Image Retrieval on the World Wide Web

by Epimenides Voutsakis

Dissertation Thesis

Technical University Of Crete

Department of Electronics and Computer Engineering

July, 2004

Abstract

Link analysis has been used very efficiently in today's search engines in order to give *important* relevant pages a higher ranking. This document describes and evaluates the use of link analysis techniques along with more traditional ranking methods, in image retrieval systems on the WWW[1]. We implement an image retrieval system for finding images on the web incorporating traditional text retrieval models combined with link analysis and image content description methods. Various query approaches and appropriate search algorithms, are also implemented and evaluated using keywords, image example or a combination of keywords and example images. We present the evaluation of these approaches based on precision and recall, and discuss their advantages and disadvantages using a large data set consisted of more than 200.000 web pages rich in logo images.

[1]World Wide Web

Contents

Chapter 1

Introduction

Search engines such as Google[www.google.com], have proven to be the most important tool for users to navigate and exploit the massive electronic content available in the World Wide Web. The majority of Internet users reach a site related to their topic of interest by submitting a few keywords in a search engine, and then browsing the search results. Thus, for a search engine to be considered as *good* it must help the user to find what he wants in a simple and quick manner. Simple in today's search engines means that the user can simply type a few words in a field of an on-line form, and quick means that in a matter of a small amount of time (usually in tenths of seconds) his query is answered with a set of relevant documents. However, the rapid growth of the WWW over the last 10 years has made this task very difficult. The search engine must be fast, it must efficiently search through 4-5 billion[1] pages in milliseconds, and it should be able to rank accurately the results at the same time. How these tasks are fulfilled by modern search engines, is described by Brin and Page at [4].

In this document we focus on ranking algorithms, which have a very crucial role in today's search engines. They affect directly the order that the search results are displayed to the users, which means that they affect how quickly a user will find what he seeks. There are a lot of issues that a ranking method

[1] at the time of writing the size of the WWW is estimated at more than 5 billion pages

must resolve in order to be considered applicable for retrieval on the WWW.

The web can be considered as a large directed graph in which a node is an electronic document (e.g. a web page) and an edge describes a link from one document to another. Such links are incorporated within electronic documents, that support links (e.g. documents written in HTML[2]). Traditionally ranking algorithms are implemented by a similarity function (between a query and a document-node on the web graph), exploiting content (e.g. text) specific similarities. Other ranking methods utilize the link structure of the web in order to provide a measure of importance for a document-node in the web graph.

In Chapter 2 first we present the basic theory about content based ranking schemes. Such schemes were developed long ago from information retrieval scientists in order to solve the problem of document relevance. These are called CBR[3] methods because they base their algorithm on a pure content comparison. These methods provide us with a standard method to measure how similar two documents are (e.g., by examining keyword-phrase occurrences or other content-specific features).

First of all, we describe the vector space model[6, 9] which utilizes keyword densities in order to find relevant documents. Then, we describe the ChainNet[10, 5] model for searching images in the web with keyword queries. At the last section of Chapter 2 we also describe a set of image specific features and ways they can be used in combination with other techniques to boost the precision of image retrieval systems, and provide additional query approaches (e.g., query by example image). While these techniques have been found to work extremely well for relatively small collections, they aren't that useful when applied on the WWW, and that's because users tend to often submit more abstract queries which result in thousands of documents. In such cases, search engines measure how important a page is (besides it's relevance to a user query) and present this document before other less important pages (by assigning to it a higher ranking). Today's search engines use link analysis techniques along

[2]**H**yper **T**ext **M**arkup **L**anguage.

[3]CBR = **C**ontent **B**ased **R**etrieval

with more traditional information retrieval techniques in order to achieve better ranking quality of their results.

Link analysis in it's simplest form was introduced as link popularity, which simply means that the amount of backlinks[4] a web site has, reflects directly or indirectly it's ranking. After this simple approach the term *link analysis* was introduced to describe the set of more sophisticated link-based ranking methods, such as PageRank[7] and HITS[2]. Analysis of the hyperlinked environment of the WWW aims to define how important a site is. That way a search engine can sort the results shown to users, not only by relevance (which CBR methods compute) but also by importance. This importance is induced by the assumption that a link between two pages denotes a recommendation by the author of the pointing page for the page that is pointed. There are two main approaches in link analysis methods. The one approach is query-independent [PageRank], while the other is query-dependent [HITS]. The query-independent approach measures the universal "reputation" of a page among the WWW, while the query-dependent approach measures the "reputation" of a page among relevant to the query documents. In Chapter 3 we describe on query independent and two query-dependent link analysis algorithms, one for text document ranking and one formulated for image ranking.

In Chapter 4 we describe how CBR methods and link analysis algorithms can be unified to form one overall ranking method in a hyperlinked environment. These algorithms are more realistic for use with modern search engines although there are still remaining problems to be resolved. The way these algorithms are formulated is mostly by assigning *weights* in the edges of the web graph which makes link analysis algorithms to take into account the importance (e.g., relevance to query, position of hyperlink in the document) of an edge-hyperlink for a specific query.

The task of evaluating all these rank methods is not an easy one. We developed a prototype retrieval system for text and image documents which we

[4]backlinks are hyperlinks from other web sites that point to a specific web site

used to index a small subset of the web and then perform queries on this index. This prototype is described throughout Chapter 5. In Chapter 6 we present the evaluation we made using the prototype described in Chapter 5. Precision and recall measurements are used for the evaluation and comparison of the various ranking approaches implemented. The description of the behavior of each method and possible explanations for their behavior are also discussed.

In Chapter 7 we close this document by citing some ideas for future development. We hope that the content of the last chapter can be form some kind of seed for further research in the domain of information retrieval.

The work done during the elaboration of this thesis is briefly described in the following:

1. Using Larbin[5] we assembled a local collection of web pages and images. We wanted the collection to be more focused on certain topics, thus we developed a wrapper for Google Images[16] service to get a set of starting points for the crawler, relevant to the topics of our desire.

2. We developed the indexing mechanism in order to provide a fast access method for information used by several image and document retrieval approaches. The part of the indexing mechanism that extracts image related text from pages, was contributed by Klavdios Kontis as a part of his thesis titled ”Relevance Feedback Methods for Web Image Retrieval”. Another part of the same mechanism, the visual features extraction tool, was contributed by Vasilios Chatzidiakos as a part of his thesis titled ”Automatic Logo Extraction from Large Web Sites”.

3. Several information processing algorithms were developed or used when available:

   (a) Vector Space Model[6] for text document retrieval

   (b) ChainNet Model for image retrieval, contributed by Klavdios Kontis during his thesis elaboration.

[5]Larbin[12] is an open source web crawler

   (c) Automatic logo detection based on visual features, contributed by Vasilios Chatzidiakos as part of his thesis.

   (d) Image similarity based on visual features extracted.

   (e) HITS algorithm for finding authority and hub web pages.

   (f) PicASHOW algorithm for finding image authorities and image hubs (a specialization of the HITS algorithm)

   (g) Weighted versions of HITS, and PicASHOW algorithms.

4. We developed a simple web interface for efficient use the available retrieval methods.

5. Using the web interface we performed an evaluation on the several image retrieval approaches, based on precision and recall.

Chapter 2

Content Based Retrieval

2.1 Image retrieval using text

Images are commonly found embedded within web pages. The main content of the web pages is text. The text of the web pages is usually related to the content of the displayed (or linked) image either directly or indirectly:

Directly Text surrounding the image usually provides a direct description of it's content.

Indirectly The text of the page as a whole relates to the same subject with the content of the image.

In this work the ChainNet[10] model is adopted for capturing the direct image description while the vector space model provides the mean for capturing the general description of the image content. Both description types are combined together in order to provide a more powerful method of image content description which is then used to search for relevant images on the web.

2.1.1 Vector Space Model

The vector space model[6] is a widely used information retrieval model for textual information. Within the vector space model each piece of information is

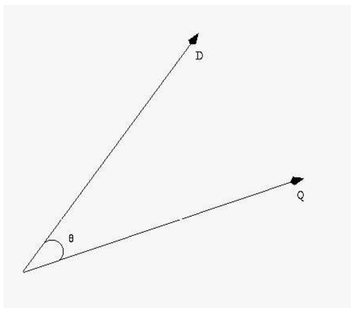


Figure 2.1: Similarity between query Q and document D is expressed as the difference θ angle between the corresponding term vectors.

reduced in a vector of terms. These terms can theoretically be selected from a controlled vocabulary, but in practice they are extracted from the text itself. Usually these terms are also reduced to their morphological roots, mostly known as word stems, using a set of rules and procedures[1]. Also, implementations of this model use a stop word list in order to keep common words like *the, of, for,* ..., out of the term vectors, since they are not providing any information about the content in which they are found.

2.1.1.1 Definition

Retrieving relevant documents using the vector space model, involves the construction of a term vector for the query as well as for each document in the collection[2], and then ordering the documents by similarity to the query. This similarity is often expressed as the difference θ between the query and a document vectors as shown in figure 2.1.

This approach is usually stated as the cosine rule :

$$cosine(Q, D_d) = \frac{1}{W_d W_q} \sum_{t \in Q \cap D_d} (1 + log_e f_{d,t}) \cdot log_e(1 + \frac{N}{f_t})$$

where :

[1]one model for stemming (suffix stripping) commonly used is Porter Stemmer described in [8]

[2]or only for the documents which contain all or at least one of the query terms. The use of an inverted index is very widely used for finding such a subset.

Variable	Description
Q	The vector representing the query
D_d	The vector representing the document, d
W_d	The weight of document, d
W_q	The weight of document, q
t	The current term
$f_{d,t}$	The frequency of term, t, within the document, d
f_t	The number of documents that contain the term, t
N	The number of documents within the collection

2.1.1.2 Term Weighting

The vector space model defines two weights that are associated with each term found in a collection of text pieces of information. The first weight is called *term frequency* or simply *tf* and is different for each document. The other weight is called *inverse document frequency* or simply *idf* and is specific for a whole document collection.

The *term frequency tf* of a term in a specific document is defined as :

$$tf = \frac{log_e(t+1)}{log_e(l)}$$

In the above equation, t is the number of occurrences of the term in the document, and l is the length of the document in terms[3].

Correspondingly, the *inverse document frequency idf* of a term appearing at least in a document collection is defined as :

$$idf = log_e(\frac{N}{n})$$

In the above equation, N is the number of documents in the collection, and n is the number of documents which contain the specific term.

[3]that means that l is the sum of every t for each term appearing in the document.

