Propagation of Meta Data over the World Wide Web

Ola Ågren
Department of Computing Science
Umeå University
SE-901 87 Umeå, SWEDEN

Abstract

In this paper we propose a distribution and propagation algorithm for meta data. The main purpose of this is to tentatively allocate or derive meta data for nodes (in our case sites and/or web pages) for which no meta data exists.

We propose an algorithm that depends on 1) meta data given to a node, site and/or web page, 2) how pervasive we perceive this meta data, and 3) the trust that we give to this meta data. We will also show that PICS labels can be used to hold the meta data even for distant web pages and sites.

Keywords: meta data, automatic propagation, PICS, spatial linking

1 Introduction

Meta data embedded in web pages are one of the most reliable means of acquiring meta data for web pages. Web developers have however a tendency to omit meta data in their pages. Those that do add meta data tend to either put in very sparse meta data or everything that might occur further down in the web hierarchy that they control. In this work we propose a way to use the meta data already available in hierarchies of web pages to automatically propagate said information to other web pages.

Meta data that has been well thought out can be used to infer content domain (and in some cases sub-domain) for web pages. Some of the embedded meta data that we have found while looking at a large number of web pages can indeed be used as substitute for the content of the web page when doing searches.

We postulate that searching and browsing can be performed on the meta data level directly first. We have furthermore found that the connection between web pages and meta data can be viewed in the same way as PICS labels (i.e. meta data is either bound to a specific web page or to an entire (sub) hierarchy of the web).

1.1 PICS

W3C has devised a protocol called Platform for Internet Content Selection (PICS), with which organizations and companies can provide filters that allow only suitable pages to be presented. The PICS standard does not say how a rating shall be done, nor how it should be used or presented. That is completely up to the software providers and this has led to an abundance of different rating systems, none of which are compatible with the others. The good part with this is that a user of these systems can choose not only among suitable ratings but also which services to trust [10, 11, 5].

There are few providers of rating services that use PICS. All of them use it for client side blocking. That means that the browser first asks the service provider for its rating of a certain page, and if that rating is within predefined boundaries...
it starts to down-load the actual web page. Parents, teachers and/or employers can choose which rating service to trust and what levels of the ratings are suitable for down-loading. Providers of these types of services include NetNanny, Cyber-sitter, Cyber Patrol, and Surf Watch [14, 15, 18].

1.2 Web-Based Meta Data

PICS labels can be used to store any kind of textual meta-information about an Uniform Resource Identifier (URI). One way to use a PICS label bureau (an off-site provider of PICS labels) is as a meta data repository/warehouse. This data can then be searched much more effectively than searching through all the corresponding web pages [5, 1].

The PICS based labels are either for a specific web page (e.g. http://www.com/users.html) or for a web path and everything beneath it (e.g. http://www.com/). If we have a label bureau set up and a request comes for information pertaining to a specific web page/address (e.g. http://www.com/research/users.html) the bureau must first try to locate information using the exact web address given, and if that fails try with each path that is a prefix of the web address (e.g. first for the http://www.com/-research/ path and then http://www.com/) until the required data is found [10] (see Figure 1).

2 Propagation Algorithm

Looking at the web as a collection of nodes (web pages) and edges (links) yields the graph corresponding to the part of the WWW that we want to model. The nodes in the graph are further augmented with meta data for the corresponding web page. This graph can be used to generate meta data for web pages lacking actual meta data using the algorithms given later in this paper.

Meta data propagation is controlled by the trust level and pervasiveness that we attribute to a each node in the graph. Pervasive meta data might propagate further than one step. Propagation will stop when reaching nodes with higher trust levels.

We propose the following rules for meta data extraction and propagation:

1. If we have meta data extracted from a web page we would retain this data for the corresponding node (possibly reformatted and altered).

2. If meta data exists for a path that is a prefix of the requested web page then it is used for the node instead.

3. Depending on how much we trust the meta data from a certain node we have different ways of propagating the meta data given: The higher trust we put in the meta data of a node, the higher the probability that it will be propagated to those nodes that there are edges to. We have found that the intersection of all meta data represented in those web pages that have a edge to a node (looking at only those that have the highest trust level among the incoming links) will yield a system that is consistent and well behaved.

The different trust levels are not defined here. We view trust levels as an enumerated value ranging from "no trust" to "full trust". Conversion to and from these values must be defined in the domain that they are to be used.
depending on the requirements of the surrounding systems.

4. Moreover, some sites may be marked as pervasive, meaning that meta data from that site will propagate recursively to nodes further than one link away. It is held in check by pages already marked in the steps 1 or 2 of this algorithm, and will not propagate to pages marked in 3 unless the new meta data has a higher trust than already given.

