lower bound. To avoid infinite runs the maximum length of the RW is set to 20% of the population size, as was done in [2].

Optimal. The optimal algorithm finds the shortest path between p_s and p_t . In a controlled experiment there is perfect knowledge of the entire directed graph. The optimal algorithm knows every possible path. Using this knowledge, it measures which is the shortest path to reach p_t from p_s and crawls these nodes. It will always reach the best possible result with the fewest number of downloaded nodes and should be seen as an upper bound.

A*. Many pathfinding algorithms exists. The aim of pathfinding algorithms are to find the best, often the shortest, path between the starting node p_s and target node p_t . This is done by inputting the p_s and p_t . I.e., in contrast to the EMRIS, the algorithms always know when p_t has been reached.

There are however similarities between pathfinding and the EMRIS. Pathfinding algorithm use path and edge information in order to decide which link(s) to follow to get the most probable path to the p_t .

A common example of pathfinding algorithms is the A* algorithm. To find the most cost cost effective path from p_s to p_t a distance cost function is calculated consisting of: (1) A path-cost function per path and (2) a heuristic estimate to the goal.

Note that what is the most cost effective path according to A* is not by itself important for EMRIS. The number of nodes to download to reach the best and find the target page is important (see section 4).

To avoid giving the A* an unfair advantage, p_t is not used as input. Instead we let the heuristic estimate of the distance towards the target node be the similarity metric of its neighbors [2].

Focused crawler. The satisfactory factor in the content based focused crawler, which is used as part of the stop criteria in the crawler, is set to the number of relevant download nodes compared to the expected relevant nodes. Since the number of relevant nodes in EMRIS is set to only one, the satisfactory is set very small: $P(C) = 1/N$. Note that in the controlled experiment N is available. However, this cannot be assumed for real web sites where the number of pages in the web site in most cases are not known.

The content based focused crawler assumes that nodes are available in the directed graph before a crawl starts [22]. This is not available.

To make the algorithm work in practice, the following two assumptions were implemented: (1) Let the crawler download all nodes from the p_s as a starting point. (2) When new nodes and edged are discovered, let the crawler continuously update the knowledge of the directed graph.

HMM/HMM 2F. Traditional classification algorithms are not created to work in directed graphs and do not have functionality to prioritize which nodes to visit. Because of this, in our experiments we let the classification algorithms download every available node in the graph. This means that the classification algorithms will not perform well with respect to the number of downloaded nodes, but should have a fair chance when it comes to accuracy.

5.1.3 Threshold

The threshold in this experiments are set to $\{0.25, 0.75\}$. The algorithms OPTIMAL, RW and FOCUSED do not depend on thresholds. A*, SIM, DEG and SIMDEG originally do not depend on thresholds, but need a threshold to work in this experimental set up. Except for OPTIMAL, RW and FOCUSED, all algorithms are run with the same thresholds. The thresholds are set up with in an attempt to give fair results. In order to be brief, and because because the thresholds $\{0.25, 0.75\}$ were enough to communicate the findings, additional thresholds were omitted. However, other thresholds which have not been tested may yield different results.

5.1.4 Results from generated directed graphs

This section, including figure 4 and 5 shows only a selected subset of the results. Additional results are available in appendix A table A. The metrics used for the results are presented in section 4.

Figure 4 shows the accuracy of the selected algorithms with different number of nodes in the directed graph (N) in threshold=0.75. Figure 5 shows the number of downloaded nodes in the same experimental runs.

The results in figure 4 and 5, show that in some cases SIMDEG is better than DEG. This is the expected behavior [2]. However, in many cases SIMDEG is not better than SIM. This is in contrast to previous findings. This behavior may be because the number of edges from a node will not always be good indicators of where the target node is located. In fact, the results suggest that occasionally the number of the edges is misleading.