The similarity function between a query and a document, is defined by the vector space model as the sum of the $tf \cdot idf$ product of each term that is contained both in the query and the document or :

$$TFxIDF(Q,D) = \sum_{w \in Q \cap D} \frac{log_e(t+1) \cdot log_e(\frac{N}{n})}{log_e(l)} \Rightarrow$$

$$TFxIDF(Q,D) = \frac{1}{log_e(l)} \cdot \sum_{w \in Q \cap D} log_e(t+1) \cdot log_e(\frac{N}{n})$$

Comparing the above equation with the cosine rule we mentioned in the previous paragraphs, we see that $W_d = \frac{1}{log_e(l)}$ and $W_q = 1($ as it remains constant for a given query).

2.1.2 ChainNet[10]

This section describes a model for finding relevant images in the WWW by exploiting in an efficient manner, the textual information that surrounds images found in web pages. We adopt an extension of the original model proposed in [10], that is called *Weighted ChainNet* model and is based on the concept of the *lexical chain.* A lexical chain (LC) is a sequence of semantically related words in a piece of text. But before we describe how the model works we should state the type of information on which the model is applied.

Searching for images that are embedded in HTML pages demands that we must have identified those portions of text in these pages that are well related to their embedded images. According to [10], these portions of text are:

- **Image title.** This is the image filename which usually related to the content of the image.
- **Image ALT (alternate text).** ALT text is found at *img* HTML tags and is a piece of text that will be shown to the user in place of a specific image, in case the image cannot be displayed on the user's browser.

- **Image caption.** The image caption is usually one or more sentences of text describing the image. This piece of text is usually the most relevant one, but it's also harder to extract from HTML pages, since there is no HTML standard about an image's caption. In this work the caption consists of text up to a maximum 30 words before and after the image link (or *img* tag).

- **Page Title.** Page title is a short sentence that describes the content of a page. It's not directly related to the image but indirectly since images are often used for enhancing the page's content.

Now that we stated the piece of information that the model will try to exploit, we are going to make a short description of it. As we stated before, the ChainNet model is based on the concept of the lexical chain. A lexical chain in this case is defined as a sentence that carries certain semantics. The ChainNet model is built by six (6) types of lexical chains. The first type is called *Title Lexical Chain (TLC)* and is representing the image title. The second is called *Alt Lexical Chain (ALC)* and represents the alternate text for an image. The third one is called *Page Lexical Chain (PLC),* and represents the page title of the page that an image is embedded. The other three chains are used in order to represent the caption of an image. *Sentence Lexical Chain (SLC)* captures the semantics of a single sentence in the image caption. *Reconstructed Sentence Lexical Chain (RSLC)* captures related sentences semantics, while *Caption Lexical Chain (CLC)* keeps the image's overall semantics. Also a query in the model is represented also by a lexical chain, the *Query Lexical Chain (QLC).*

Now, the similarity between a query and an image in the database, is computed as :

$$imageSimilarity = S(TLC, QLC) + S(ALC, QLC) + S(PLC, QLC) +$$

$$+ \sum_{i=1}^{SLC\,number} S(SLC_i, QLC) + \sum_{i=1}^{RSLC\,number} S(RSLC_i, QLC) + S(CLC, QLC)$$

where S is the similarity between two lexical chains. The similarity between two lexical chains is defined as :

$$Similarity_{list1,list2} = \frac{\sum_{i=0}^{list1.size} \sum_{j=0}^{list2.size} e_i * e_j}{\sqrt{list1.size} * \sqrt{list2.size}} * MatchScale$$

where e_i,e_j are matched words in list1 and list2 respectively. MatchScale is defined as the closeness of two lists from the view of match order. The definition of MatchScale follows :

$$MatchScale_{v1,v2} = \frac{v1 \cdot v2}{||v1|| * ||v2||}$$

where v1,v2 represent the child LC of the first and second original LC's respectively. Also the dot product between two lexical chains is redefined as the following formula:

$$v1 \cdot v2 = \sum_{i=1}^{v1.size} v1_i * v2_j$$

where $v2_j$ is the matched word in $v2$ for $v1_i$ in v1.

2.2 Image Content Description

In the previous section we described a model that exploits textual information surrounding images in web pages. In this section we describe how the content of the image can serve as the means of determining how relevant an image is to a query. The query in this case cannot does not only addresses the text part of a page, but also the content of the image itself. This method can be used when a user decides that he wants to be more specific about a query, and wants to supply as a supplementary information an example image. Moreover image content can also be used to answer keyword queries when query processing involves user's feedback as in the following scenario:

1. The user submits a text query in a search engine.
2. The user evaluates search results and re-submits the query.
3. The search engine (through relevance feedback mechanisms) takes into account the evaluation of the previous step, in order to rank more efficiently using also the image features of the evaluated results.

Another use case could be when the user supplies a query with only image content. We cannot say that this is a very popular for an WWW search engine though, because it is simple enough for the user to describe with a few keywords what he wants to find. Also a good precision CBR method for images, tends to be a lot more complex and resource hungry, which does not fit in the character that internet search engines want to work. This does not mean of course that pure CBR methods for images are not useful in other more specific areas.

The theory we describe in the next sections derive from the thesis of a fellow student mr. Basilios Chatzidiakos, with title "Automatic Logo Extraction From Corporate Web Sites".

2.2.1 Visual Features

The methods described in this section were developed mainly as a part of another thesis with title *"Automatic Logo Extraction from Large Web Sites"* by Vasilios Chatzidiakos. They consist of mainly two portions. The first portion focuses on automatic classification of images as logo/trademark or non-logo/non-trademark images, while the second focuses on defining a distance measure between logo/trademark images. Both of these portions exploit only image content related information. This image content information is based on a specific set of image features. This set consists of the following :

- Intensity spectrum
- Color spectrum

- Frequency spectrum
- Invariant Moments
- Size of the image

2.2.2 Automatic Logo Detection

The concept behind the use of the above set of features for automatic logo recognition is consisted from the following observations:

- logos are usually small graphic images and,
- they are rich in frequency content, with not too many intensity levels of gray or color.

The problem in this approach is to determine the appropriate thresholds (or combination of thresholds) on the above features. The solution to this problem is accomplished by using machine learning decision trees in conjunction with the Otsu bi-level threshold selection algorithm. Then the logo recognition module is constructed using the following procedure :

1. A training set is obtained from an image collection (such as an internet image search engine).
2. Each image in the set is classified by a person as logo or non-logo.
3. The set of features is extracted for each image in the training set
4. From the manual classification the appropriate input (features + classification in appropriate format) is constructed and passed in the machine learning algorithm in order to build a so-called decision tree.
5. Once the decision tree is build, we can use it to automatically classify images, by extracting their features and passing them to the machine learning component.

The precision obtained from the procedure described above depends mainly on the size of the training set and the correct classification.

2.2.3 Image Similarity

The problem of identifying similar images (i.e., images displaying the similar content) , is treated as a problem of feature similarity. The distance for one-dimensional spectrums are computed using the normalized histogram intersection value, while for feature vectors the distance can be defined as the Euclidean distance. After defining the distance functions between individual image features, it's mandatory that an overall distance or similarity measure be defined. This general similarity function may consist from appropriate thresholds (or combination of thresholds) on the distances between features of images. For this task to be completed , the use machine learning with decision trees seems very promising. The procedure to building such a decision tree is similar to the procedure used in the previous section, for automatic logo recognition. The procedure consists of the following:

1. A training set is obtained from an image collection, which should be rich in similar images
2. A human person groups the similar images together, in a way that non-similar images are also grouped.
3. The set of features is extracted for each image in the training set, and the distances between image features are computed.
4. From the distances computed in the previous step, and the evaluation in step 2, an appropriate input is constructed which consists of several feature distances, along with the evaluated decision if the images are similar or not. This input is then used by the machine learning component which builds a decision tree which will decide if two images are similar or not.

This process defines image similarity function to be related closely to the size of training set.

An alternative approach implemented in this work, is to view the distinct feature distances (normalized histogram intersection for 1-D features, Euclidean distance for others) as a vector of distances, and then define the overall distance as any distance measure between vectors. This approach though, ignores the fact that some features are more related to the query content while others are not, and due to this fact we expect such an approach cannot achieve a good precision when used as a similarity function between images.

Chapter 3

Link Analysis

3.1 PageRank

PageRank[7] was proposed by Brin and Page as a measure that defines how *important* a page is in the WWW. It's used in Google[www.google.com] by it's search engine rank algorithm. PageRank serves as an election system between pages. Each hyperlink is considered as a vote from the page containing it to the pointed page. Also each page is considered to have exactly one(1) vote, which it distributed among the pages it points to. Thus, when a page has 2 outgoing links then according to PageRank , the page gives half a vote to each one of the two pages it points to.

PageRank is computed using the following formula:

$$r_i = \sum_{j \in B_i} \frac{r_j}{N_j}$$

where

Variable	Description
r_i	PageRank of page i
r_j	Is the initial ranking of page j which is set to 1
B_i	The set of pages that point to page i (backlinks of i)
N_j	Number of forward links of page j

Equivalently, PageRank can be computed as:

$$PageRank[p] = r_1[p]$$

$$r_{i+1} = A^T r_i \, , \, r_0 = [\,1\,1\,...\,1\,]$$

with $A[i][j] = 1/N_j$ when page i points to page j and zero otherwise, and when pages are given an index number varying from 0 to N-1 where N is the number of pages in the WWW. This means that PageRank values correspond to the eigenvalues of the principal eigenvector of A^T, which can be easily computed using a power iteration algorithm.

The definition of PageRank suffers from some obvious problems. The most obvious is that relatively small communities of pages that link to each other but not often to outside communities receive higher rank, because they distribute their importance only within the same community. To overcome these obvious drawbacks, a parameter d (0 < d < 1) was proposed, which would define how much of it's rank a page distributes among the pages that it points to, while the remaining rank would be distributed equally among all the pages on the WWW. Using this approach the PageRank of a page is defined from the following formula:

$$r_i = d * \sum_{j \in B_i} (\frac{r_j}{N_j}) + \frac{1-d}{m}$$

where:

Variable	Description
r_i	PageRank of page i
r_j	Is the initial ranking of page j which is set to 1
B_i	The set of pages that point to page i (backlinks of i)
N_j	Number of forward links of page j
d	number between zero and one (usually close to 0.85)
m	the number of all pages in the web

3.2 HITS

HITS is a link analysis algorithm that was proposed by J. Kleinberg in [2] as an algorithm that finds authorities and hubs that correspond to a topic t. Authorities are considered to be high quality quality pages for the topic t, while hubs are pages that point to many high quality pages for the topic t. For a page to be considered as a good authority many hubs have to point to it, while for page to be considered as a good hub it must point to many authority pages. The algorithm is described in the following steps:

1. A root set S of pages is obtained from an index such as an inverted file.
2. From S, we construct a new set C, which consists from the pages in the root set S, pages that link to pages in S and pages that are linked by pages in S. C and it's link structure provide a page-to-page adjacency matrix W. This operation can be repeated until S contains a minimum number of pages.
3. Initialize $a_s = 1$,$h_s = 1$,$\forall s \in C$
4. Repeat the following operations until a and h sets converge:
    (a) (I operation) $a_s = \sum_{x\,:\,x\,points\,to\,s} h_x$, $\forall s \in C$
    (b) (O operation) $h_s = \sum_{x\,:\,s\,points\,to\,x} a_x$, $\forall s \in C$
    (c) normalize a and h sets so that $\sum_{x\,:\,x \in C} a_x^2 = 1$, $\sum_{x\,:\,x \in C} h_x^2 = 1$

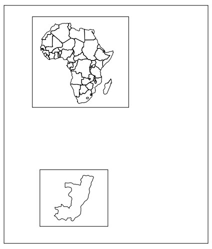


Figure 3.1: Example of two co-contained images.