We do **not** advocate setting all meta data systems to pervasive and fully trusted. Pervasive propagation should only be used when the propagation can be controlled in some other way. Uncontrolled propagation would yield an increasing amount of data to propagate to all nodes in the graph, diminishing the value and truthfulness of the meta data. The resulting graph from using pervasive classification of all instances of given meta data will yield a graph where all nodes will be marked with (possible diluted and/or wrong) meta data. The main reason for this is that everything that one can find a link chain to without data given by rules 1 or 2 will be set to the meta data of the source.

Given the small example in figure 2 we can show the differences between trust levels, i.e. meta data propagation using rules 3 and 4. If B is marked as a low trust system we can find the following propagations:

- D will be marked with x.
- If A is more trusted than C then E would be marked with x.
- If C is more trusted than A then E would be marked with y.
- If A and C are equally trusted then E would be marked with the intersection of x and y.
- If A is marked as pervasive and has at least the same trust as B then F would be marked with x, since the value given to D would continue to propagate.

It would not propagate if B is more trusted than A, since D would get the meta data from B rather than A. Just because A and B have the same meta data is not sufficient reason for further propagation of the meta data to F. \(^1\)

![Figure 2. Sample graph with nodes (A...F) marked with meta data (x...y).](image)

### 3 Definitions

The following definitions are the ones typically found in works about compilers, e.g. [7, 19, 3], but adapted to the needs of this paper.

**Definition 1** A semilattice is a set \( L \) with a binary **meet** operation \( \wedge \) such that for all \( a, b, c \in L \):

1. \( a \wedge a = a \) (idempotent)
2. \( a \wedge b = b \wedge a \) (commutative)
3. \( a \wedge (b \wedge c) = (a \wedge b) \wedge c \) (associative)

\(^1\)This might seem counterintuitive (since both A and B agree on the evidence) but our algorithm uses only the highest level of trust.
**Definition 2** A semilattice has a zero element 0 iff $a \land 0 = 0$ for every $a \in L$. $L$ has a one element 1 iff $a \land 1 = a$ for every $a \in L$.

**Corollary 1** If 0 exists, then it is unique. This holds true for 1 as well.

**Definition 3** If $(L, \land)$ is a semilattice and $a$ and $b$ are arbitrary elements in $L$ then we can define a relation $\leq$ in $L$:

$$a \leq b \iff a \land b = a$$

The $<$, $>$ and $\geq$ relations can be defined in a similar way.

**Corollary 2** Let $(L, \land)$ denote a semilattice and $\leq$ the relation introduced in Definition 3. Then $\leq$ is a partial order on $L$.

**Definition 4** A chain is a sequence $a_1, a_2, \ldots$ of elements from a semilattice $L$ iff $a_i \geq a_{i+1}$ for all $i = 1, 2, \ldots$

**Definition 5** A semilattice $L$ is bounded iff for every $a \in L$ there exists a $c_a \in \mathbb{N}$ such that the length of every chain beginning with $a$ is at most $c_a$.

**Definition 6** A total function $f : L \to L$ is monotonic iff for all $a, b \in L$:

$$f(a \land b) \leq f(a) \land f(b)$$

**Definition 7** A monotone data flow system (MDS) is a tuple $\Omega = (L, \land, F, G, FM)$, where:

1. $(L, \land)$ is a bounded semilattice with 0 and 1.
2. $F$ is a monotonic function space for $L$.
3. $G = (N, E)$ is a directed graph modeling the web, with web pages as nodes and the links between the web pages as edges. This would normally also contain a start node $s$, but since the WWW is not a totally connected graph we will ignore start nodes and ordering between nodes in the system.
4. $FM : N \to F$ is a total function over $N$.

**Section 4** A Monotone Data Flow System on Meta Data over Web Pages

We can now look at the web as a directed graph $G = (N, E)$, with web pages and paths as the nodes $N$ and the links represented by the edges $E$. There are a few basic rules that must apply in order to get a functioning and stable system:

1. Nodes that contain meta data that is valid for a path (including an entire web server) and everything beneath it in the site tree must be seen as highly trusted systems and have links to at least the web pages beneath it in the tree.

2. The meta data directly attributed to a specific node $n$ in the system must be marked as such, and will not be changed later on by the algorithm.

Furthermore, all meta data must be marked with the trust given to it and how pervasive it is. The data to be distributed over the graph is the meta data given to the system at start-up. In our system we have:

$$(L, \land) = (\mathbb{P}($meta data$), \cap),$$

$0 = \emptyset$ and

$1 = $ meta data.

The definitions in section 3 can then be used to model our web of meta data and web pages using algorithm 1 and 2 (a heavily rewritten general iterative algorithm [7]).

