World Wide Web (2011) 14:623–649
DOI 10.1007/s11280-011-0124-6

Indexing and querying segmented web pages: the BlockWeb Model

Emmanuel Bruno · Nicolas Faessel · Hervé Glotin · Jacques Le Maitre · Michel Scholl

Received: 22 April 2010 / Revised: 4 February 2011 /
Accepted: 9 February 2011 / Published online: 8 March 2011

Abstract We present in this paper a model for indexing and querying web pages, based on the hierarchical decomposition of pages into blocks. Splitting up a page into blocks has several advantages in terms of page design, indexing and querying such as (i) blocks of a page most similar to a query may be returned instead of the page as a whole (ii) the *importance* of a block can be taken into account, as well as (iii) the *permeability* of the blocks to neighbor blocks: a block b is said to be permeable to a block b' in the same page if b' content (text, image, etc.) can be (partially) inherited by b upon indexing. An engine implementing this model is described including: the transformation of web pages into blocks hierarchies, the definition of a dedicated language to express indexing rules and the storage of indexed blocks into an XML repository. The model is assessed on a dataset of electronic news, and a dataset drawn from web pages of the ImagEval campaign where it improves by 16% the mean average precision of the baseline.

E. Bruno (✉) · H. Glotin · J. Le Maitre
LSIS, Université du Sud Toulon-Var, BP 20132, 83957, La Garde Cedex, France
e-mail: bruno@univ-tln.fr

H. Glotin
e-mail: glotin@univ-tln.fr

J. Le Maitre
e-mail: lemaitre@univ-tln.fr

N. Faessel
LSIS, Université Paul Cézanne, Avenue Escadrille Normandie-Niemen,
13397, Marseille Cedex 20, France
e-mail: nicolas.faessel@lsis.org

M. Scholl
Cedric/Wisdom, CNAM, 292 Rue St Martin, 75141, Paris Cedex 03, France
e-mail: michel.scholl@cnam.fr

Keywords web page segmentation · block importance · block permeability · web image indexing · document indexing · document retrieval

Mathematics Subject Classifications (2010) H.3.1 · H.3.3

1 Introduction

We present in this paper a model for indexing and querying web pages according to their content as well as to their visual rendering. A web page can be viewed as a set of blocks containing multimedia information. The visual presentation of a block (font size, background color, ...) and its location in the page give information about its *importance*. Moreover, a block content is *permeable* to other blocks: it can receive from a neighbor block in the page or an embedded block, part or the totality of its content (e.g. the text surrounding an image can be used to index this image). Another advantage of splitting up a web page into blocks is that answers to queries can be precisely localized: blocks containing the keywords of a query can be returned instead of the page taken as a whole. One objective of this paper is to show that taking into account the organization of the page layout, in addition to its textual content can significantly improve the precision of web page querying. For example, Figure 1 sketches the home page of an electronic news paper. This page (block 1)

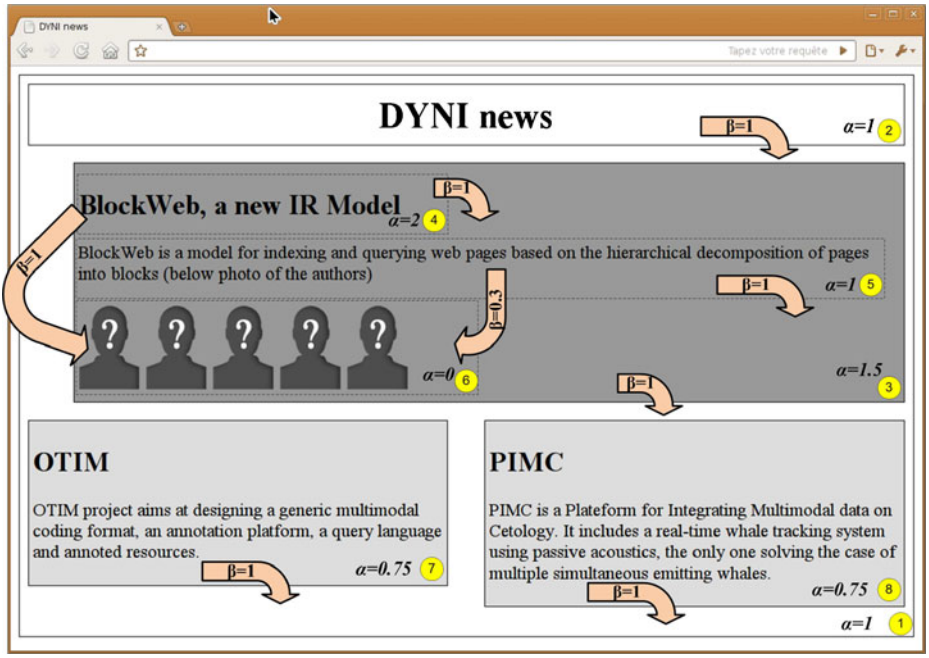


Figure 1 A segmented web page into eight blocks (the block number is in the yellow disc).

is decomposed into a title block (2) and three article blocks (3, 7, and 8). Block 3 is itself decomposed into three blocks: a title block (4), a body block (5) and an image block (6).

Segmentation of a web page into a hierarchy of blocks comes from the DOM tree of this page and from visual information extracted by using the rendering engine of a web browser (e.g., the Mozilla Gecko engine). In this paper we assume blocks do not overlap and that the concatenation of the leaves of the hierarchy represents the content of the page. The major contributions of this paper are (i) to take into account the hierarchical decomposition of a page into blocks and (ii) to model a page as a directed acyclic graph where each node represents a block, and each edge label accounts for the *permeability* of the source node to the target node content. To our knowledge, the latter feature is novel. We choose as a block querying paradigm keyword indexing and querying according to the vector space model [11]: a block content (text) is modeled by a vector in the N-dimensional space where N is the size of the vocabulary of terms. Without loss of generality, we decide in this paper that the only blocks whose content is not empty are the leaves of the hierarchical decomposition. The content of other blocks (in particular those blocks containing the leaves) are initially empty. Thanks to permeability, the content of the (non empty) leaves is propagated to other blocks and eventually all blocks reachable by permeability from a non empty block are modeled by a non null vector and therefore can be indexed and queried. Querying returns blocks ranked according to the classical cosine similarity measure. As other contributions, (i) we extend the notion of block importance which does not depend on the content but on the block visual rendering and location and we take it into account for block indexing, (ii) we exhibit an XML jargon called "XML Indexing Management Language" (XIML) for specifying the block permeability and importance and develop a complete query engine. To assess the model we apply it to electronic newspapers and to a corpus of web pages taken from the ImagEval campaign.

The paper is organized as follows: in Section 2 related work is given. The data model is presented in Section 3 which starts with an introduction to the major features of our block model illustrated on the web page of Figure 1. Section 4 presents the engine developed for block extraction, indexing and querying. A set of experiments is reported in Section 5 which illustrates the effectiveness of our decomposition into blocks in terms of precise localization of the answer and illustrates the impact of permeability on web page image indexing. Some concluding remarks are given in Section 6.

2 Related work

Related work concerns web page segmentation, evaluation of block importance, index propagation and image retrieval by surrounding texts. A large number of studies have been reported in these areas, out of which we only cite the pionnering ones or those which directly influence our study.

Page segmentation First approaches for block segmentation used manually defined wrappers while earlier approaches used HTML analysis. VIPS (VIsion-based Page Segmentation) [4] is an efficient and popular top-down algorithm to detect the

structure of a web page based on its DOM tree and visual representation rendered by a web browser. A simpler version of VIPS using the recursive X-Y cut algorithm for on-line medical journal papers analysis is proposed in [17]. The recursive X-Y cut algorithm was initially elaborated in the framework of a system for technical journal analysis [8]. An automated information extraction system is presented in [15] which takes advantage of the Web pages information regularities to organize their content into a hierarchical XML-like structure. Like VIPS the page segmentation algorithm relies on the DOM tree representation of the HTML page. The quality of the information extraction is improved by using a statistical domain model. When the visual schema of pages is regular and known, algorithms like VIPS are not really necessary because segmentation can be done by transforming the HTML source of the page on the basis of this schema, using classical XML manipulation languages. It is our current choice.

Block importance evaluation Several methods have been proposed to compute the importance of a block in a web page. The simplest [6, 9, 16] consist in classifying blocks into informative ones and noisy ones. In [9] this classification is done on the entropy of keywords in block textual content, and in [6, 16] it is based on block features (text, presentation tags, links, etc.) and block frequency in the corpus (by analogy with idf). The method presented in [12] is more ambitious. Blocks are classified into four levels of importance: (i) non informative, (ii) useful but not really linked to the main subject of the page, (iii) pertinent with this subject but of relative importance, (iv) at the heart of the page. This importance is deduced from spatial and content features of blocks. Classification is done using two learning methods: regression by neural networks for continuous real importance labels and classification by support vector machines for discrete importance labels. It is this kind of approach that can be implemented to learn block importance in our model. In the current state of our tool, importance is manually instantiated.