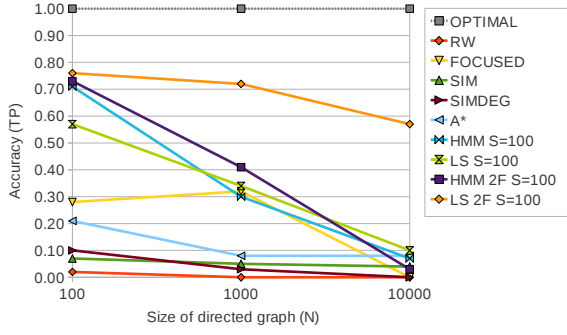


Figure 4: Accuracy of algorithms

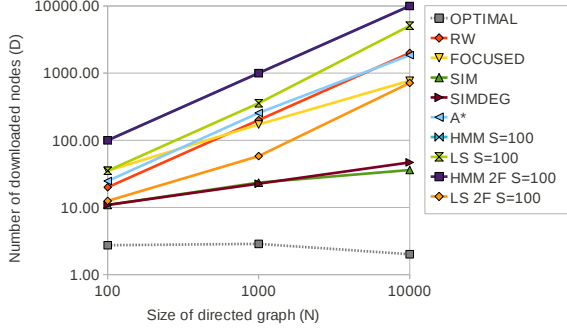


Figure 5: Number of downloaded nodes

The further results show that the LS2F with 100 states (LS2F S=100) outperforms all other algorithms (not considering optimal) when it comes to accuracy. The superiority is significantly more visible when $N = 10000$ compared to when $N \in \{100, 1000\}$. LS2F reaches an accuracy of 0.57. It even outperforms the HMM2F S=100 which reaches an accuracy of 0.03. Keep in mind that HMM2F and LS2F uses the the exact same classification algorithm, and that HMM2F downloads the entire directed graph. This suggests that the tested classifiers work better when less nodes are available to classify.

For the directed graph when $N = 10000$, the LS2F S=100 downloads in average on 715.1 nodes. Several other algorithms download significantly less nodes but are nowhere near when it comes to accuracy; SIM downloads 36.1 nodes and reach an accuracy of 0.04, while SIMDEG downloads 46.9 nodes reaching an accuracy of ~ 0.0 .

Thus the LS2F S=100 can reach an accuracy of 0.57 in a directed graph of 10000 by only downloading 715.1 nodes. In a directed graph of 1000 nodes in reaches the accuracy of 0.72 by only downloading 58.2 nodes.

Thus, the LS significantly outperforms existing algorithms, and it performs comparatively better as the size of the graph increases.

5.2 Results on real web site

To show how the algorithms work in practice, this section presents selected algorithms applied on real web site.

The test is finding the mail record web page on Norwegian municipality web sites. This is a page important for the transparency on local Norwegian government. The target page p_t , the page the algorithms should find, is therefor set to the web page presenting mail records. For the web sites which did not have mail records, no target page is available.

The algorithms used in this experiment are the LS2F with 100 states (LS2F S=100) and the Similarity Search (SIM), with the threshold for stopping set to 0.75. LS2F and SIM were chosen since it is the setups which yielded the best results in the simulated environment.

The query used is derived from the most common words, after removing stop words, in a random selection of 10 mail record pages. To ensure that no algorithm has an unfair advantage, the exact same query is used for both algorithms. Furthermore, all pages from which the query have been extracted, have deliberately been removed from the experiment.

The web site size is estimated to consist of 26000 pages. This estimations is based on a median value of number of results from Google per site.²

In total 420 municipality web sites where used for this experiment. Each result is verified manually and categorized as either correct or incorrect. The accuracy, A_c and number of downloaded pages, D are calculated as described in section 4.

Algorithm	26000p (real web sites)	
	A_c	D
	Threshold=0.75	
LS2F S=100	0.84	25.35
SIM	0.28	84.03

Table 2: Real results from evaluating 420 Norwegian municipal web sites. Detecting mail records in municipal web sites.