The result of the algorithms are found in trust (only used as an intermediate result between the two algorithms) and INF. INF is meant to supersede the given value of $FM$ in the final MDS.
ALGORITHM 1 (HANDLES ALL NON-PERVASIVE DATA)

Input: An MDS $\Omega = (L, \vee, F, G, FM)$ with $G = (N, E)$

1. $INF(n)$ : The actual meta data associated with node $n$. Given by rules 1 or 2, otherwise undefined.
2. $stabled(n)$ : Array of booleans marking that this node got its value by rules 1 or 2
3. $trust(n)$ : The trust level of the meta data given to the current node, enumeration or other values
4. $pervasive(n)$ : The data of this node is pervasive

Output: $INF(n)$ : See above, but updated by the algorithm

$trust(n)$ : See above, but updated by the algorithm

Variables: $n \in N$ : The node that we are currently looking at

1. $new(t)$ : Possible meta data for current node, per trust level
2. $N' \in \mathbb{P}N$ : Stable but not pervasive nodes
3. $t, hi$ : Temporary variables of trust levels

begin

Initialize

forall $n \in N \bullet undefined (INF(n))$ do

1. $INF(n) \leftarrow 0$;
2. $trust(n) \leftarrow no trust$;
end do

1. $N' \leftarrow \{ n \in N \mid stabled(n) \land \neg pervasive(n) \}$;
Handle all non-pervasive meta data once

forall $n \in N \bullet \neg stabled(n)$ do

Initialize required data

forall $t \in trust levels$ do

1. $new(t) \leftarrow 0$;
end do

1. $hi \leftarrow no trust$;
Find intersection of incoming links with highest trust

forall $n' \in N' \bullet (n', n) \in E$ do

1. $t \leftarrow trust(n')$;
2. $new(t) \leftarrow new(t) \cap INF(n')$;
3. if $t > hi$ then $hi \leftarrow t$; fi
end do

Update INF if a change has been found

if $t > no trust$ then

1. $INF(n) \leftarrow new(t)$; $trust(n) \leftarrow t$;
fi
end do
end
Algorithm 2 (Modified Iterative Algorithm)

Input: An MDS $\Omega = (L, \wedge, F, G, FM)$ with $G = (N, E)$
- $INF(n)$
- $trust(n)$: As given by algorithm 1
- $stabled(n)$
- $pervasive(n)$: As in algorithm 1

Output: $INF : N \rightarrow L$, A total function

Variables: $n$, $t$, $hi$, $new(t)$: As in algorithm 1
- $N' \subset P N$: Pervasive nodes
- $stable \in Boolean$: Have we reached a stable state?

begin

Repeat data propagation until stable

$stable \leftarrow false$;

while $\neg stable$ do

$stable \leftarrow true$;

$N' \leftarrow \{n \in N \mid pervasive(n)\}$;

Check for incoming meta data

foreach $n \in N \bullet \neg stabled(n)$ do

Initialize required data

foreach $t \in trust levels$ do

$new(t) \leftarrow 0$;

$od$

$hi \leftarrow no trust$;

foreach $n' \in N' \bullet (n', n) \in E$ do

$t \leftarrow trust(n')$;

$new(t) \leftarrow new(t) \cap INF(n')$;

if $t > hi$ then

$hi \leftarrow t$ fi

$od$

foreach $n'$

if $hi = trust(n)$

then $new(hi) \leftarrow new(hi) \cap INF(n)$;

fi

if $hi \geq trust(n)$

then

$INF(n) \leftarrow new(hi)$;

$trust(n) \leftarrow hi$;

$pervasive(n) \leftarrow true$;

$stable \leftarrow false$;

fi

$od$

foreach $n$ while $\neg stable$

end

5 Related Work

This work builds on all previous forms of web mining [9] of semi-structured (i.e. HTML) data. Typical examples of this includes wrapper induction like STALKER [12], and information extraction like RSV [6] or WHISK/CRYSTAL [16, 17].

Extracting the information from the web was however only the first step; we are more interested in how meta data can be viewed outside of the web. The extracted data can either be seen as a data base over the web [13] or as a source for web structure mining such as HITS [8], Clever [4], PageRank and Google [2].

6 Discussion

We have used the algorithms described in this paper to model the web structure (and meta data content) of Umeå University. The university is a medium sized university in northern Sweden with approximately 25,300 undergraduate and 1,300 graduate students. Its web structure contains less than 100 official web servers with a total of more than 200,000 static web pages (counting only HTML pages, not pictures and other binary data).

Very reliable results from this data set has been obtained when none of the meta data has been marked pervasive. Trust levels were set, in decreasing order, according to 1) individual web pages containing meta data, 2) meta data for a sub-tree in a hierarchy, and 3) for the corresponding server.

We have checked the validity of the given meta data. Most (approximately 95%) of the checked individually marked web pages had correct meta data set. Almost all sub-trees had correct meta data (less than 1% contained errors) and the meta data given on the server level were 100% correct.

Looking at this data set we find that $\approx 3\%$ of the pages contain embedded meta data keywords. Applying rule 1 in Section 2 makes this value jump to $\approx 55\%$ and rule 2 increases this even further to $\approx 67\%$. Setting some of the nodes/web
pages pervasive might yield an even higher percentage, depending on which nodes are marked pervasive.

References