5. return set a as the set of authorities and h as the set of hubs.

- I operation is equivalent to $a = \mathcal{W}^{\mathcal{T}} \cdot h$ and O operation to $h = \mathcal{W} \cdot a$.
- Kleinberg showed that the resulting a, h sets correspond to the normalized principal eigenvectors of $\mathcal{W}^{\mathcal{T}} \cdot \mathcal{W}$, $\mathcal{W} {\cdot} \mathcal{W}^{\mathcal{T}}$ respectively.
- $\mathcal{W}^{\mathcal{T}} \cdot \mathcal{W}$ is the *co-citation matrix* of the collection. denotes the number of pages which jointly point at pages i and j.
- $\mathcal{W} \cdot \mathcal{W}^{\mathcal{T}}$ is the *bibliographic coupling matrix* of the collection. denotes the number of pages jointly referred by pages i and j.

An interesting concept on these link analysis algorithms is their stability, which questions their ability to give stable rankings under small changes to the hyperlink structure of the web, addressed in [17].

3.3 PicASHOW

PicASHOW[3] uses link analysis for searching for high quality images. The reasoning behind PicASHOW's approach which is also described in [3] is:

- Images which are co-contained in pages are likely to be related to the same topic (example shown in figure 3.1).

- Images which are contained in pages that are co-cited by a certain page are likely related to the same topic (example shown in figure i3.2).

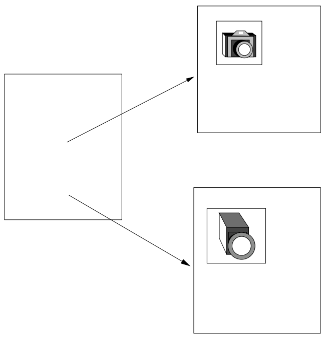


Figure 3.2: Example of two co-cited images.

Definition: A page p is said to contain an image i in either of the two following cases :

1. p displays i (*img* tag)
2. p points to i's file (*a* tag)

The algorithm which computes the authority and hub scores is like the original HITS algorithm but it is applied in a page-to-image adjacency matrix. Assuming that we have a set $\mathcal{P}$ of relevant Web pages and $\mathcal{W}$ is the respective page-to-page adjacency matrix, $\mathcal{M}$ the page-to-image adjacency matrix and $\mathcal{I}_{|\mathcal{P}|}$ the $|\mathcal{P}| \times |\mathcal{P}|$ identity matrix, PicASHOW introduces three schemes of the page-to-image adjacency matrix:

1. The page-to-image adjacency matrix used is the original page-to-image adjacency matrix $\mathcal{M}$. PicASHOW notes that this approach fails to relate images that appear in pages that are co-cited, which should state that these images are related (because we consider the pages that contain them to be related).
2. The page-to-image adjacency matrix used is $\mathcal{W} \cdot \mathcal{M}$. This approach addresses the problem that arises when using the matrix $\mathcal{M}$, by associating each page p with images that are contained in pages to which p links. However approach as noted in [3] fails to associate images with the pages that contain them.

3. The page-to-image adjacency matrix used is $(\mathcal{W}+\mathcal{I}_{|\mathcal{P}|})\cdot\mathcal{M}$. This approach defines each page p to be related both to the images that are contained in p and images that are contained in pages to which p links to.

In this work we adopt the definition no. 3 of the page-to-image adjacency matrix in searching for high quality images of a subject, because it seems to provide the most fair relation between images and pages.

Chapter 4

Combination of CBR and Link Analysis

4.1 Weighted HITS

Link analysis by itself is blind to the actual content of the documents it ranks, and there are also some issues produced by the algorithm itself. A weighted version of HITS can be obtained by modifying operations (a),(b) in step 4 of the original HITS algorithm in section 3.2, as follows :

(a) $\quad a_s = w_{a_s} \cdot \sum_{x\,:\,x\,points\,to\,s} h_x,\ \forall s \in C$

(b) $\quad h_s = w_{h_s} \cdot \sum_{x\,:\,s\,points\,to\,x} a_x,\ \forall s \in C$

where w_{a_s},w_{h_s} is the similarity value (e.g., textual similarity based on the vector space model) between the document s and the query. This weighted scheme was proposed in [1] in order to reduce the influence of less relevant documents on the scores of their neighbors.

The above equations can also be rewritten as:

(a) $\quad a_s = \sum_{x\,:\,x\,points\,to\,s}(w_{h_{x,s}} \cdot h_x),\ \forall s \in C$

(b) $\quad h_s = \sum_{x\,:\,s\,points\,to\,x}(w_{a_{x,s}} \cdot a_x),\ \forall s \in C$

Weight $w_{h_{x,s}}$ is a number describing how much should we consider document x as a good hub for document s, while weight $w_{a_{x,s}}$ describes how much should we consider document x to be a good authority for document s. This approach allows us to use the same formula for more sophisticated weighting schemes, which we will describe in the following paragraphs.

An important situation that is addressed at [1] where the original HITS fails is resolved with specifying these weights. Sometimes a set of documents on one host point to a single document on a second host, and sometimes a document on one host points to a set of documents on a second host. Since such situations happen when the documents are authored by the same person (or organization), we can state that mutually reinforcing relationships between hosts give undue weight to the opinion of a single person. This issue is resolved in [1] by setting $w_{h_{x,s}}$ to $1/k_{x,s}$ where $k_{x,s}$ is the number of links from documents hosted at x's host which point to page s, while $w_{a_{x,s}}$ is set to $1/l_{x,s}$ where $l_{x,s}$ is the number of links from page s to pages hosted at x's host.

We can also recognize that we can set $w_{a_{x,s}}$ to a relevance weight between page x and the query. We could also set $w_{h_{x,s}}$ to the same relevance weight because if x is a good hub for topic t then we can also agree that it will have some relevance with topic t. Taking this in consideration, as well as the previously stated solution for the problem of mutually reinforcing relationships between hosts, we can rewrite the equations $(a), (b)$ as follows:

(a) $\quad a_s = \sum_{x\,:\,x\,points\,to\,s}(\frac{s_x}{k_{x,s}} \cdot h_x),\ \forall s \in C$

(b) $\quad h_s = \sum_{x\,:\,s\,points\,to\,x}(\frac{s_x}{l_{x,s}} \cdot a_x),\ \forall s \in C$

where

Symbol	Description
a_s	authority score of page s
h_s	hub score of page s
s_x	relevance weight between x and query
$k_{x,s}$	number of links from pages on x's host to page s
$l_{x,s}$	number of links from page s to pages on x's host

Another use of content similarity methods in conjunction with HITS, is for pruning non-relevant nodes from the focused subgraph of the Web. When non-relevant nodes exist in the focused subgraph (which is the result of the back and forth expansion of a set of relevant pages), and these nodes are strongly connected, then the top authority and hub pages tend to be irrelevant to the topic. For example, the query "*car and BMW*" results in top authorities being pages of different car manufacturers while the best hubs are lists of car manufacturers. Possible solutions to this issue are also presented in [1] using several approaches on thresholding the relevance weight, however we cannot state that any of these approaches is optimal.

4.2 Weighted PicASHOW

In the previous section we described the weighted version of HITS. The same approach is used with PicASHOW in order achieve better precision in ranking authorities and hubs. We only experiment with $(\mathcal{W}+\mathcal{I}_{|\mathcal{P}|})\cdot\mathcal{M}$ (review section 3.3) as the page-to-image adjacency matrix because it seems as the most efficient method to compute image authorities and image hubs. In this situation we have to assign weights at the edges of $\mathcal{W}$, $\mathcal{M}$ and $\mathcal{I}_{|\mathcal{P}|}$.

- In matrix $\mathcal{W}$ we can assign it's edges to be a relevance score between query and anchor text of the link the edge denotes.
- The edges of matrix $\mathcal{M}$ can be an image feature similarity between the image and an image example the user gives as part of the query, and-or relevance between image related text (alternate text, caption) and query text.
- $\mathcal{I}_{|\mathcal{P}|}$ identity matrix can be left without weights or assigned the relevance score between the page and the query (text based similarity).

Considering the theory described in earlier sections the above can become more realistic as follows:

$\mathcal{W}_{i,j} = R_{q,a(i,j)}$

$\mathcal{M}_{i,j} = RP_{q,i} * \frac{(RC_{q,j}+RI_{q,j})}{2}$

$\mathcal{I}_{i,j} = (i == j?1 : 0)$

where

Variable	Description
$R_{q,a(i,j)}$	$TF \cdot IDF$ relevance score between query and anchor text of edge i->j
$RP_{q,i}$	$TF \cdot IDF$ relevance score between query and document's i text
$RC_{q,j}$	ChainNet relevance score between query and image j's related text
$RI_{q,j}$	similarity between q's and j's feature vectors (normalized (1-Euclidean distance))

The above weights are chosen assuming that the query is consisted of both keywords and an example image. If the example image is missing then $RI_{q,j} = 1$, or if the keywords are missing then $R_{q,a(i,j)} = RP_{q,i} = RC_{q,i} = 1$. This way the above system, in order to define the weighted relations between pages and pages with images, takes into account only those types of features that are supplied with the query.

Chapter 5

System Implementation

5.1 Implementation Overview

In this chapter we describe a prototype system which for finding images (focusing on logo images) on the WWW. The system itself is consisted of mainly three parts. The crawling part, the indexing part and the retrieval part. These three parts implement a task. The crawler's task is to gather locally a set of pages among with their embedded images. The indexing's part is to index the information contained in the pages and images set, so that access to certain information is fast enough. The retrieval part uses the database that is produced from the indexing part and the algorithms that are implemented, in order to answer queries submitted from the use of a web interface. Figure 5.1 presents graphically the components and how they interact with each other, of a web search engine.