Index propagation A lot of studies have been done in this field in the framework of the Initiative for the Evaluation of XML Retrieval (INEX[1]) in particular in the ad hoc and multimedia tracks. In these works, it is the logical structure of the documents (title, sections, paragraphs, etc.) which is taken into account whereas in our approach it is the visual structure. Two important issues of information retrieval in structured documents are indexing propagation between document elements and providing best entry points. A very interesting solution is proposed in [5] for hierarchically structured documents: indexing terms equally distributed on every child of an element are pruned from this element and propagated to their parent element. The distribution of a term is measured by its entropy which depends on its frequency in children and parent elements. This process allows retrieving the most specific elements pertinent to a query but also decreasing the number of terms indexing each block. We plan to adapt this idea to improve our propagation algorithm.

Web image retrieval using their surrounding text Numerous propositions have been made for web image retrieval using the surrounding text of each image. The authors

[1] http://www.inex.otago.ac.nz/data/publications.asp

of [3] described an interesting web image classification method which operates as follows. Each page is segmented into blocks using the VIPS algorithm and an image graph is built. Each block contains one or several images and the text surrounding them. The weights defined on the edges of the image graph reflect semantic relationships between images. Three representations of images are then derived: visual features based representation, textual features based representation, and links graph based representation which are combined to cluster images into semantic categories. This work is somewhat close to our proposal with regard to image indexing. It shows that extracting the surrounding text can be done more precisely when the page has been previously segmented into blocks. We propose to go further in introducing the permeability concept which allows to quantify the contribution of each textual neighbor block of an image. On the other hand we do not use image features analysis as developed in [14] for ImagEval image web corpus, but it is one of our perspectives to use it.

3 The data model

Our model was first introduced in [1, 2]. With respect to these former versions, the current article not only presents in depth the model but characterizes precisely the impact of importance and permeability on block indexing as well. Moreover this paper describes the implementation of the engine and gives a refined evaluation of the impact of block decomposition and permeability on finding the best entry point related to a query and on the learning of the permeability parameters for image indexing.

Our starting point is a set of web pages. In this paper we are not concerned by the links between pages and leave as a future work the possible enrichment of the data model with hyper-links. We first informally present the basic concepts of the model illustrating its applicability scope through indexing and querying which are the focus of the rest of the paper. Then we give a formal presentation of the data model based on (i) a decomposition of a page into a hierarchy of blocks, (ii) the construction of a graph of blocks taking into account their importance and their permeability to other blocks content, and (iii) the use of the IR vector space model [11] for indexing and querying blocks.

3.1 Informal model

A large number of applications such as electronic news, use a *regular* decomposition of pages into a hierarchy of blocks. This decomposition is not only regular but also relatively stable with time and with different page designers in the same application. Then a better modeling of blocks might have some impact not only on the quality of answers to keyword querying, but also on the quality of the overall design of pages.

Relative *importance* of blocks is extracted from visual properties of the page layout. For example, one among the sequence of news blocks, called a “scoop” has more importance (larger size, displayed on the upper left part of the screen, larger text font, etc.). In Figure 1, article 3 may be seen as more important than articles 7 and 8 because it has a larger size than articles 7 and 8 which have the same importance because they have the same size.

Importance of a block b, denoted $\alpha(b)$, is chosen as relative to the importance of its twin blocks in the decomposition of a (parent) block into a set of blocks. In Figure 1, each block is labeled with its importance: block 2 has importance 1, block 3 has importance 1.5 and blocks 7 and 8 have importance 0.75 (assuming that the importance of an article is proportional to its size); block 4 has importance 2 and block 5 has importance 1 (taking into account the text font size and assuming that image blocks have null importance). Block importance may have several interpretations such as "delete (select) blocks whose importance is under (above) a given threshold", or "retrieve the blocks containing the word XML whose importance is larger than 1".

Another rationale behind the page decomposition in fine grained units is efficient indexing and querying. When searching for pages including one or several keywords, the user expects as a result a set of ranked pages containing the query keywords. The larger the page is, the harder information localization when browsing through the page is. By customizing the vector model to blocks, one can index and search blocks instead of pages tending to rank the block where the information is located before the block/page that contains this block. As an example (Figure 1) if the query is "`web`", one expects the system to find first block 5, then with lower ranks block 3 and block 1 (the page block).

Our following observation is the influence of spatial proximity of two blocks. A newspaper reader attracted by a scoop is tempted to read next, one of the neighbor blocks. This is why we introduce the concept of *permeability*: there is some permeability between the content of two blocks close to each other in the page layout. Taking into account *permeability* when weighting (for indexing purposes) two neighbor blocks allows for a richer indexing and querying of blocks: query "`OTIM PIMC`" should return first the page block (block 1) acknowledging that by permeability the parent block (we assume it has no initial content) inherits the keywords of its children blocks. Image annotation is another application of permeability. An image (with no annotation) could be reasonably indexed by the keywords of the neighbor block: the author photo (Figure 1) has for keywords that of its neighbor blocks (BlockWeb, IR, web ...). Permeability of a block b to the content of a block b' is denoted $\beta(b', b)$, a real ranging from 0 to 1. In Figure 1 arrows between blocks represent the non zero permeabilities between blocks. We have assumed that the image of an article is fully permeable to the article title ($\beta(4, 6) = 1$) and partially permeable to the article body ($\beta(5, 6) = 0.3$). A 1 permeability can be interpreted as "a keyword of the source block is a keyword of the target block". Two blocks far away in the page layout should not be permeable to each other: their mutual permeability is equal to zero. Note that if the initial content of a block is non empty (it already has keywords) by permeability, eventually, it is indexed on its initial keywords as well as on the keywords of blocks reaching it by permeability.

Last we observe that the page *hierarchy* of blocks corresponds to the HTML structure of the page where some subtrees have been aggregated into a single leaf. We might be interested only in the leaf blocks of this hierarchy, we call it a *view*[2]

[2]The view represents the specification of a partition of the page into blocks.

or in the parent blocks as well. In Figure 1, leaf blocks are blocks 2, 4, 5, 6, 7, and 8 which constitute a view. However, another view could be obtained after aggregation of the articles content (title, body, and images): the gray blocks. When parent blocks initially have no keyword, thanks to permeability, they recursively inherit the keywords of the leaf blocks. This is our choice in the following for querying blocks. The restriction to the view (zero permeability between a block and its parent) would imply parent blocks are not indexed. This might be useful for some applications. There might be permeability between blocks which are not in an ancestor relationship as well: it is the case for block 6 in Figure 1 which inherits keywords from blocks 4 and 5.

3.2 Model

3.2.1 Blocks

We choose a tree representation of page information, appropriate for modeling text information decomposition into sequences of more elementary texts. In other words, the information contained in a block b is representable as an inclusion hierarchy of more elementary blocks $b_1, b_2, \ldots, b_n$. We say that b includes or contains b_i, $i = 1, \ldots, n$. Pages are the largest blocks: they are not included into any block of the database. The tree representation of a non leaf block b is denoted $T(b)$. Note that two pages may share the same block. Because of lack of space, we present a simplified model in which two pages may *not* share the same block. Therefore, database *DB* is a forest of trees with roots, the pages of the database. Block information has one or two among the following types: text and image (short cut for a variety of media: graphics, images, videos, sound). While texts can be further decomposed into sub-texts, an image is atomic: a block representing an image cannot be further decomposed. The representation of a page includes some information about the page layout. Tree $T(b)$ induces a tree $TL(b)$ of minimum bounding rectangles (*mbb*) homomorphic to $T(b)$. Let b' be a block in $T(b)$. Either b' comes in the representation with a minimum bounding rectangle ($mbb(b')$) as an attribute of its layout, or $mbb(b')$ is computed as the *mbb* of its children.

$T(b)$ is said to be valid if for any two blocks b_1 and b_2 in $T(b)$ (i) if b_1 (b_2) is an ancestor of b_2 (b_1) in $T(b)$ then $mbb(b_1)$ includes $mbb(b_2)$ ($mbb(b_2)$ includes $mbb(b_1)$), else (ii) (b_1 and b_2 are not in an ancestor relation) $mbb(b_1) \cap mbb(b_2) = \emptyset$. A page *view* $V(p)$ is the set of leaves of $T(p)$. Since there are several decompositions of a page into a tree, there are several partitions of a page into a view. As an example a page might be viewed as its sequence of articles. Next a reader focusing on the scoop replaces in the former view the block "Scoop" by its 3 sub-blocks "Title", "Text" and "Image". This leads to the following definition of a page view:

1. a page is a view,
2. let $\{b_1, b_2, \ldots, b_n\}$ be a view and let $\{b_{i1}, \ldots, b_{ij}, \ldots, b_{ip}\}$ be a partition of b_i, then $\{b_1, \ldots, b_{i-1}, b_{i1}, \ldots, b_{ij}, \ldots, b_{ip}, b_{i+1}, \ldots, b_n\}$ is a view.