The experiments show that both LS2F S=100 and SIM work better when applied to real web sites than in the generated directed graphs. The difference between the experiments in live web sites and generated graphs may be because (1) The generated graphs made the experiment particularly hard problem, (2) The setup of detecting mail records in public municipal web sites was particularly easy. Or perhaps a combination of these. For example, the target page p_t in the generated graphs where randomly selected. However, we can expect that the target page in the live web sites, the page with mail record, is closer to the starting page than randomly generated. This would be an easier problem.

In the results from real web site, the LS2F yields a significantly better accuracy than SIM, as well as significantly fewer number of downloaded pages.

²Note that the number of pages per site is *not* gaussian distributed, and have a arithmetic mean of ~ 87000 . There are a few huge web sites, while most are around the median of 26000 pages.

6. CONCLUSION

This paper addresses finding a single node in a directed graph of nodes with directed edges and labels. It is a problem equivalent to finding a single page in a web site. This problem, referred to as the exact match on rare item searches (EMRIS), has not been addressed much in the literature. However, many very closely related problems have been addressed in the fields of information retrieval, web page classification and directed search.

This paper presents a formal description of the EMRIS, including a description of the environment from which the EMRIS interacts.

This paper also defines metrics valuable for evaluating the EMRIS. These metrics are based on findings in related fields. The analysis shows that the most viable metrics to use for EMRIS is the number of downloaded nodes and accuracy.

As a solution to the EMRIS, this paper presents a new innovative algorithm called the lost sheep. It is specifically designed to work in a directed graph consisting of edges, labels and nodes. The lost sheep can be seen as a pre-classifier on node labels (link texts) and decide if a node (web page) is candidate for further evaluation before downloading it.

State-of-the-art algorithms, both algorithms previously used for EMRIS field, and the research fields closely related, have been implemented and empirically tested both on generated graphs and real web sites.

The results show that the lost sheep significantly outperforms the state-of-the-art algorithms, both when it comes to accuracy and number of downloaded nodes. That it outperforms the other algorithms in the number of downloaded nodes is not surprising as the lost sheep is the only algorithm which utilises the labels which describe the corresponding nodes. However, the results also show that the lost sheep has a significant increased accuracy of the results. The performance of the lost sheep drastically increases as the number of nodes part of the directed graph increases.

7. FUTURE WORK

In some cases the link texts in web pages are not descriptive, however the text close to the link text may be valuable information [25]. An example is the text “find the contact information *here*” where *here* is the link text. This not considered in the directed graph model. By extending the edge labels with text close to the link may increase the accuracy.

The lost sheep does not addresses that nodes may be of different sizes. In the real web, as well as in the model presented in this paper, nodes may be of different sizes which could influence the results. The performance of the lost sheep may improve if the size of the nodes are considered.

This paper presents empirical results on the lost sheep algorithm for graph sizes 10^2 , 10^3 and 10^4 and real web sites. The authors plan to run experiments for more graph sizes. Additionally the authors plan to present analytically results on how the lost sheep works and the complexity of the algorithms.

Acknowledgements

The eGovMon project, <http://www.egovmon.no>, is co-funded by the Research Council of Norway under the VERDIKT program. Project no.: VERDIKT 183392/S10.