Due to the complexity of a search engine system, we made the following decisions regarding the development of our retrieval system:

- ◦ Limited to specific topics dataset.
- ◦ Perl as the main programming language.
- ◦ Embedded database (less features for more speed and simplicity).

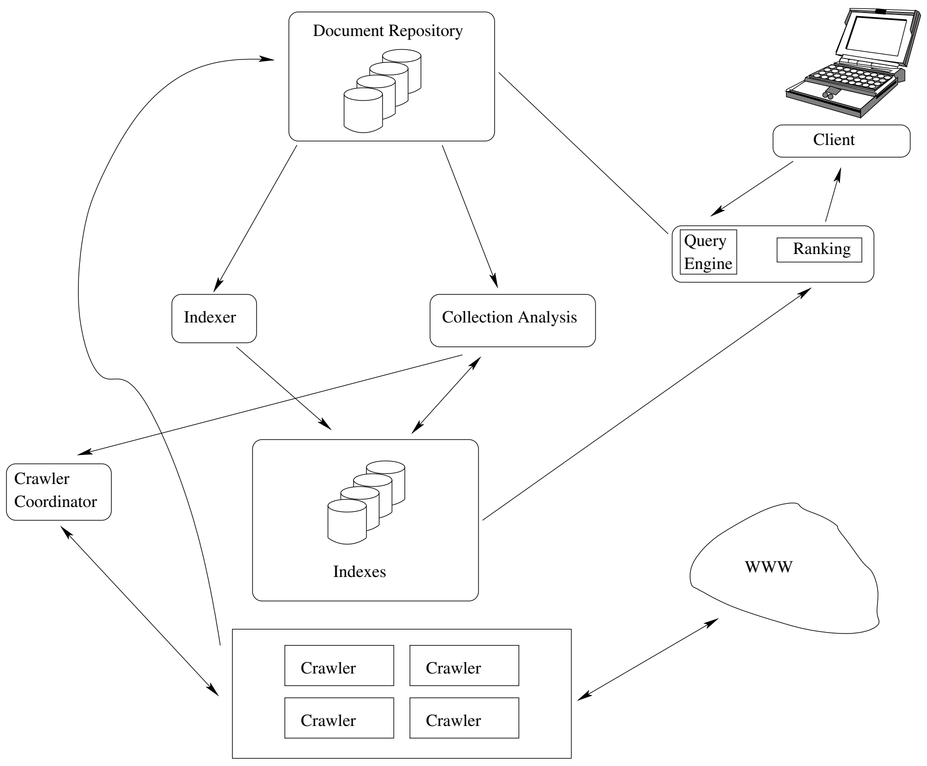


Figure 5.1: Schematic view of the components of a web search engine

These decisions where mainly made due to the limited computing power that we were able to use. The computing power that we had available was that of a relatively modern personal computer. The fact that we had to process a lot of images in order to extract their features made the computing power of our available hardware look even less.

The reason we chose Perl as the main programming language was the time in which one person should deal with the complexity of the system. Perl is high level programming language which is optimized for text processing, but has also been used in a variety of other projects. Perl's most important features are :

- Scripting-like syntax
- Builtin regular expressions
- Object Oriented
- Huge library of custom modules

Perl's biggest disadvantage is that it's very slow for computations, which is nearly a hundred times slower than native. The use of an embedded database was decided in order to minimize overhead , and because of the fact that Perl can handle in simple manner such types of databases through tied access to builtin hash variables.

The system that was implemented provides the following functionality in it's final form:

- Retrieval of web pages.
- Retrieval of images.
- Retrieval of connectivity information (links and backlinks for a WWW document).

The retrieval of web pages and images may be done using several ranking schemes, the most complex of which are the weighted HITS, and weighted PicASHOW ranking methods.

5.2 Crawling

Generally a crawler's job is to fetch documents from the world wide web. In production systems it always works as a distributed system and sometimes it uses link analysis algorithms in order to fetch more important pages before the less important. While it fetches documents it must also consider the fact that it must consume the available bandwidth in a polite fashion, so that it does not over-consume resources (bandwidth, cpu power, disk usage of web servers). This behavior is defined by the robots exclusion protocol which consists of mainly two rules:

- a robot must respect the robots.txt rule file located at the root of the web server's hostname (e.g : http://www.microsoft.com/robots.txt).
- a robot must wait at least 60 seconds between requests on the same host.

The crawler can also have a more complicated policy for which pages to fetch first, such as examining the anchor text in links to fetch more relevant to a certain topic pages. In our system we want to emulate such a behavior in order to create an appropriate dataset, that consists of several pages along with their embedded images, that pertain to a set of topics. The created dataset will be used as input to the indexing part of our system, and defines what type of queries can be answered. In contrast to the other two parts of our system, this task is fulfilled using mainly a freely available software that implements an http crawler. The software we used is called Larbin[12] and implements a very fast crawler with the basic functionality that a crawler should have. However focused crawling is not part of such functionality and it wasn't either part of Larbin. In order to emulate this behavior we decided that we would create a set of queries-topics and use the results of a search engine as the input to the crawler. In other words the crawler will have as the origin points from where the crawler will start it's recursive crawl of the web. We also need to define the recursive behavior of the crawler, in order to define some thresholds, that is what to fetch. The default behavior of the crawler we use is to fetch pages in a pure breadth-first walk. Another mode of this crawler is a breadth-first walk with a threshold in how many links away from a page in the origin points set. The last behavior is what we used in order to make the crawler download only those pages which are *close* to the pages in the root set. If we used the default behavior, then very quickly the crawler would reach a state in which it would download a lot of pages irrelevant (*distant*) to the pages of the root set. The task of creating the root set is also done in an automatic manner. We developed in a small perl script a wrapper for the google image search engine, which takes a query a it's argument and using the google search engine returns the pages that contain the images-results that google returns. That way using twenty (20) queries as the topics set, we obtain a root set consisting of more than fourteen thousand (14000) pages. This root set when used as input in the crawler can lead to a crawl that theoretically can lead to several 10ths of millions when we crawl up to 5 links in depth from the documents in the root set(assuming that a web page has on average 10 outgoing links then there are $14000 * 10^5$ documents that can be retrieved) , that pertain to a set of topics. Summing up the crawling part of our system is accomplished through the following method:

1. write some queries for a topic of interest
2. use the google images wrapper with input the set of queries in the previous step, in order to obtain pages with relevant images on certain topics
3. use the output from the wrapper in the previous step as an input to Larbin web crawler
4. wait a day or more for the crawler to fetch a set of documents (pages, embedded images) that we are able to process-index in a sane amount of time

Using the procedure described above with a maximum depth of 5 links from the root set, we obtained around 3.000.000 documents (pages and images), after we waited approximately two days for the crawler. The Internet line of our university allowed internet speeds at most 400Kbytes/second. The queries we used concerned mostly logo images (e.g., ”suse logo”).

5.3 Database - Indexing

The database of an Internet search engine has to provide a storage place and a fast access method for the pieces information that are necessary for the retrieval module to work. The pieces of information such a database needs to store are:

- the urls of the documents fetched by the crawler
- the content of these documents itself
- content meta data (mime_type, last-modified date etc...)
- title for documents that provide one
- word positions for the text document
- hyperlinks of documents that support them
- static data as stop-words

A database is usually described with an *entity relationship* diagram. Figure 5.2 shows what an entity relationship diagram for an image retrieval system looks like.

It describes the basic entities found in the information pieces, their attributes and how these entities are related (in the data level) with each other (even with themselves). There are rules according to which from a given entity relationship model (ER) we design the relational model of the database the E-R describes. We don’t use a relational database but an embedded one, Berkeley DB[15] specifically, which is also a built-in module in the Perl programming

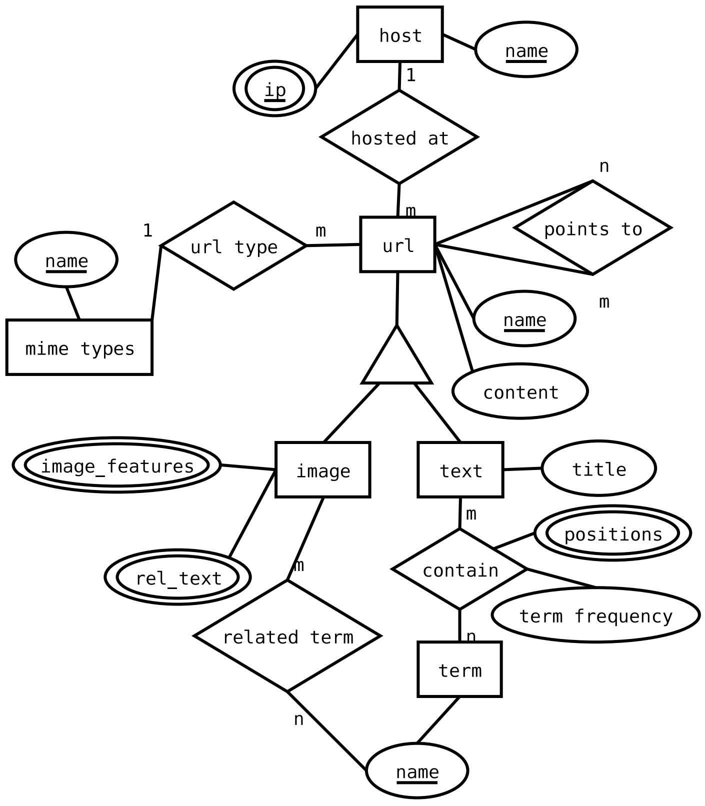


Figure 5.2: A simple Entity-Relationship diagram for an image retrieval system on the WWW.

language, however, the same rules apply in order to design the structure of the embedded database.

Berkeley DB defines in it's documentation that it is an open source embedded database library that provides scalable, high performance, transaction-protected data management services to applications. Berkeley DB provides a simple function-call API[1] for data access and management. It supports hash tables, B-trees, simple record-number-based storage, and persistent queues. We could say that Berkeley DB is a database system for use with systems that would not be benefited by the use a relational database. Indeed relations such the text->contains->term are implemented by search engines with an inverted index, which is an index of pairs of the form <key,value> where the key is a word (stemmed) and the value is a chain of keys to documents that contain the specified attribute. The only disadvantage with the inverted index approach, is that it's structure is usually not that flexible (dynamic updates require more complicated structures). In our implementation the database is considered static, once the indexing process ends it's run. This approach was used in order to keep the database structure simple, so that it's access is done in a lite manner.

The transformation of a relational model (which corresponds to an entity relationship diagram as the one we presented previously) into a database structure using Berkeley DB results in a set of individual user files, each of one providing a fast access method for storing and retrieving pairs of keys and values, by key value. This allows us to reduce the tables of the equivalent relational model, to individual db files, with additional splitting when there is need for indexed access to non-key attributes - the key attributes, and inverted index usage when the relation is many-to-many. An inverted index is an index whose values are sequences of keys to related with the key information. The structure of an inverted index is graphically displayed at figure 5.3.