3.2.2 Identifier and content of a block

A block has an *identifier*[3] and a *content*. Let b denote the identifier of a block. Let V be the vocabulary of terms included in the database[4]. Then the content of block b, denoted by $I(b)$ is defined as:

1. if b is a leaf and is of type text, then $I(b) = t_b$ where t_b is the bag of terms of V in b. We say that b is *indexed by* t_b,
2. if b is a leaf and is of type image, then $I(b) = i_b$ where either b comes with an initial bag i_b of terms in V indexing b (the image has been indexed prior to the inclusion of the block into the database), or $I(b) = \emptyset$,
3. if b is a non atomic leaf (text + image), then $I(b) = t_b \cup i_b$ where t_b is the bag of terms of V in b and i_b is the bag of terms indexing image i,
4. if b is not a leaf, then its content is the empty bag: $I(b) = \emptyset$[5].

3.2.3 Importance of a block

We choose to base block importance on visual cues, independently of its content. We define importance as a function $\alpha : Bl \to \mathbb{R}^+$ where Bl is the set of the blocks of the database. If b is a block in Bl then $\alpha(b)$ denotes the importance of b. A variety of importance models may be defined among which we exhibit two.

1. **block context.** The importance of a block is relative to the importance of its siblings blocks. Let $b_1, \ldots, b_n$ be the set of blocks with parent f then the block importances are real satisfying $\sum_{i=1}^{n} \alpha(b_i) = n$.
2. **view context.** In contrast to the previous model which restricts the scope for a block importance to its siblings, in the view context model, the importance of a block is relative to all blocks in the view. It is undefined for other (internal blocks of the hierarchy). Let α_{bl} denote the importance of a block in the former "block context" model. Then in the view model of importance, let b be a block in the view whose identifier is a path through blocks $b'_1, b'_2, \ldots, b$. Its importance is $\alpha(b) = \alpha_{bl}(b'_1) \times \alpha_{bl}(b'_2) \times \cdots \times \alpha_{bl}(b)$. The block importances of the view satisfy:$\sum_{i=1}^{m} \alpha(b_i) = m$ where m is the number of blocks in the view.

In the following we choose the block context model for importance.

3.2.4 Permeability

A block is permeable to another block if the former inherits some part of the content of the latter. We define permeability as a relation which is anti-reflexive,

[3] As an identifier, one can choose an *XPath* expression whose prefix is an *url* (*uri*) allowing to reach the HTML representation of the page, and the suffix if any allowing to identify the root of the block inside the page.

[4] V is the set of distinct words obtained after preprocessing all texts in the database using usual linguistic tools (elimination of stop-words, stemming, etc.).

[5] Note that with such a definition, only the leaf nodes have a non empty content. An alternate definition for non leaf node content, that might be useful for other applications, would be "the (bag) union of its child node contents".

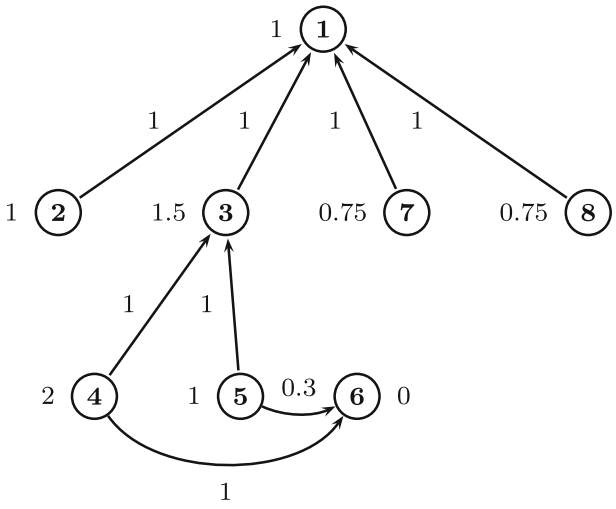


Figure 2 The *IP graph* of the page of Figure 1.

anti-symmetric and transitive. In order to quantify the part of the content of a block inherited by another block we introduce a function $\beta : Bl \times Bl \rightarrow [0, 1]$ (where Bl is the set of the blocks of the database). If b and b' are two blocks $\beta(b, b') = p$ denotes "the part p" of the content of b inherited by b'. If $p = 0$ then we say that b' is impermeable to b, and if $p = 1$, we say that b' is totally permeable to b. Since we do not take into account hyper-link, $\beta(b, b') = 0$ if blocks b and b' are not in the same page.

3.2.5 IP graph

Let $IPG(p)$ denote the *Importance-Permeability graph (IP graph)* of page p defined as follows. The nodes of $IPG(p)$ are the nodes of $T(p)$ *i.e.* the nodes associated with the blocks of page p. Each node is labeled by the importance of the block it is associated with. There is an edge from node b to node b' with label $\beta(b, b')$, $0 < \beta(b, b') \leq 1$, if b' is permeable to b. Because permeability is an anti-reflexive and anti-symmetric relation, the *IP graph* of a page is a directed acyclic graph (DAG). The *IP graph* of the page of Figure 1 is depicted in Figure 2.

3.2.6 Vector space model

To index a block, we customize the classical vector space model [11] with importance and permeability. The index of a block b is a vector $(w(b, t_1), \ldots, w(b, t_N))$ where N is the cardinality of V and $w(b, t_i)$ is the weight of term t_i in the index of b. The way in which this weight is calculated is explained in Section 3.2.7. A query is also a vector $(w(q, t_1), \ldots, w(q, t_N))$. The answer to a query is the set of blocks most similar to this query. The similarity between a block b and a query q is computed by the classical cosine formula: $sim(b, q) = \frac{\vec{b} \cdot \vec{q}}{\|\vec{b}\| \cdot \|\vec{q}\|}$ where $\vec{b}$ (respectively $\vec{q}$) is the vector associated with block b (respectively query q).

3.2.7 Block indexing

The index of a block depends on its content and on the content of its predecessors in the *IP graph*. Given a page p and its *IP graph* $IPG(p)$:

1. the *local* index of a block with empty content is the null vector,
2. the *local* index of a block b in p whose content is non empty is the $\vec{bl}$ vector where the weight of term t is $tf(t,b) \times idf(t)$ where $tf(t,b)$ is a function of the number of occurrences of t in the content of b and $idf(t)$ is a function of the number of occurrences of t in the content of all the blocks in database DB,
3. the index of a block b in p is the vector

$$\vec{b} = \alpha(b) \times \left(\vec{bl} + \sum_{k=1}^{m} (\beta(b_k, b) \times \vec{b}_k) \right) \tag{1}$$

where $\alpha(b)$ is the importance of b, $\vec{bl}$ is its local index, $\vec{b}_1, \ldots, \vec{b}_m$ are the indexes of the m predecessors of b in $IPG(p)$ and $\beta(b_k, b)$ is the permeability of b to b_k.

Equation (1) expresses that in order to produce the index of a block, its local index is enforced by the partial or total inheritance of the indexes of its predecessors in the *IP graph* and weighted by its importance.

An alternate block indexing schema would be that of equation (1′):

$$\vec{b} = \alpha(b) \times \vec{bl} + \sum_{k=1}^{m} \left(\beta(b_k, b) \times \vec{bl}_k \right) \tag{1'}$$

Equation (1′) is irrelevant for our current block interpretation, internal node importance being *not* taken into account, since the local index of an internal node is empty: only the importance of the leaves would impact on the weights of the internal nodes. However equation (1′) might be relevant for other applications. To illustrate this, take the example of a page with two embedded blocks. The *IP graph* of this page is depicted in Figure 3. From equation (1) we have: $\vec{b}_1 = \alpha_1 \times \beta \times \alpha_2 \times \vec{bl}_2$. In

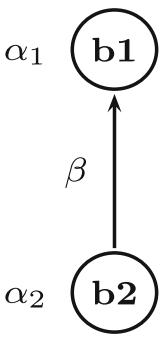


Figure 3 A simple *IP graph* for a page with two embedded blocks.

other words the inherited weight is not only sensitive to the target block importance but also to the source block importance while with the indexing schema induced by equation (1′) only the source block importance has an impact on the inherited weight: $\vec{b_1} = \beta \times \alpha_2 \times \vec{bl_2}$. In the following, block indexing is that of equation (1).

Let $b_1, \ldots, b_m$ be the m blocks of page p, $ALPHA$ be the diagonal $m \times m$ matrix whose (i, i) element is $\alpha(b_i)$, and $BETA$ be the $m \times m$ matrix whose (i, j) element is $\beta(b_i, b_j)$. Let B (BL) be the column-matrix whose $(i, 1)$ element is the (local) index of block b_i. Then from (1) we have:

$$B = \left(I_m - ALPHA \times {}^tBETA\right) - 1 \times ALPHA \times BL \tag{2}$$

where I_m is the $m \times m$ identity matrix. Equation (2) allows to compute the index of each block in a page as a function of the (local) index of each block, given its block importance and the permeability of its predecessors. It is easy to show that since the *IP graph* is a directed acyclic graph (DAG), the determinant of the matrix $(I_m - ALPHA \times {}^tBETA)$ is not null and therefore that equation (2) has a unique solution.