8. REFERENCES

- [1] Lundblad, N.: e-Exclusion and Bot Rights: Legal aspects of the robots exclusion standard for public agencies and other public sector bodies with Swedish examples. *First Monday* **12**(8-6) (2007)
- [2] Ke, W., Mostafa, J.: Scalability of findability: effective and efficient IR operations in large information networks. In: *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, ACM (2010) 74–81
- [3] Chakrabarti, S.: Data mining for hypertext: A tutorial survey. *ACM SIGKDD Explorations Newsletter* **1**(2) (2000) 1–11
- [4] Menczer, F.: Mapping the semantics of web text and links. *IEEE Internet Computing* **9**(3) (2005) 27–36
- [5] Kleinberg, J., Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A.: The web as a graph: Measurements, models, and methods. In: *Proceedings of the 5th annual international conference on Computing and combinatorics*, Springer-Verlag (1999) 1–17
- [6] Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A., Upfal, E.: Stochastic models for the web graph. In: *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, IEEE (2002) 57–65
- [7] Mehler, A., Dehmer, M., Gleim, R.: Towards logical hypertext structure. A graph-theoretic perspective. *Lecture notes in computer science* **3473** (2006) 136
- [8] Qi, X., Davison, B.D.: Web page classification: Features and algorithms. *ACM Comput. Surv.* **41**(2) (2009) 1–31
- [9] Ester, M., Kriegel, H., Schubert, M.: Web site mining: a new way to spot competitors, customers and suppliers in the world wide web. In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM (2002) 258
- [10] Oh, H., Myaeng, S., Lee, M.: A practical hypertext categorization method using links and incrementally available class information. In: *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, ACM (2000) 264–271
- [11] Fürnkranz, J.: Hyperlink ensembles: A case study in hypertext classification. *Information Fusion* **3**(4) (2002) 299–312
- [12] Sun, A., Lim, E., Ng, W.: Web classification using support vector machine. In: *Proceedings of the 4th international workshop on Web information and data management*, ACM (2002) 96–99
- [13] Tian, Y., Huang, T., Gao, W.: Two-phase Web site classification based on Hidden Markov Tree models. *Web Intelligence and Agent Systems* **2**(4) (2004) 249–264
- [14] Ulltveit-Moe, N., Olsen, M.G., Pillai, A., Thomsen, C., Gjørseter, T., Snapprud, M.: Architecture for large-scale automatic web accessibility evaluation

- based on the uwem methodology. In: Norwegian Conference for Informatics (NIK). (2008)
- [15] Sastry, P., Thathachar, M.: Learning automata algorithms for pattern classification. *Sadhana* **24**(4) (1999) 261–292
- [16] Pal, S., Pal, A.: Pattern recognition: from classical to modern approaches. World Scientific Pub Co Inc (2001)
- [17] Granmo, O., Oommen, B., Myrer, S., Olsen, M.: Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **37**(1) (2007) 166–175
- [18] Granmo, O., Oommen, B.: A Hierarchy of Twofold Resource Allocation Automata Supporting Optimal Sampling. *Next-Generation Applied Intelligence* (2009) 523–534
- [19] Chakrabarti, S., Van den Berg, M., Dom, B.: Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks* **31**(11-16) (1999) 1623–1640
- [20] Novak, B.: A survey of focused web crawling algorithms. *Proceedings of SIKDD* (2004) 55–58
- [21] Liu, H., Milios, E., Janssen, J.: Probabilistic models for focused web crawling. In: *Proceedings of the 6th annual ACM international workshop on Web information and data management, ACM* (2004) 16–22
- [22] Aggarwal, C., Al-Garawi, F., Yu, P.: Intelligent crawling on the World Wide Web with arbitrary predicates. In: *Proceedings of the 10th international conference on World Wide Web, ACM* (2001) 96–105
- [23] Culberson, J.: On the futility of blind search: An algorithmic view of “no free lunch”. *Evolutionary Computation* **6**(2) (1998) 109–127
- [24] Mason, J.E., Shepherd, M., Duffy, J.: An n-gram based approach to automatically identifying web page genre. *Hawaii International Conference on System Sciences* **0** (2009) 1–10
- [25] Pant, G., Srinivasan, P.: Learning to crawl: Comparing classification schemes. *ACM Trans. Inf. Syst.* **23**(4) (2005) 430–462
- [26] Bharat, K.: Patterns on the Web. In: *String processing and information retrieval: 10th international symposium, SPIRE 2003, Manaus, Brazil, October 8-10, 2003: proceedings, Springer-Verlag New York Inc* (2003) 1
- [27] Adamic, L., Huberman, B.: Zipf’s law and the Internet. *Glottometrics* **3**(1) (2002) 143–150
- [28] Norwegian Agency for Public Management and eGovernment (Difi): Los, ein informasjonsstruktur for offentlege tenester. Retrieved October 29th, 2010, from <http://los.difi.no/> (2010)
- [29] Olsen, M., Snaprud, M., Nietzio, A.: Automatic Checking of Alternative Texts on Web Pages. *Computers Helping People with Special Needs* (2010) 425–432
- [30] Najork, M., Heydon, A.: High-performance web crawling. *Handbook of massive data sets* (2002)