[1]Application Programming Interface.

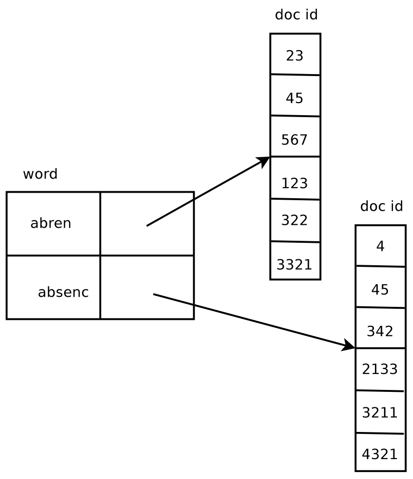


Figure 5.3: Structure of inverted index

The advantage of using inverted index (also known simply as inverted files) is the fast access they provide for retrieval. The disadvantages are it's storage requirements, and the fact that it's structure does not easily allow dynamic updates and deletions (in [11] is described a version of inverted files allowing dynamic maintance in an efficient manner).

Using the methodology previously explained, the database results in a file structure consisting of more than twenty (20) files-indexes, with four of them being inverted indexes. The two of these inverted indexes provide a fast access method for the link and backlink information, while the other two provide the same for pages and images related to stemmed words. All the other files provide indexed access for necessary information, such as term frequencies and inverse document frequencies all the words appearing in the documents being indexed.

The indexing process needs to extract four(4) types of information:

1. Term frequencies. These are extracted from the rendered (what the user actually sees) html text, while ignoring commonly used words and suffixes. These frequencies are normalized according to the document length, in terms (as described in section 2.1.1).

2. Hyperlinks. These are extracted from the html structure, using the corre-

sponding attributes in *a,img* and *frame* tags.

3. Image related text. Since we will be implementing the ChainNet image similarity model described in section 2.1.2 of this document, this set is consisted of the filename,alternate text and caption of the embedded image. The filename and the alternate text are extracted directly from the corresponding *img* tag. The caption of the image is defined as a sentence or small paragraph following the image. Often there is no caption in an image, but the related text is found just after or just before the image. As a result of this behavior, the extraction of image caption is not well defined, so we defined it using the following rules:

   (a) Image is inside a table cell. In this case the caption of the image is considered to be the text within the table cell.

   (b) Image is found between two paragraphs or sentences (*p* tag). In this case the caption of the image is considered to be the text contained in the surrounding paragraphs.

   (c) To prevent very big (in terms) captions to occur, we define a threshold of 30 terms maximum before and after the image tag.

4. Image features. From section 2.2 of this document we find out that the needed features are:

   (a) unique number of colors

   (b) image dimensions (width and height)

   (c) image integral in Fourier plane (circle from the origin)

   (d) invariant moments

   (e) intensity, hue, saturation histograms

   (f) histogram in Fourier plane (integral of 128 rings)

   (g) Otsu bi-level thresholds for intensity,hue,saturation and Fourier histograms

For the extraction of the third and fourth types of information we used software that was developed during the elaboration of other peoples thesis. Visual features extraction was provided by Vasilios Chatzidiakos, while image related text extraction was provided by Klaudios Kontis. Those tools were available in the form of CLUI[2] programs, which are used as they are from our indexing software. The first and second types of information are extracted from pages thanks to a perl module we developed, based on the HTML::Parser html parser. In contrast with other popular html parsing frameworks, the HTML::Parser implements a parser designed to work with HTML that is actually "out there", and it normally parses as close as possible to the way the popular web browsers do, instead of strictly following one of the many HTML specifications from W3C. The stop list our indexer uses for ignoring common words, is the one available with the S.M.A.R.T information retrieval software, and is consisted from about 572 stop words. The stemming of words is accomplished thanks to a cpan[www.cpan.org] available perl module called Lingua::Stem which currently implements stemming algorithms for more than 8 languages.

Our first approach on the indexer development, was to implement it as a set of perl modules using a pure object oriented approach. The result was a simple interface, which on the high level, was consisted of objects implementing an indexing and retrieval engine. This approach ended up in a useful API which could be used very simply to implement applications with text retrieval and relevance ranking capabilities. However the result proved not to scale well when the size of the collection exceeded a certain amount of documents. The reason for this was the absence of a fast loading mechanism for the database, and when the size of the index exceeded some percentage of the computer's physical memory, the rising frequency of disk usage made the indexing process too slow. Still the search capabilities of this approach are enough for implementing a search engine for a site, even a small intranet (about 50.000 pages).

[2]CLUI = Command Line User Interface

The second approach (which was the one we finally used), was to implement the indexing process on a per-index implementation. This approach resulted in an indexing system that was easy to fine tune, as each index was created separately and it's creation was optimized to efficiently use the available memory. Each indexing module used a cache, which when filled it's contents were sorted and dumped in the database. The result of this approach was an indexing method that scaled very well to bigger collections. Each indexing module is a perl program, and it's task is to build one or more indexes. Each module also may depend on other indexes, and these dependencies are solved automatically with using the *GNU make* utility. The dependencies are written in a file, which the *make* program utilizes to execute in an appropriate order the necessary commands to build the database. With this approach we easily managed to index about 200.000 documents (images and pages) per day, with visual feature extraction being the bottleneck in the whole indexing process.

5.4 Retrieval - Ranking

The information that we can use for retrieval consists of text, hyperlink structure and image features. This information allows our system to be able to answer queries consisting of:

1. one or more keywords
2. one or more keywords and an example image
3. just an example image

We implemented (except those that were contributed from other peoples) the Vector Space Model, ChainNet Model, Visual Features Similarity, HITS, Weighted HITS, PicASHOW, Weighted PicASHOW methods that were described at sections 2,3.

In the following subsections, we describe the procedure that our retrieval method (Weighted PicASHOW) follows, in order to answer queries with images as the results.

5.4.1 Retrieval by keyword(s)

This is the most commonly used scenario. The procedure that our system follows to answer this type of queries is described as a series of steps:

1. From the inverted index that associates images with their related stems, we get an initial image set **s** which contains image identifiers with images that are somehow relevant to the stems of the query keywords. Also images with a low logo probability, are ignored, since our system focuses on logos.

2. From the set **s** we construct two adjacency matrices : the **page-to-page** adjacency matrix and the **page-to-image** adjacency matrix.

   The construction is done in the following way :

   (a) If the set **s** is very big we reduce according to a parameter of the system. When we want to reduce the size of **s** , we purge images with the lowest chainnet score[10, 5], until the size of **s** reaches the desired size.

   (b) From **s** a set **s'** is obtained which contains all the image identifiers in **s** as well as the document identifiers which point to images contained in **s**.

   (c) From **s'** a set **s"** is obtained which contains every document/image identifier in **s'**, the document identifiers of documents that point to documents in **s'** and at most **n** document identifiers that are pointed by documents in **s'**. Here, **n** is also a parameter of the system.

   (d) Now from the inverted index which associates document identifiers with the document/image identifiers to which they point and the set **s"**, we construct the **page-to-page** adjacency matrix ($\mathcal{W}$) and the **page-to-image** adjacency matrix ($\mathcal{M}$). Matrix $\mathcal{W}$ is constructed so that $\mathcal{W}_{i,j} = 1$ when document i has a link to document j, while matrix $\mathcal{M}$ is constructed so that $\mathcal{M}_{i,j} = 1$ when document i displays or links to image j.

3. In this step we assign weights to $\mathcal{W}$,$\mathcal{M}$ edges.

   (a) $\mathcal{W}_{i,j}$ will contain the normalized sum of $tf \cdot idf$[3] between the anchor text of the link that $\mathcal{W}_{i,j}$ describes.

   (b) $\mathcal{M}_{i,j}$ will contain the normalized chainnet [5, 10] score for the image j multiplied by the normalized sum of $tf \cdot idf$ between the document i and the query keywords.

4. We construct the **new page-to-image** adjacency matrix $\mathcal{G} = (\mathcal{W}+\mathcal{I})\cdot\mathcal{M}$ (see [3]).

5. We apply Kleinberg's HITS algorithm to find the principle eigenvectors of $\mathcal{G}^T \cdot \mathcal{G}$ and $\mathcal{G} \cdot \mathcal{G}^T$, matrices. The implementation of the computation needed in this process also applies the extension described in section 4.2 for resolving the mutual reinforcement relationships between hosts when the case is image retrieval , as well as pruning of links between pages in the same host (but not pruning of links to images).

6. Present the results to the user ordered by their authority values according to the principle eigenvector of $\mathcal{G}^T \cdot \mathcal{G}$ which was computed in the previous step.

The above ranking procedure, implements the weighted PicASHOW, as described in section 4.2 of this document, when the query is a set of keywords.

5.4.2 Retrieval by keyword(s) and an example image

In the case where the user supplies a query as a set of keywords and an example image, the procedure remains pretty much the same as the one described in the previous subsection, except the fact that the features of the example image

[3]tf : term frequency, idf : inverse document frequency on the whole collection

are also used during the focused subgraph creation, and for the assignment of weights on the page-to-image adjacency matrix ($\mathcal{M}$). If we watch carefully the procedure described in the previous subsection, we'll find out that only 2(a) and 3(b) steps need to be re-examined. These are also the only steps, that need to utilize any kind of image similarity. The previously described procedure, designed to answer keyword queries, utilizes only the textual information about the concerning images, which is exploited using the chainnet procedure described in section 2.2 of this document. In the case we are currently examining we need to utilize textual and image content information, therefore we replace the chainnet score with an overall score which is defined as a combination of the chainnet score and an image similarity function. The image features similarity function which we use is defined as:

$$imageSimilarity_{q,d} = 1 - \frac{ni_{q,d} + gd_{q,d} + oid_{q,d} + ohd_{q,d} + osd_{q,d} + ofd_{q,d}}{6}$$

where

Symbol	Description
q	the query image
d	candicatecandidate relevant image
$ni_{q,d}$	1 - (normalized histogram intersection between q and d)
$gd_{q,d}$	geometrical distance between q and d
$oid_{q,d}$	Euclidean distance of q,d Otsu thresholds on intensity spectrums
$ohd_{q,d}$	Euclidean distance of q,d Otsu thresholds on hue spectrums
$osd_{q,d}$	Euclidean distance of q,d Otsu thresholds on saturation spectrums
$ofd_{q,d}$	Euclidean distance of q,d Otsu thresholds on Fourier plane histograms

Also the variables taking place in the above formula are first calculated for all the images being compared, and the six(6) resulting distance vectors are normalized according to the Gaussian normalization procedure using $3 * standard\_deviation$ as the normalization parameter, and then we calculate the image similarity as defined using the normalized values of distances. The overall similarity function is defined then as the sum of the normalized (a Gaussian normalization again) chainnet, features score divided by 2. This approach defines that each feature has an equal role in the similarity function, and also defines the overall score to be equally dependent to textual and image features scores. However this equality is not correct, because some features or scores can be more or less precise for relevance ranking. These formulas however, can be altered easily so that appropriate weights can define how important each feature or score is, and these weights can be computed easily through relevance feedback mechanisms.