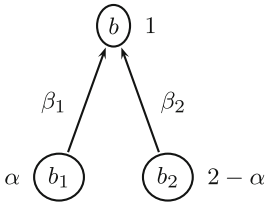
3.2.8 Properties of inheritance on indexing

Two important properties are offered by the BlockWeb model:

- **Best Entry Point Property**. *When a query is contained*[6] *in the content of two blocks, the most specific one in the block hierarchy is returned first.*
- **Inheritance Indexing Property**. *The more important a child block is, the more its content contributes to the index of its parent block.*

The Best Entry Point property is a consequence of our choice (see Section 3.2.6) to use the cosine similarity formula: given two blocks b_1 and b_2 and a query q such that $\vec{b_1} \cdot \vec{q} = \vec{b_2} \cdot \vec{q}$, $sim(b_1, q) \leq sim(b_2, q)$ if $\| \vec{b_1} \| \geq \| \vec{b_2} \|$. The Inheritance Indexing Property is a consequence of our choice (see Section 3.2.3) to constrain the sum of the importances of the n children of a block to be equal to n: the more the importance of a child block increases, the more the importances of its siblings decreases.

In the following, we give a proof of these two properties in a particular case. Let us consider a very simple page composed of three blocks: a block b which contains two blocks b_1 and b_2. The *IP graph* of this page is the following:



[6] We say that a query is contained in the content of a block when every term of this query appears in the content of this block.

Notice that, block b has importance 1 because it is the only block at its level. Furthermore let us suppose that:

- the content of b is empty and the content of b_1 is disjoint from the one of b_2 (for example: $content(b_1) = \{OTIM, query, language\}$ and $content(b_2) = \{PIMC, whale\}$),
- the terms of a query q are contained in the content of b_1 but not in the one of b_2 (for example: $q = \{query, language\}$).[7]

Although the following proof is done for two siblings it is easy to show that it holds for more than two siblings.

Property 1 *$sim(b, q)$ increases with the importance α of block b_1.*

According to the cosine similarity (see Section 3.2.6), we have:

$$sim(b,q) = \frac{\vec{b} \cdot \vec{q}}{\left\|\vec{b}\right\| \cdot \left\|\vec{q}\right\|} = \frac{\left(\alpha\beta_1 \vec{bl_1} + (2-\alpha)\beta_2 \vec{bl_2}\right) \cdot \vec{q}}{\left\|\alpha\beta_1 \vec{bl_1} + (2-\alpha)\beta_2 \vec{bl_2}\right\| \cdot \left\|\vec{q}\right\|} \tag{3}$$

Let us remark that $\vec{bl_1}$ and $\vec{q}$ are orthogonal to $\vec{bl_2}$ because (i) content of b_1 and content of b_2 are disjoint and (ii) q has no intersection with the content of b_2. Then $\vec{bl_2} \cdot \vec{q} = 0$ and $\| \vec{b} \|^2 = (\|\alpha \ \beta_1 \vec{bl_1} \|)^2 + ((2-\alpha) \ \beta_2 \| \vec{bl_2} \|)^2$. Let $P = \vec{bl_1} \cdot \vec{q}$, and L_q be the norm of $\vec{q}$, L_1, L_2 be the respective norms of $\vec{bl_1}$ and $\vec{bl_2}$. We have:

$$sim(b,q) = \frac{P\alpha\beta_1}{L_q\sqrt{\left(L_1\alpha\beta_1\right)^2 + \left(L_2(2-\alpha)\beta_2\right)^2}} \tag{4}$$

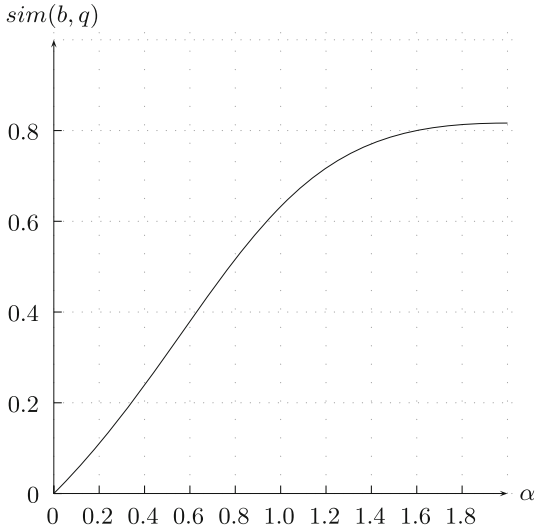
Given that P, β_1, β_2, L_q, L_1 and L_2 are positive real numbers (b_1, b_2 and q being not empty) which are not dependent on α, it is easy to show that $sim(b, q)$ is an increasing function of α in the interval [0, 2]. It increases from 0 to $\frac{P}{L_1 L_q}$ when α varies between 0 and 2.

Corollary (Inheritance Indexing Property) *The more important a child block is, the more its content contributes to the indexing of its parent block.*

For example, let us consider the case where $\beta_1 = \beta_2 = 1$ and $V = \{t_1 = language, t_2 = OTIM, t_3 = PIMC, t_4 = query, t_5 = whale\}$, and block b_1 is indexed by "language", "OTIM" and "query", block b_2 is indexed by "PIMC" and "whale",

[7] As a matter of fact, one can generalize the proof to the case where the intersection of the content of b_1 and the content of q is disjoint from the content of b_2.

and query q is "query language". We have $\vec{bl}_1 = (1, 1, 0, 1, 0)$, $\vec{bl}_2 = (0, 0, 1, 0, 1)$, $\vec{q} = (1, 0, 0, 1, 0)$ and the graph of the similarity function $sim(b, q)$ is the following:



We now turn to the proof of the best entry point property:

Property 2 (Best entry point property) When a query is contained in two blocks, the most specific block in the hierarchy has a larger (cosine) similarity with the query.

Let us compare the similarity of query q with blocks b_1 and b (the parent of b_1). We have:

$$sim(b_1, q) = \frac{\alpha\beta_1 \vec{bl}_1 \cdot \vec{q}}{\left\|\alpha\beta_1 \vec{bl}_1\right\| \cdot \left\|\vec{q}\right\|} = \frac{P\alpha}{L_q\sqrt{(L_1\alpha)^2}} \tag{5}$$

and according to (4):

$$sim(b, q) = \frac{P\alpha}{L_q\sqrt{(L_1\alpha)^2 + \big(L_2(2-\alpha)(\beta_2/\beta_1)\big)^2}} \tag{6}$$

Therefore, because $((L_1\alpha)^2 + (L_2(2-\alpha)(\beta_2/\beta_1))^2 \geq (L_1\alpha)^2)$, we have $sim(b_1, q) \geq sim(b, q)$: gain $G = sim(b_1, q)/sim(b, q)$ is greater than 1. This concludes the proof of the Best Entry Point Property. In our example, we have $sim(b_1, q) = 0.82$ and $sim(b, q) = 0.63$. Block b_1 is effectively the best entry point.

The following remarks on the behavior of

$$G = sim\big(b_1, q\big)/sim(b, q) = \frac{\sqrt{\big(L_1 \cdot \alpha\big)^2 + \big(L_2 \cdot (2-\alpha) \cdot \big(\beta_2/\beta_1\big)\big)^2}}{\sqrt{\big(L_1 \cdot \alpha\big)^2}}$$

are noteworthy:

1. in contrast to property 1, for given values of β_2/β_1, L_1 and L_2, the smaller α, the larger G, the better b_1 is an entry point. In other words importance blurs the precision of localization.

2. as expected, for given values of α, L_1 and L_2, the larger the ratio β_2/β_1, the larger G. In other words, the less the parent inherits the content of b_1, the better b_1 is an entry point.
3. as expected by the cosine similarity measure, for given values of α, β_2/β_1 and L_1, the larger the norm L_2 of b_2, the larger the norm of parent b and therefore the larger G. In other words the larger the norm of the parent block is in comparison with the norm of b_1, the better b_1 is an entry point.

Of course, if query q is itself partially or fully contained in block b_2 the best entry-point could be block b in place of block b_1. In our example, if block b_2 is not only indexed by "PIMC" and "whale" but also by "query" and "language", we would have $sim(b_1, q) = 0.82$ and $sim(b, q) = 0.85$. The best entry point is now the block b. In Section 5, we present an experimental validation of this property.