APPENDIX

A. RESULTS

Algorithm	100p		1000p		10000p	
	Ac.	D	Ac.	D	Ac.	D
	No threshold					
OPTIMAL	1.0	2.74	1.0	2.87	1.0	2.01
RW	0.02	20	0.00	200	0.00	2000
FOCUSED	0.28	35.14	0.32	171.9	0.00	771.7
	Threshold=0.25					
SIM	0.02	8.00	0.01	15.8	0.03	23.6
DEG	0.0	8.50	0.0	15.1	0.0	29.7
SIMDEG	0.01	7.38	0.0	14.1	0.0	24.5
A*	0.04	11.94	0.04	40.8	0.02	537.4
HMM S=3	0.80	100	0.41	1000	0.07	10000
HMM S=100	0.68	100	0.28	1000	0.02	10000
HMM S=200	0.58	100	0.22	1000	0.14	10000
HMM S=400	0.47	100	0.16	1000	0.09	10000
LS S=3	0.78	20.84	0.68	106.4	0.52	429.3
LS S=100	0.49	27.10	0.31	77.0	0.14	826.0
LS S=200	0.40	24.13	0.18	57.2	0.25	1670.7
LS S=400	0.14	14.67	0.18	146.1	0.13	1736.0
HMM2F S=100	0.75	100	0.49	1000	0.04	10000
HMM2F S=200	0.68	100	0.40	1000	0.06	10000
HMM2F S=400	0.75	100	0.47	1000	0.04	10000
LS2F S=100	0.79	11.77	0.70	48.4	0.47	195.6
LS2F S=200	0.78	12.25	0.65	50.8	0.67	368.8
LS2F S=400	0.73	13.94	0.67	52.6	0.51	187.0
	Threshold=0.75					
SIM	0.07	10.89	0.05	23.4	0.04	36.1
DEG	0.00	11.16	0.0	31.6	0.0	51.9
SIMDEG	0.10	10.91	0.03	22.6	0.0	46.9
A*	0.21	24.76	0.08	255.6	0.08	1853.4
HMM S=3	0.69	100	0.45	1000	0.09	10000
HMM S=100	0.71	100	0.30	1000	0.07	10000
HMM S=200	0.60	100	0.26	1000	0.06	10000
HMM S=400	0.47	100	0.20	1000	0.09	10000
LS S=3	0.75	22.52	0.67	124.0	0.32	1839.2
LS S=100	0.57	35.19	0.34	356.5	0.10	5088.0
LS S=200	0.37	35.64	0.25	519.4	0.17	5442.7
LS S=400	0.30	43.45	0.16	633.8	0.09	8278.1
HMM2F S=100	0.73	100	0.41	1000	0.03	10000
HMM2F S=200	0.75	100	0.45	1000	0.06	10000
HMM2F S=400	0.74	100	0.47	1000	0.04	10000
LS2F S=100	0.76	12.52	0.72	58.2	0.57	715.1
LS2F S=200	0.80	12.64	0.66	75.8	0.41	578.4
LS2F S=400	0.78	11.72	0.64	75.2	0.35	793.0

Table 3: The average accuracy ($Ac.$) and average Number of downloads (D) of algorithms for web sites with 100 , 1000 and 10000 pages (100p, 1000p and 10000p).