Thus in the case where the query is consisted of keywords plus an example image, the chainnet score in the procedure described in the previous subsection is replaced with a score provided above, which incorporates textual as well as image similarities. This score can also be used autonomously as a relevance score to provide a **ChainNet model with visual features ranking method**.

5.4.3 Query by Example Image

In order to answer queries consisting of an example image, one should implement a different indexing-clustering scheme than the one used in keyword retrieval methods (inverted file). Such work was beyond the scope of this thesis, and since the dataset size (~200.000 images) allowed full sequential scans to be done in a matter of minutes, we simply implemented a mechanism that answers such queries by computing the visual features relevance score for each image in the collection. More specifically the query is answered the following way:

1. The user uploads an image file.

2. Once uploaded the image file is scanned by the visual features extractor and the visual features are computed.

3. For each image in the dataset if it is classified (from the automatic logo detection module) as a logo, then the visual features similarity score is computed between the query and the image. If the image is classified as a non-logo then it is ignored.

4. The images in the database are sorted according to the similarity score computed earlier, and the n (e.g., 100) most relevant images are kept as the result dataset.

5. The system continues the ranking procedure according to the ranking scheme that will be used. If weighted link analysis is used then each weight that can't be computed (since the query has no textual information) is set to 1.

This query model was implemented to test further the visual features comparison. The use of this approach revealed that the visual features description was pretty good, as the results returned were similar to the query images, if someone does not focus on the objects that the query image contains. However, this method failed in most cases to actually return relevant images.

Chapter 6

Experimental Results

6.1 Evaluation Method

The effectiveness of information ranking schemes is determined by:

- The percentage of query-relevant documents in the results.
- The ability of the system to rank relevant documents higher than non-relevant.

A document is considered relevant when the query is somehow related to it. In our case the query can be a set of keywords, or a set of keywords and an example image, and the results are image documents. When a query is a set of keywords, a result is considered relevant if the keywords can be considered as a short description for it. When the query is a set of keywords and an example image, then a result image is considered to be relevant only when it shares common objects with the example image. In this case the example image is considered to be the more descriptive portion of the query. Using these guidelines a human inspects the answers from several queries, and for each answer he decides if an answer is relevant or not relevant to the query.

The quantities that are computed are

Precision which is, the percentage of relevant images retrieved with respect to the total number of retrieved images

Recall which is, the percentage of relevant images retrieved with respect to the total number of relevant images in the collection. Because of the size of the collection (~200.000 images) it is practically not possible to compare every query with every image in the collection. Thus, we use relative recall instead of recall which means that the answers obtained by all candidate methods are merged and this set is considered to contain the total number of correct answers. This is considered as a valid sampling method, known as *pooling method,* which has been used thoroughly in the text retrieval literature.

Ranking quality which computes the differences between the ranking of the results obtained by a method and by a human referee. Higher *ranking quality* means that the respective ranking method manages to rank relevant documents higher than non-relevant. *Ranking quality* R_{norm} is computed using the following procedure:

1. The answers of the candidate method are evaluated.
2. Each answer is assigned a number which is equal to it's order within the result set.
3. These answers are taken in pairs so that
   (a) Only pairs with one relevant and one non-relevant answer are taken
   (b) In each pair, the relevant answer is the first entry and the non-relevant is the second
4. R_{norm} is computed as follows:

$$R_{norm} = \{ \begin{matrix} \frac{1}{2}(1+\frac{S^{+}-S^{-}}{S^{+}_{max}}) \text{ , } S^{+}_{max}>0 \\ 1 \text{ , } S^{+}_{max}\leq 0 \end{matrix}$$

where

Symbol	Description
S^+	the number of correctly ranked pairs(the relevant entry has higher rank)
S^-	the number of incorrectly ranked pairs
S^+_{max}	is the total number of ranked pairs

Afterwards, a *precision-recall* diagram is presented for each experiment. In this diagram, the horizontal axis corresponds to the measured relative recall, while the vertical axis corresponds to precision. Each curve in the diagram represents exactly one ranking scheme. For each ranking scheme, each query retrieves the top 30 results and each point in the corresponding curve is equal to the average over 20 queries. Precision and recall are computed after each result (1 to 30) therefore each curve has exactly 30 points. The top-left point of a curve corresponds to the presicion,recall values for the top 1 result (best match) while the bottom-right point corresponds to the precision,recall values for the entire result set.

A ranking scheme is considered better than another if it achieves better precision,recall values (it's curve is above the other curves). It is possible that two or more curves cut each other. This means that there are methods that perform better for small result sets, while the others behave better for larger result sets.

Presicion and recall are used by tradition for evaluation of ranking schemes on a pre-labeled collection (such as the TREC[13] collection). Ranking quality instead, is rarely used. It represents the capability of a method to rank relevant results higher than non-relevant.

The collection we used consisted of approximately 400.000 documents with half of them being web pages and the other half being images contained in those web pages. This collection was obtained using the crawling and indexing mechanisms described in sections 5.2,5.3.

6.2 Experiments

The main reason behind this set of experiments is to identify how well a weighted scheme of link analysis compares to unweighted link analysis schemes and content-based ranking schemes. We made two experiments one for each type of query. We remind that the two types of query that our system supports are:

1. a set of keywords

2. a set of keywords plus an example image

The methods we compare when the query is a set of keywords are:

1. ChainNet model described in section 2.1.2

2. PicASHOW described in section 3.3

3. Weighted version of PicASHOW described in section 4.2

Methods 2,3 are provided by the same link analysis mechanism described at section 5.4.1. The difference in the case of unweighted PicASHOW in the ranking mechanism is that the procedure of assigned weights at the edges of $\mathcal{W}, \mathcal{M}$ matrices is replaced by a simpler one that just sets each edge on these matrices to 1,0 when the edge corresponds to a link,non-link respectively.

The methods we compare when the query is a set of keywords plus an example image are:

1. ChainNet model plus use of visual features similarity described in section 5.4.2

2. Weighted version of PicASHOW also described in section 5.4.2

We don't compare PicASHOW it does not support example images in the way we consider. In PicASHOW[3] it is mentioned that the model can be used to support *"find similar images"* queries given an image url, while we do not consider the existence of the example image on the web as necessary.

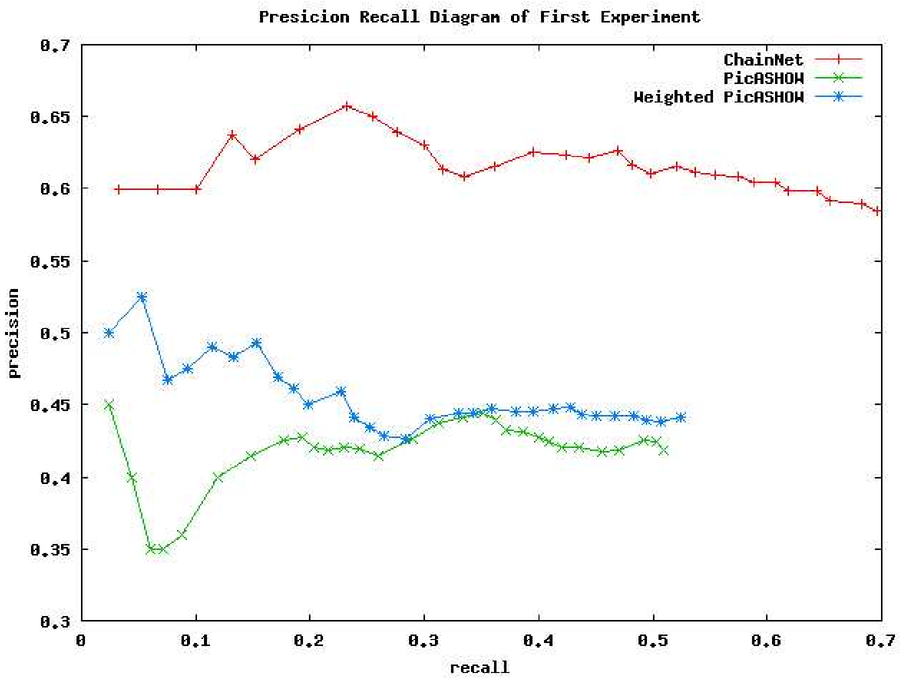


Figure 6.1: Precision-Recall diagram for query-by-keywords methods.

Method	Ranking Quality
ChainNet	0.587055
PicASHOW	0.536222
Weighted PicASHOW	0.597533

Table 6.1: Ranking Quality for query-using-keywords ranking methods.

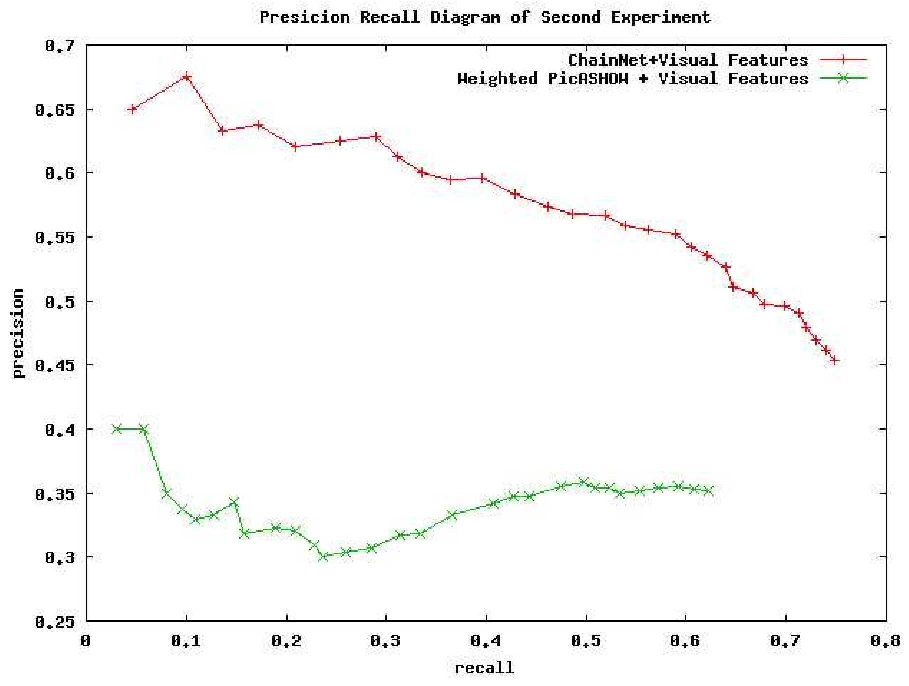


Figure 6.2: Precision-Recall diagram for queries consisted of keywords and an example image.