3.2.9 Modeling of a web page

Several strategies to model a web page corpus according to the BlockWeb model are possible. In the case where the layouts of the pages of the corpus share the same schema, a block hierarchy schema and an *IP graph* schema can be built as follows:

1. The block hierarchy schema is built.
2. A coefficient of importance is assigned to each child block of a parent block in accordance to the block context model (see Section 3.2.3) i.e. the sum of the importances of the n children blocks of a block must be equal to n.
3. A permeability edge with value $\beta = 1$ is drawn between each children block and its parent block. Setting $\beta < 1$ would have the effect to decrease the contribution of the corresponding block to its parent block such that it has been set up by its importance coefficient in step 2.
4. Transverse permeability edges may be drawn between two blocks which are not in an ancestor relationship, for example from a textual block to an image block in order to index the latter. A transverse edge from a textual block to another textual block allows to strengthen the content of the latter. When drawing such transverse permeability edges no cycles must be introduced in the *IP graph* which is constrained to be a directed acyclic graph (see Section 3.2.5).

This strategy is the one used to model the page of Figure 1. However recall that other strategies are applicable which consist in simplifying and reconfiguring the visual block hierarchy for the purpose of specific applications. We give an example of such another strategy in Section 5.2 where we present an application of the BlockWeb model for image indexing by permeability.

4 An engine for block extraction and indexing

This section is devoted to the description of the prototype designed and implemented for block extraction, indexing and querying according to our model. Recall that the main objective of this paper is to define a fine grained indexation system dedicated to web pages. As it is fine grained, a dedicated tool is needed to help the system administrator in defining the indexing parameters (importance and permeability) for a large corpus of documents using rules based on logical and visual properties.

4.1 General architecture

Web pages are described using HTML, XHTML or even arbitrary XML. Page visual rendering is generally described using CSS and the pages can be dynamic (on the client side) thanks to embedded script languages (like Javascript). Indexing and querying the content and the logical structure is interesting but relying only on the logical structure is not enough. To illustrate this, Figure 4 shows the HTML source corresponding to the web page shown in Figure 1. If we use only the logical structure (the `body` element) important visual information is lost: complex CSS rules (in the `style` element) need to be applied to reveal that the first article (the first `div` element) is more important (according to its area). That is why, to achieve block indexing of web pages, our architecture has three main objectives:

- the first one is to combine the logical structure of the page described in the DOM representation and the visual rendering of the page (*i.e.* like a human user would see it in a web browser) in an abstract block hierarchy,
- the second one is to provide a language which enables the system administrator to automatically produce the *IP graph* (and so the indexing) on top of the previous block hierarchy,
- the third one is to provide a query language to query both the original logical structure and content, and the result of the indexing (a vector of terms associated with each block).

Figure 4 HTML and CSS source of the document in Figure 1

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>DYNI news</title>
<style type="text/css">
h1 {border:solid 1px;margin:.25em;
    padding:.25em;text-align:center;}
body {border:solid 1px;}
.article {border:solid 1px;margin:1%;
    width:46%;padding:.25em;background:#DDD;}
body>div:nth-child(2n).article {float:right;}
body>div:nth-child(2n+1).article {float:left;}
body>div:nth-child(2) {float:none;margin:1%;
     width:92%;color: black;background:#999;}
.spacer {clear:both;}
</style>
</head>
<body>
<h1>DYNI news</h1>

<div class="article">
<h2>BlockWeb, a new IR Model</h2>
<p>BlockWeb is a model for indexing  ...</p>
<div class="AuthorsPictures">
  <img src="nobody.png">
  ...
</div>
</div>
<div class="article"><h2>OTIM</h2><p>OTIM ...</p></div>

<div class="article"><h2>PIMC</h2><p>PIMC ...</p></div>

<div class="spacer"> </span>
</body>
</html>
```

Indexing is done in four steps as shown in Figure 5:

1. the web page is transformed into a block hierarchy keeping pointers to the HTML source,
2. the indexing parameters (ALPHA and BETA matrices) are calculated using generic rules applied on the previous block hierarchy. These rules are expressed in an indexing sheet using a dedicated XML language called XIML.
3. the local indexing of each block is computed and inserted into the XML document representing the block hierarchy which is stored in an XML database,
4. the index of each block is calculated by propagation (equation (2)) using ALPHA and BETA matrices produced at step 2. Each index is also inserted into the corresponding block in the XML database.

Querying, which is not detailed in this paper, is done using XQueries on the block hierarchy and on a XMLized version of the original web page. The system provides (i) functions dedicated to information retrieval (similarity measures and IR-queries evaluation on indexed blocks) and (ii) functions dedicated to navigation between blocks in the hierarchy and corresponding elements in the web pages. To improve the performance, indexes for terms are stored in a relational database and used by the XQuery extension.

4.2 Transformation of a web page into a block hierarchy

As already mentioned, in more and more applications the structure and rendering of the pages of a given site does not change with time, is often shared by several sites and is similar to that of other sites within the same community or domain. For example, the following french newspapers web sites http://www.lefigaro.fr/, http://www.lemonde.fr/, http://www.liberation.fr/ have all two main classes of pages (welcome pages, articles, ...) and the welcome page is composed of a list of the beginnings of articles and by a list of short news. A hierarchy of blocks can be produced from a web page by either automatic mapping (using segmentation algorithms) or manual mapping.

To produce the block hierarchy, we did not use a visual segmentation algorithm like the XY-cut algorithm or the VIPS algorithm, which did not provide a schema fine enough so as to represent the actual block hierarchy. Since the latter is known in advance (it does not change from one page to another), it was "manually" generated

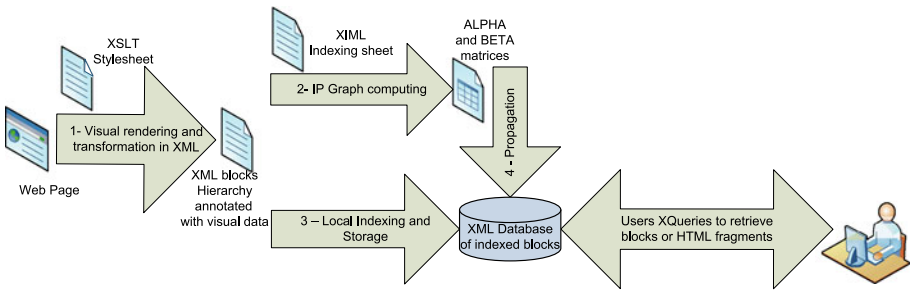


Figure 5 Functional diagram of the engine.

Figure 6 XML Syntax for a hierarchy of blocks extracted from the document in Figure 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<Block id="dyni.html#/html/body"
       label="news" area='786432'>
  <Block id="dyni.html#/html/body/div[1]"
         label="article" area='160000'>
    <Block label="title"
        id="dyni.html#/html/body/div[1]/h2[1]">
      <Text>BlockWeb, a new IR Model</Text>
    </Block>
    <Block label="content"
        id="dyni.html#/html/body/div[1]/p[1]">
      <Text>BlockWeb is a model for...</Text>
    </Block>
    <Block label="pictures"
        id="dyni.html#/html/body/div[1]
            /div[@class='AuthorsPictures'][1]"/>
  </Block>
  <Block id="dyni.html#/html/body/div[2]"
         label="article" area='80000'>
   ...</Block>
  <Block id="dyni.html#/html/body/div[3]"
         label="article" area='80000'>
   ...</Block>
</Block>
```

using XSLT stylesheets. Moreover, web pages are often produced using content management systems, then it is easy to define stylesheets to transform pages from a "wide" classe of web sites to the same abstract block hierarchy.

After the production of the block hierarchy, we layout the web page with the HTML engine Gecko[8] at a standard resolution of 1024×768. Then, we extract the visual properties of each block from the corresponding visual box.

The hierarchy of blocks and their corresponding visual data can easily be represented as an XML document. Figure 6 shows the simplified XML representation of the block hierarchy document corresponding to the HTML document of Figure 1. Only one visual property (area) is shown, the others (geometry, position, font, colors ...) have been hidden to ease the reading. Notice that the vectors of terms are not yet calculated at this step but they are later inserted as an `index` element in each block.

4.3 XIML a language to define block indexing parameters

To help the application administrator to tune the indexing of a web site document, we have defined the "XML Indexing Management Language" (XIML). An XIML sheet works like a stylesheet but is dedicated to indexing: it applies to the XML representation of a block hierarchy and produces the *ALPHA* and *BETA* matrices. Figure 7 shows the XML syntax of the XIML sheet used to index the document described in Figure 1 (*i.e.* to produce the *IP graph* from Figure 2). Notice that, the indexing models used in the experiments in the next section have also been generated with XIML. An XIML sheet is composed of two parts: one to define

[8] http://developer.mozilla.org/fr/docs/Gecko

```
<?xml version='1.0'?>
<Rules>
 <Alpha>
  <!-- Default importance is set to 1 -->
  <ItemSelector select="//Block" value="1" />
  <!-- The importance of the title of an Article is set to 2 -->
  <ItemSelector select="//Block[@label='article']/Block[@label='title']"
                value="2"/>
  <!-- The importance of an Article is proportional to its surface -->
  <ItemSelector select="//Block[@label='article']"
   value="@area div sum(../Block[@label='article']/@area) *
                        count(../Block[@label='article'])"/>
 </Alpha>

 <Beta>
  <!-- Every Block is 1-permeable to its children -->
  <Relation target='//Block' source='parent::*' value='1'/>
  <!-- and images are more permeable to the title of the same article
       than to its content and they do not propagate to their article -->
  <Relation target='//Block[@label="article"]' source='.//Block[@label="image"]'
        value='1'/>
  <Relation target='//Block[@label="image"]'  source='../Block[@label="title"]'
        value='1'/>
  <Relation target='//Block[@label="image"]' source='../Block[@label="content"]'
      value='0.3' />
 </Beta>
</Rules>
```

Figure 7 XML Syntax for an XIML instance.

the rules dedicated to importance computing (the `Alpha` element) and one for the permeability (the `Beta` element).

The `Alpha` element contains a sequence of item selectors. Each item selector selects a sequence of blocks using an absolute XPath expression (`select` attribute). The value of the importance $\alpha(b)$ for each selected block b is computed with another XPath expression absolute or relative to b (`value` attribute). If a block is selected more than once, the last selection takes precedence. In the example, the first element `ItemSelector` sets the default importance of every block to 1, the second one defines an exception for the article title blocks, and the last one shows how XPath can be used to dynamically compute the importance according to visual properties.

The `Beta` element contains a sequence of relations. A relation is made of three XPath expressions: *target*, *source* and *value*. Every relation adds a set of edges to the *IP graph* to represent: $\beta(s,t) = val(value(s,t))$ for all blocks t selected by *target*, blocks s selected by $source(t)$ (source is absolute or relative to t). The expression *value* can use the variables `$target` and `$source` binded to s and t. If an edge is selected more than once, the last value takes precedence. In the example, by default children indexing is propagated to parent indexing (first `Relation` element). The three last relations state that images are more permeable to the title of the same article than to its content and that they do not propagate their indexing to their article.

4.4 Implementation of the prototype

The engine is implemented in Java. It satisfies the functional decomposition of Figure 5: it includes the web pages integration system (and the stylesheets required for three French electronic newspapers and the ImagEval experiment) and a processor for the XIML language. The indexed block hierarchy associated with

each web page (computed using XIML) is stored in the EXIST[9] XML database management system. A set of functions extending XQuery is used (i) to transform a set of words into a query (stemming, weight computation, ...), (ii) to query the index to retrieve the matching blocks, and (iii) to retrieve the corresponding HTML fragments. Classical XQuery statements can also be used to combine information retrieval, structured querying and content constraints. Some dedicated indexes have been stored in a relational database to speed up the indexing and querying processes.

4.5 Manual indexing with the help of graphical tools

We have presented an engine to extract and index blocks from web pages. Even though the engine requires manual steps using stylesheets to produce blocks and XIML sheets to define the *IP graph* the amount of work is not prohibitive for three reasons. First, as aforementioned, the number of different stylesheets is not large as many web pages especially in the same application classes have close visual structures and so basic templates can be provided and adapted. Second, graphical tools integrated in a web browser allow users to visually create XPath expression by selecting HTML elements or blocks. These XPath expressions are the ones used in XIML for producing the *IP graph*.

Examples of such tools are the Firefox extensions FireXPath and AutoPager. Third, if a general block hierarchy schema can be exhibited, then standard basic parameters (α and β) can be learned. The latter point is illustrated in Section 5, in the context of image indexing.

5 Evaluation

The objective of this section is to experimentally illustrate two functionalities of our model: (1) finding the most specific block containing a query (best entry point), (2) the impact of permeability on image indexing. The impact of importance on indexing and querying is left as a future work.

5.1 Best entry point

An Intel® Bi-Xeon™ at 3.20GHz computer was used for the experiments with 8 GB of RAM. We chose as a corpus, electronic versions of one French daily newspaper in the period running from January 2008 to April 2008. 1,300 web pages were extracted from these documents. Note that all blocks have same importance $\alpha = 1$. The total number of blocks is 48,000. The indexing vocabulary is the set of word stems encountered in the leaf blocks (22,639 terms), excluding the stop words. The weight of terms in local vectors was computed using the $tf * idf$ formula where tf is the number of occurrences of the term in the block and idf is the inverse *page* frequency. The indexes of blocks are computed according to equation (2). The *IP graph* of the page is presented in Figure 8. 500 queries are generated with two terms randomly chosen inside the same paragraph (at depth 5 in the tree decomposition).

[9] http://exist.sourceforge.net/

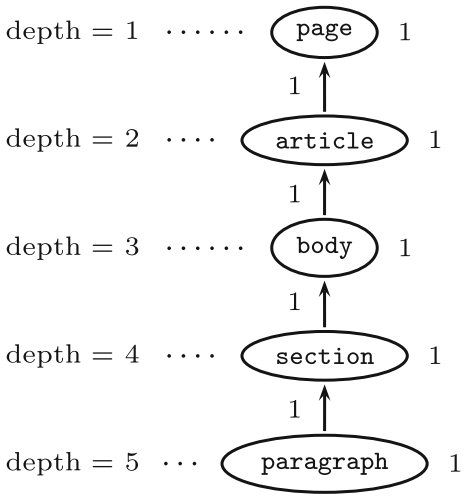


Figure 8 IP graph of the pages of the corpus of experiment 1 (Finding the best entry point).

The capacity of our model to find the best entry point is based on two factors: the decomposition of pages into blocks and the choice of the cosine similarity $\left(sim(b,q) = \frac{\vec{b}\cdot\vec{q}}{\|\vec{b}\|\cdot\|\vec{q}\|}\right)$ which normalizes the length of block textual content (see Sections 3.2.6 and 3.2.8 above). In order to evaluate the contribution of the normalization, we compare answers obtained with the cosine similarity with those obtained with the inner product similarity $sim(b,q) = \vec{b}\cdot\vec{q}$ which does not normalize the length of block textual content.

The measures reported in Table 1 are the average rank and average depth (in the page tree) of the highest rank answers including either the two query terms or only one of the query terms. Two blocks having same score have same rank. Ranks are ≥ 0 and depth is between 2 and 5. For example, with cosine similarity, for answers with the two query terms, note that since the highest ranked block is not zero, there are blocks with better scores with one term only (on the average 1.158). Several conclusions, consistent with the expected impact of normalization can be drawn from these measures.

(i) The cosine measure favors blocks with higher depth and therefore finds better entry points as expected by Property 1 (see Section 3.2.8). Indeed for example with two terms, the average depth of best answers is almost 4 with the cosine similarity (the depth would be 5 if in all cases, the two terms of the query appear only in a paragraph, and not as well in another paragraph, section ...); in contrast the depth is only 2.660 with the inner product.

Table 1 Best entry point: rank and depth of the first block in the answer set satisfying "number of terms" condition.

Similarity	Nb term	Rank	Depth
Cosine	2	1.158	3.992
	1	2.181	4.602
Inner product	2	0.614	2.660
	1	1.491	3.082

Table 2 Best entry point: rank of the first block in the answer set satisfying "depth" and "number of terms" conditions.

Similarity	Nb term	Depth 5 (paragraph)	Depth 4 (section)	Depth 3 (body)	Depth 2 (article)
Cosine	2	2.814	8.780	10.832	8.468
	1	2.541	13.356	20.233	14.914
Inner product	2	5.092	2.362	1.194	0.614
	1	5.205	3.197	1.675	1.480

(ii) Compared to the cosine similarity, the inner product similarity favors answers with two terms wrt answers with one term: on the average only 0.614 answers with one query term have a higher rank, while with the cosine similarity the average is 1.158. However this result must be analyzed more carefully. With the inner product similarity, blocks containing a query have the same similarity whatever their content length which is not really what one expects. This is confirmed by the fact that this better average rank (0.614 *vs* 1.158) is associated with a smaller depth (2.660 *vs* 3.992) corresponding to larger blocks (`body` vs `section` blocks approximately). On the other hand with the cosine similarity, blocks totally containing a query have a similarity inversely proportional to their content length and even more blocks containing only partially this query may have a higher similarity than blocks containing totally this query if they have a shorter content length which is in favor of a better answer precision.

(iii) Both measures tend to rank (on the average) queries with two terms prior to queries with only one term, which is what one expects from a nice query engine; indeed, the rank with cosine (inner product) is 1.158 (0.614) for two terms which is less than 2.181 (1.491) for one term. These observations are confirmed by the Table 2 where the rank of the first block at depth 2, 3, 4 or 5 with the two terms (with only one term) is displayed. As an example, with the cosine similarity, on the average, there are 8.780 blocks whose score is higher than the first `section` block score answering the query. As expected, with the cosine similarity, the paragraph blocks in the answer are found first, while with the inner product, articles are found first!

5.2 Image indexing

This section is devoted to the evaluation of the impact of the block structure on web images indexing and retrieval. The experiment is conducted in the framework of the recent international ImagEval campaign, which aims at evaluating several tasks about image indexing and retrieval. One of these tasks (task2) aims at assessing how text techniques improve image search by similarity. Its web page corpus is described in the following section.