For the first experiment, the precision-recall diagram is shown at figure 6.1. Table 6.1 shows the corresponding ranking quality for each ranking method.

We can see that the text similarity approach, provided by ChainNet, supersedes the link analysis based approaches. We can also see that Weighted PicASHOW scheme is significantly better than the simple PicASHOW scheme, which is what we expected. Also, ranking quality measures, show that PicASHOW fails (relatively to the others), to present relevant images before non-relevant.

For the second experiment, in which we test two candidate ranking schemes for queries consisting of keywords and an example image, the corresponding precision-recall diagram is shown in figure 6.2. Table 6.2 shows the corresponding ranking quality for each ranking method.

As we can see the link analysis approach fails even more in respect to the content-based approach, to retrieve good precision and recall on the results. Ranking quality measures also confirm this behavior. Comparing the results of

Method	Ranking Quality
Visual ChainNet	0.644514
Visual Weighted PicASHOW	0.508352

Table 6.2: Ranking Quality for query consisted of keywords and an example image ranking methods.

the two experiments we can see that when the queries become more specific (the example image makes the query more specific), then pure content based ranking methods achieve better results, while link analysis based methods provide a poorer (in terms of precision and recall) ranking scheme.

From our observations on the various query results, there were often images totally unrelated to the query topic which were given a very high ranking when using link analysis methods. The characteristics of those images are:

- They are replicated on several pages of different topics. Such images have a better possibility to be near a related to a query term.
- They are logos of companies that collaborate with many other companies (e.g., distributors of software).

The above observations makes us consider that when link analysis is used in the form of PicASHOW, the focused subgraph which is used, although it contains a relatively small percentage of non relevant documents, these documents tend to be very popular (contained in a lot of different pages). Weighted link analysis methods, achieve to lower the rank of such documents through relevance influence, however relevance influence cannot succeed in driving off non relevant nodes.

6.3 Conclusions

While in our experiments, link analysis approaches perform generally as a poorer ranking scheme, it is important to notice that they provide a very useful ranking method for commercial search engines when used in an efficient way. In our experiments the queries we used where specific to logo images. Specific queries is not often the case that an WWW search engine must handle. A lot of users ask questions (e.g., "holidays in Greece" or "cheap cars") which result in a very big in size, set of relevant results. The same situation occurs for specific queries (e.g., "Microsoft products") when the object of the query is very popular. In all these cases link analysis methods achieve better results (e.g., returning "http:///www.debian.org/" for the query "debian" instead of an other page in which the "debian" term has a high frequency).

We believe that there are two things that a commercial search engine must resolve, in order to provide quality search services on the web. The first thing is to detect properly the type of the queries the users submit, and in each case decide which mechanism (content based or link analysis) should be more efficient. The second thing, is that in each case that link analysis is used, the focused subgraph should contain only highly relevant results, so one must find a way to efficiently prune non-relevant documents from it. Pruning non-relevant documents is something we consider crucial for a retrieval system that uses link analysis and we believe that the significantly *poorer* results of link analysis approaches we used in our experiments are due to the fact that we did not used any mechanism to resolve this. Several approaches have been presented in [1], however we consider more prominent a method based on machine learning concepts.

6.4 Important Issues

We must point out that precision and recall diagrams are not the only means that should be used when evaluating retrieval systems for the WWW. Link analysis was not invented in order to boost these measures. Instead it is used in order to give relevant documents which are also important a better position than non-important (but also relevant) documents, in search results. An attempt to evaluate link analysis methods, on how well they provide this measure of importance has been made in [1].

The lack of standard datasets for evaluating such systems, is a main disadvantage for anyone trying to engage with this topic. Such datasets do exist for text based retrieval methods (e.g., the TREC[13] data sets). This absence of standard datasets, makes impossible for the people that evaluate such systems, to compare the behavior of various implementations of link analysis retrieval methods.

Another disadvantage in the evaluation of link analysis based approaches, is the absence of a framework (e.g., the S.M.A.R.T[6, 14] framework for text retrieval) on which someone can deploy their algorithms. This usually leads people engaged to spend additional time and effort developing retrieval systems similar to ours, and at the same time being unable to compare speed and accuracy with enough certainty.

Chapter 7

Epilogue

7.1 Further Research

In general, what we presented in this document, shows pretty well the advantages and disadvantages of content based and link analysis ranking approaches for image retrieval on the world wide web. However, there are still some important topics that should be examined, before someone decides to deploy such a service, in order to compete with existing image retrieval systems on the web.

We consider that a proper procedure for pruning non-relevant nodes from the focused subgraph that HITS, PicASHOW link analysis approaches use, is a necessary topic of research that someone could process. A procedure that we think as more prominent to accomplish this task would be through relevance feedback mechanism in conjunction with machine learning algorithms for classification, although we have not thought this procedure in more detail.

Another topic of further research would the analysis of user queries, in order for the system to use alternative ranking methods suited for each type of query. Such a mechanism, could possibly result in proper exploiting of individual ranking approaches (e.g., link analysis for more abstract queries, content based methods for more specific queries).

However, it would be more wise to first investigate further the results of from our work, in order to justify more properly any future related work. Such work could include some graphs showing individual scores (chainnet, visual features, etc) in respect to the results ordering. Another useful investigation could involve to measure the "noise" coming from the expansion occurring in the focused subgraph creation (HITS & PicASHOW algorithms). An evaluation of these methods on a larger collection is also a task we believe must be done (we can easily increase the dataset size by a factor of 10, if we forget the visual features extraction at indexing time). These and other statistics on the behavior of these retrieval methods can act as proof for or reveal other important issues for the behavior of the several ranking approaches we study.

Bibliography

[1] Krishna Bharat, Monika R. Henzinger, 1998 “Improved Algorithms for Topic Distillation in a Hyperlinked Environment.”

[2] J. Kleinberg, 1998 “Authoritative sources in a hyperlinked environment.”

[3] R. Lempel, Aya Soffer , 2001 “PicASHOW: Pictorial Authority Search by Hyperlinks on the Web.”

[4] Sergey Brin, Lawrence Page, 1998 “The Anatomy of a Large-Scale Hypertextual Web Search Engine.”

[5] Heng Tao Shen, Beng Chin Ooi, and Kian-Lee Tan, 2000 “Finding Semantically Related Images in the WWW.”

[6] G. Salton, A. Wong, and C.S. Yang, 1975 “A Vector Space Model for Information Retrieval.”

[7] Page, L. 1997 “PageRank: Bringing Order to the Web.”

[8] Porter, M.F. 1980 “An Algorithm for Suffix Stripping.”

[9] Salton, G. and Buckley, C 1988 “Term-Weighting Approaches in Automatic Text Retrieval.”

[10] Heng Tao Shen, Beng Chin Ooi, Kian-Lee Tan, 2000 , “Giving Meanings to WWW images”

[11] Lipyeow Lim, Min Wang, Sriram Padmanabhan, Jeffrey Scott Vitter, Ramesh Agarwal, 2003, "Dynamic Maintenance of Web Indexes Using Landmarks"

[12] http://larbin.sourceforge.net/

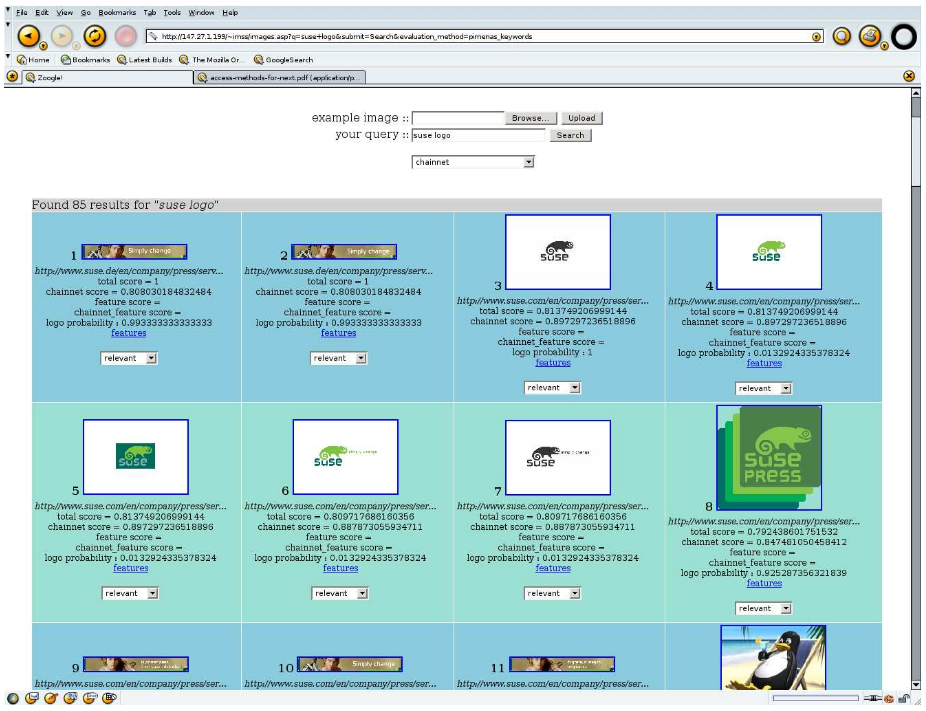
[13] Harman, D.K. 1995 "The TREC Conferences" R. Kuhnlen and M. Rittberger (Eds.) *Hypertext - Information Retrieval - Multimedia: Synergieeffecte Elektronischer Informationssysteme, Proc. of HIM '95*

[14] ftp://ftp.cs.cornell.edu/pub/smart/

[15] http://www.sleepycat.com/

[16] Google Images[http://images.google.com/]

[17] Andrew Y.Ng, Alice X. Zheng, Michael I. Jordan, 2001, "Link Analysis, Eigenvectors and Stability"


File Edit View Go Bookmarks Tab Tools Window Help
Home Bookmarks Latest Builds The Mozilla Or... GoogleSearch
Zoogle!
access-methods-for-next.pdf (application/p...
Browse...
Upload
Search
relevant
Simply change

http://147.27.1.199/~imss/images.asp?q=suse+logo&submit=Search&evaluation_method=pimenas_keywords

example image ::
your query :: suse logo
chainnet

Found 85 results for "*suse logo*"