For this experiment, we index every image contained in a web page. For each image, we identify four interesting blocks, as illustrated in Figure 9:

- *Page* block: the whole page. This block is the concatenation of all text blocks in the page.
- *Title* block: the block containing the title of the page.

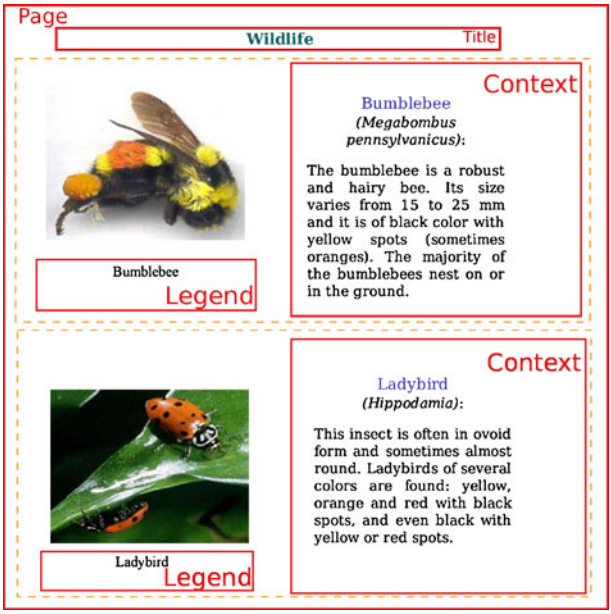


Figure 9 A web page of the ImageEval corpus and its block decomposition.

- *Legend* block: This block is constructed with the available index of the image block. Its textual content is the concatenation of the image legend (the `alt` attribute of the image, which gives an alternative textual content for the image if the browser is not able to display the image), and words extracted from the image file name.
- *Context* block: Let a be the farthest ancestor block of the image, apart from the Page block, which contains no other image. The Context block is the concatenation of a and its descendant blocks apart from the Legend block.

To assess the quality of image indexing we compute the Average Precision which is equal to the integral of the Recall-Precision curve, and the Mean Average Precision (MAP) over all the queries for the five following permeability schemas:

1. the Page schema which consists in indexing an image by the whole text of the page: $\beta(Page, Image) = 1$ and $\beta(b_1, b_2) = 0$ for all other pairs (b_1, b_2) of blocks,
2. the Title schema which consists in indexing an image by the page title only: $\beta(Title, Image) = 1$ and $\beta(b_1, b_2) = 0$ for all other pairs of blocks,
3. the Legend schema which consists in indexing an image by its legend only: $\beta(Legend, Image) = 1$ and $\beta(b_1, b_2) = 0$ for all other pairs (b_1, b_2) of blocks,
4. the Context schema which consists in indexing an image by its context only: $\beta(Context, Image) = 1$ and $\beta(b_1, b_2) = 0$ for all other pairs (b_1, b_2) of blocks,
5. the BlockWeb schema, depicted in Figure 10, which consists in indexing an image by cumulating the Page index, the Title index, the Legend index, and the Context index: $\beta(Page, Image) = \beta_{page}$, $\beta(Title, Image) = \beta_{title}$, $\beta(Legend, Image) = \beta_{legend}$, $\beta(Context, Image) = \beta_{context}$, and $\beta(b_1, b_2) = 0$ for all other pairs (b_1, b_2) of blocks, where β_{page}, β_{title}, β_{legend} , and $\beta_{context}$ are learnt as explained in the following section.

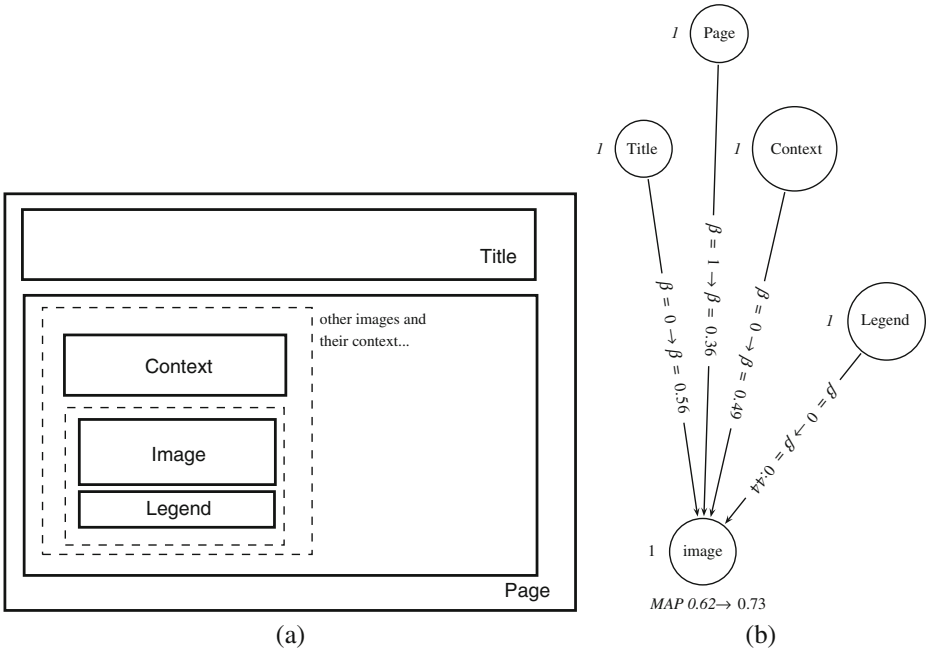


Figure 10 BlockWeb permeability schema for web page image indexing. Permeability values prior learning (only the page block indexes the image) and after learning, and the MAP improvement on the test set are given.

For each permeability schema the MAP is computed: MAP(Page), MAP(Title), MAP(Legend), and MAP(BlockWeb).

5.2.1 ImagEval web page corpus

The dataset is built from the 2009 web pages of the ImagEval task2 corpus (see http://www.imageval.org/e_presentation.html). This campaign aims at assessing how text techniques improve image search by similarity. As this task is "search on the web" oriented, the database has been created by extraction of pages from the Web, especially from Wikipedia for copyright reasons. The web pages have been found using classical search engines (Google and Alltheweb). The database is composed of a list of URLs and the corresponding text and images files. Query terms and web pages are in French. In this paper the terms and examples are translated into English. Pages were uploaded using topics like: "Eiffel Tower", "Clown Fish", "Uluru rock", "Ethiopian flag", using Wikipedia. This campaign focused on "encyclopaedic" and "picturable" topics: animals, places, monuments and objects. A query is composed of one to five terms.

A complete description of the ImagEval campaign and Average Precision metrics can be found in [10]. The authors of the latter paper addressed the issue of query complexity due to the presence of image synonyms in the test set. For example, there are photos of clowns, and some of fish. But none of them are good answers for the "clown fish" query. For solving this issue, a fusion of text and image content analysis for efficient web image retrieval is proposed in [13].

Table 3 Number of target images in the global set of 4,705 images for 25 queries.

Number of target images for the query	The 52 terms of the 25 queries
11	{flag, Ethiopian}
17	{flag, european}
17	{screwdriver}
12	{ball, tennis}
16	{planete, venus}
28	{roch, Perce}
24	{zebra}
41	{fish, clown}
42	{roch Uluru, Uluru, Ayer, Ayer-roch, roch}
90	{lemon}
47	{fall, Niagara}
41	{tower, Eiffel}
56	{big, wall, China}
43	{flow, lava}
26	{lawyer, fruit}
21	{cat, siamois}
22	{Guernica, Picasso}
18	{ground, tennis}
12	{statue, Liberty}
9	{Joconde}
8	{bear, teddy}
7	{people-tree, tree}
5	{map, Norway}
6	{ladybird, insect}
7	{liberty, guide, people, Delacroix}
626	

Tackle this issue. In the following section, we show that the BlockWeb model is a generic approach to deal with this problem. In our experiment, the dataset contains 4,705 web pages including images, out of which 626 appear in one of the queries described in Table 3.[10] The indexing terms are the 52 terms occuring in the queries.

We randomly split this corpus into two sets of equal size, the training set and the test set, such that both sets have a similar number of web pages including images satisfying each query: the *target images*. The training set is used to learn the four permeability values β_{page}, β_{title}, β_{legend}, and $\beta_{context}$ by a gradient descent optimization [7], with as constraints, maximizing the target images similarities and minimizing the non target images similarities. The test set is used to validate these four automatically learned permeability values.

5.2.2 Results and discussion

The training lasted 2 minutes on an usual Dell laptop. The evolution of the permeability values during 30 training stages is depicted in Figure 11, which shows that only a few training stages are necessary. The learnt permeability values are the following:: $\beta_{page} = 0.36$, $\beta_{title} = 0.56$, $\beta_{legend} = 0.44$, $\beta_{context} = 0.49$.

[10] In the original online corpus 906 images appear in one of the queries, but since 2006, some web pages have been removed.

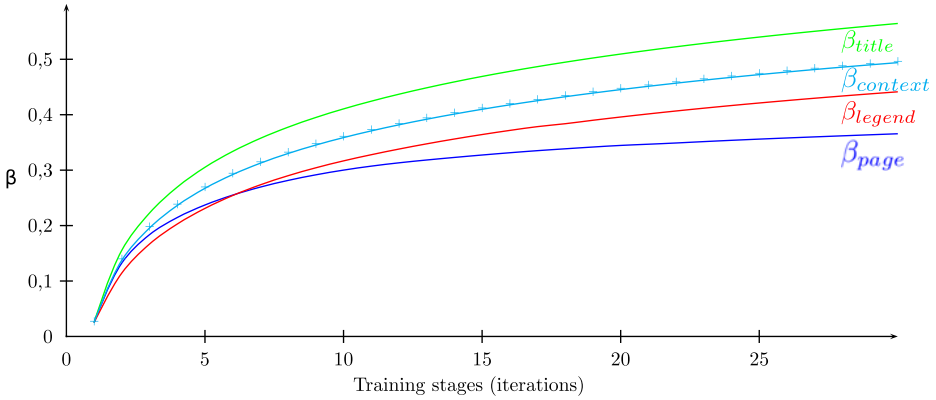


Figure 11 Evolution of the permeability values (β_{page}, β_{title}, β_{legend}, $\beta_{context}$) during the training stage.

Table 4 gives the MAP for the fives permeability schemas (Page, Title, Context, Legend, BlockWeb). Note that MAP values on the training set and on the test set are close (naturally, MAP(BlockWeb) is smaller for the test set), even for the BlockWeb schema, demonstrating that the permeability values on the training set well generalizes on unknown web pages.

Looking at Table 4, the first observation is that when we do not learn the permeability values, the Page block alone provides the best MAP to queries. It is probably because a Page block contains all the terms of the page what favours a better recall. Surprisingly it is the Title block which provides the second best MAP although it was the Context which was expected since it is much closer to the image block. A possible explanation is that page titles are well chosen and that their conciseness favours the precision in contrast to image contexts which are more developped.

The second observation concerns the Legend. For this corpus, the Legend does not always exist. Sometimes, but not always, the page designer provides, often manually, an alternative text, which might be the best information source to index an image. Therefore Legend is an example of bad quality block which justifies again the use of an optimal combination of all page blocks for image indexing, as provided by the BlockWeb model.

The last noteworthy result is that the Block permeability schema provides a significantly better quality of the answers. Indeed $MAP(BlockWeb) = 0.73$ *vs* $MAP(Page) = 0.62$, resulting in a relative improvement of 16%.

Table 4 MAP of the 25 queries for each permeability schema.

	Page schema	Title schema	Legend schema	Context schema	BlockWeb schema
MAP on the training set	0.634	0.510	0.128	0.241	0.740
MAP on the test set	0.622	0.485	0.118	0.262	0.726

These results show not only the interest of the BlockWeb model strategy compared to image indexing based on all terms of the Page block. They also bring out that in this corpus, images are poorly indexed by using only the Title block or the Context block or the Legend block alone. Although the latter statement is true for the majority of pages of the corpus, there are of course some counter examples.

Finally we observed that only a few number of positive examples are necessary to have a correct training. Half of the training set is enough to get similar results with the ones of Table 4. This suggests that for any other application domain, the few hand labeled web pages needed for the training can be downloaded from any web page browser, and refined to quickly obtain fifty labeled web pages containing positive target images. The labeling of a web page is a set of booleans, set to *yes* for each image in the web page satisfying the query, and set to *no* otherwise. The negative samples can be simply randomly uploaded from the web. In conclusion, the effort needed for assessing the ground truth of the training set keeps reasonable.

6 Conclusion

This paper was devoted to a novel approach dedicated to Information Retrieval in web pages. We have proposed a model, called BlockWeb, and an architecture for indexing segmented web pages taking into account their visual rendering and their multimedia content. The concept of permeability was introduced as a flexible tool for indexing blocks in various applications.

The three key features of our model are: (i) web page segmentation into blocks including data on visual rendering, (ii) use of block importance, (iii) and use of block permeability to improve other blocks indexing.

We have presented an engine which implements our model. It enables an user to define the transformation from web pages to block hierarchies represented as XML documents. A dedicated language called "XML Indexing Management Language" (XIML) has been defined to express generic rules to compute the permeability β between blocks and the importance of each block (i.e. the *IP graph*). The rules can rely on the structure of the block hierarchy, on the content of the blocks, and on automatic annotations about the real visual rendering in a web browser. The result of the indexing is a corpus of XML documents which represents the blocks hierarchy with visual annotations and pointers to the real HTML elements, the *IP graph* for each page, and the final indexing of each block.

We have shown that the model is adequate to find the best entry point in a page, i.e. to locate more precisely in the page the information one looks for. The model was first illustrated and assessed on a dataset of electronic news. One conclusion of this first experiment is that the classical cosine measure (or any measure taking into account the "size" of the block) is well adapted to block indexing based on keywords inheritance from their component blocks. As a future work we intend to look at the impact of block importance.

We have also presented a second experiment on a dataset from the ImagEval campaign in which our model is applied to indexing images occurring in web pages. The experiment shows that the indexing of an image from a weighted combination of the terms of blocks of the page as provided by the BlockWeb model is better than the indexing by all terms of the page or by the terms of blocks close to the image such

as the Legend. Joint optimization of permeability and importance coefficients will be addressed in future work.

Acknowledgements We would like to thank CEA and TRIBVN society for providing the IMAGEVAL web image Task data set. We also thank ANR AVEIR ANR-06-MDCA-002 which partially supports this research.

References

1. Bruno, E., Faessel, N., Glotin, H., Le Maitre, J., Scholl, M.: Indexing by permeability in block structured web pages. In: Proceedings of the 9th ACM Symposium on Document Engineering (DocEng 2009), pp. 70–73, (2009) (short paper)
2. Bruno, E., Faessel, N., Le Maitre, J., Scholl, M.: Blockweb: an IR model for block structured web pages. In: Proc. of 7th Int. Workshop on Content Based Multimedia Indexing (CBMI 2009), pp. 219–224. Chania, Crete, June 3–5 (2009)
3. Cai, D., He, X., Li, Z., Ma, W.-Y., Wen, J.-R.: Hierarchical clustering of WWW image search results using visual, textual and link information. In: Proc. of the 12th ACM Int. Conf. on Multimedia, MULTIMEDIA '04, pp. 952–959. ACM, New York, NY, USA (2004)
4. Cai, D., Yu, S., Wen, J.-R., Ma, W.-Y.: VIPS: A Vision-Based Page Segmentation Algorithm. Technical report, Microsoft Research (2003)
5. Cui, H., Wen, J.: Hierarchical indexing and flexible element retrieval for structured documents. In: Proc. of the 25th European Conf. on IR Research (ECIR 2003), pp. 73–87. Pisa, Italy (2003)
6. Debnath, S., Mitra, P., Pal, N., Giles, C.L.: Automatic identification of informative sections of web pages. IEEE Trans. Knowl. Data Eng. **17**(9), 1233–1246 (2005)
7. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley-Interscience (2000)
8. Ha, J., Haralick, R.M., Phillips, I.T.: Recursive X-Y cut using bounding boxes of connected components. In: Proc. of the Third International Conference on Document Analysis and Recognition (ICDAR'95), vol. 2, pp. 952–955. Washington, DC, USA (1995)
9. Lin, S.-H., Ho, J.-M.: Discovering informative content blocks from Web documents. In: Proc. of the 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 588–593. Edmonton, Alberta, Canada (2002)
10. Moëllic, P.A., Fluhr, C.: ImagEval 2006 official campaign. CEA List (2006)
11. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11), 613–620 (1975)
12. Song, R., Liu, H., Wen, J.-R., Ma, W.-Y.: Learning block importance models for web pages. In: Proc. of the 13th Int. Conf. on World Wide Web (WWW 2004), pp. 203–211. Manhattan, NY, USA (2004)
13. Tollari, S., Glotin, H.: Web image retrieval on ImagEVAL: Evidences on visualness and textualness concept dependency in fusion model. In: Proc. of the ACM Int. Conf. on Image and Video Retrieval (CIVR 2007), pp. 65–72 (2007)
14. Tollari, S., Glotin, H.: Learning optimal visual features from web sampling in online image retrieval. In: IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP 2008, pp. 1229–1232. IEEE (2008)
15. Vadrevu, S., Gelgi, F., Davulcu, H.: Information extraction from web pages using presentation regularities and domain knowledge. World Wide Web **10**(2), 157–179 (2007)
16. Yi, L., Liu, B., Li, X.: Eliminating noisy information in web pages for data mining. In: Proc of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 296–305. Washington, DC, USA, ACM (2003)
17. Zou, J., Le, D., Thoma, G.R.: Combining DOM tree and geometric layout analysis for online medical journal article segmentation. In: Proc. of the 6th ACM/IEEE-CS Joint Conf. on Digital Libraries, pp. 119–128. Chapel Hill, North Carolina, USA (2006)