1
http://www.suse.de/en/company/press/serv...
total score = 1
chainnet score = 0.808030184832484
feature score =
chainnet_feature score =
logo probability : 0.993333333333333
features

2
http://www.suse.de/en/company/press/serv...
total score = 1
chainnet score = 0.808030184832484
feature score =
chainnet_feature score =
logo probability : 0.993333333333333
features

3
http://www.suse.com/en/company/press/ser...
total score = 0.813749206999144
chainnet score = 0.897297236518896
feature score =
chainnet_feature score =
logo probability : 1
features

4
http://www.suse.com/en/company/press/ser...
total score = 0.813749206999144
chainnet score = 0.897297236518896
feature score =
chainnet_feature score =
logo probability : 0.0132924335378324
features

5
http://www.suse.com/en/company/press/ser...
total score = 0.813749206999144
chainnet score = 0.897297236518896
feature score =
chainnet_feature score =
logo probability : 0.0132924335378324
features

6
http://www.suse.com/en/company/press/ser...
total score = 0.809717686160356
chainnet score = 0.887873055934711
feature score =
chainnet_feature score =
logo probability : 0.0132924335378324
features

7
http://www.suse.com/en/company/press/ser...
total score = 0.809717686160356
chainnet score = 0.887873055934711
feature score =
chainnet_feature score =
logo probability : 0.0132924335378324
features

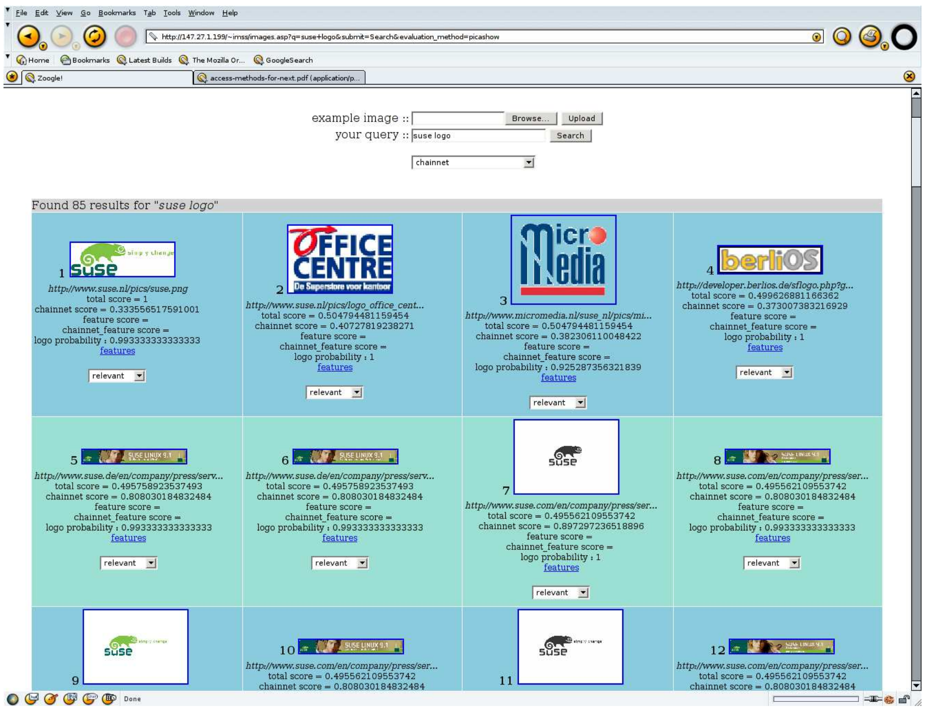
8
http://www.suse.com/en/company/press/ser...
total score = 0.792438601751532
chainnet score = 0.847481050458412
feature score =
chainnet_feature score =
logo probability : 0.925287356321839
features

9
http://www.suse.com/en/company/press/ser...

10
http://www.suse.com/en/company/press/ser...

11
http://www.suse.com/en/company/press/ser...

Figure 7.1: Weighted PicASHOW results for the query "suse logo"


File Edit View Go Bookmarks Tab Tools Window Help
Home Bookmarks Latest Builds The Mozilla Or... GoogleSearch
Zoogle!
access-methods-for-next.pdf (application/p...
Browse...
Upload
Search
relevant
Done

http://147.27.1.199/~imss/images.asp?q=suse+logo&submit=Search&evaluation_method=picashow

example image ::
your query :: suse logo
chainnet

Found 85 results for "*suse logo*"

1
http://www.suse.nl/pics/suse.png
total score = 1
chainnet score = 0.333556517591001
feature score =
chainnet_feature score =
logo probability : 0.993333333333333
features

2
http://www.suse.nl/pics/logo_office_cent...
total score = 0.504794481159454
chainnet score = 0.40727819238271
feature score =
chainnet_feature score =
logo probability : 1
features

3
http://www.micromedia.nl/suse_nl/pics/mi...
total score = 0.504794481159454
chainnet score = 0.382306110048422
feature score =
chainnet_feature score =
logo probability : 0.925287356321839
features

4
http://developer.berlios.de/sflogo.php?g...
total score = 0.499626881166362
chainnet score = 0.373007383216929
feature score =
chainnet_feature score =
logo probability : 1
features

5
http://www.suse.de/en/company/press/serv...
total score = 0.495758923537493
chainnet score = 0.808030184832484
feature score =
chainnet_feature score =
logo probability : 0.993333333333333
features

6
http://www.suse.de/en/company/press/serv...
total score = 0.495758923537493
chainnet score = 0.808030184832484
feature score =
chainnet_feature score =
logo probability : 0.993333333333333
features

7
http://www.suse.com/en/company/press/ser...
total score = 0.495562109553742
chainnet score = 0.897297236518896
feature score =
chainnet_feature score =
logo probability : 1
features

8
http://www.suse.com/en/company/press/ser...
total score = 0.495562109553742
chainnet score = 0.808030184832484
feature score =
chainnet_feature score =
logo probability : 0.993333333333333
features

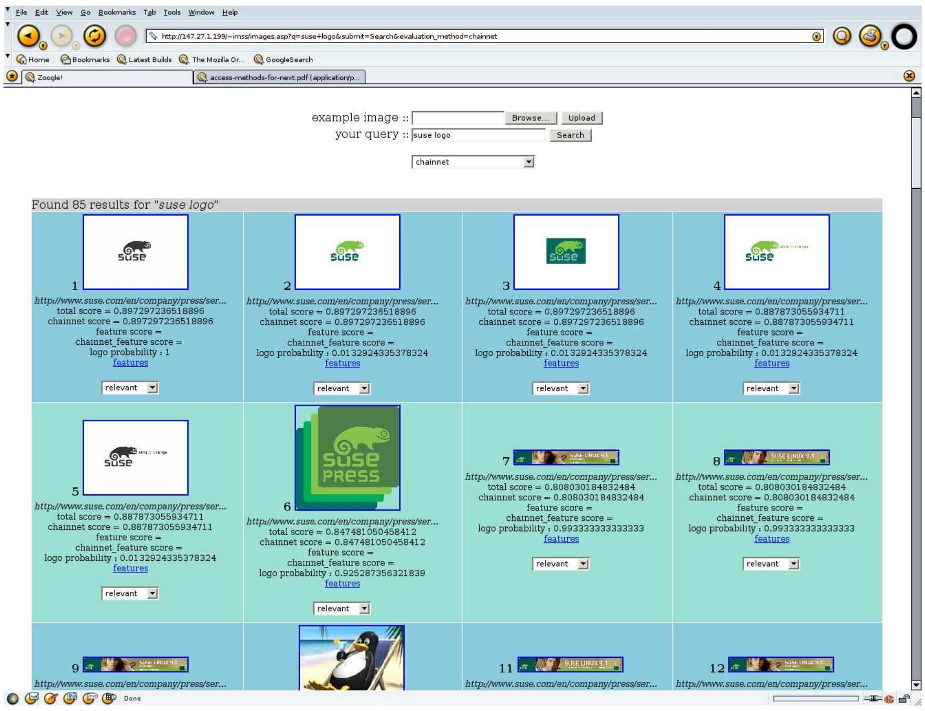
9

10
http://www.suse.com/en/company/press/ser...
total score = 0.495562109553742
chainnet score = 0.808030184832484

11

12
http://www.suse.com/en/company/press/ser...
total score = 0.495562109553742
chainnet score = 0.808030184832484

Figure 7.2: PicASHOW results for the query "suse logo"


File Edit View Go Bookmarks Tab Tools Window Help
Home
Bookmarks
Latest Builds
The Mozilla Or...
GoogleSearch
Zoogle!
access-methods-for-next.pdf (application/p...
Browse...
Upload
Search
Done

http://147.27.1.199/~imss/images.asp?q=suse+logo&submit=Search&evaluation_method=chainnet

example image ::

your query :: suse logo

chainnet

Found 85 results for "*suse logo*"

1
http://www.suse.com/en/company/press/ser...
total score = 0.897297236518896
chainnet score = 0.897297236518896
feature score =
chainnet_feature score =
logo probability : 1
features
relevant

2
http://www.suse.com/en/company/press/ser...
total score = 0.897297236518896
chainnet score = 0.897297236518896
feature score =
chainnet_feature score =
logo probability : 0.0132924335378324
features
relevant

3
http://www.suse.com/en/company/press/ser...
total score = 0.897297236518896
chainnet score = 0.897297236518896
feature score =
chainnet_feature score =
logo probability : 0.0132924335378324
features
relevant

4
http://www.suse.com/en/company/press/ser...
total score = 0.887873055934711
chainnet score = 0.887873055934711
feature score =
chainnet_feature score =
logo probability : 0.0132924335378324
features
relevant

5
http://www.suse.com/en/company/press/ser...
total score = 0.887873055934711
chainnet score = 0.887873055934711
feature score =
chainnet_feature score =
logo probability : 0.0132924335378324
features
relevant

6
http://www.suse.com/en/company/press/ser...
total score = 0.847481050458412
chainnet score = 0.847481050458412
feature score =
chainnet_feature score =
logo probability : 0.925287356321839
features
relevant

7
http://www.suse.com/en/company/press/ser...
total score = 0.808030184832484
chainnet score = 0.808030184832484
feature score =
chainnet_feature score =
logo probability : 0.993333333333333
features
relevant

8
http://www.suse.com/en/company/press/ser...
total score = 0.808030184832484
chainnet score = 0.808030184832484
feature score =
chainnet_feature score =
logo probability : 0.993333333333333
features
relevant

9
http://www.suse.com/en/company/press/ser...

11
http://www.suse.com/en/company/press/ser...

12
http://www.suse.com/en/company/press/ser...

Figure 7.3: ChainNet results for the query "suse logo"