Available online at www.sciencedirect.com
ScienceDirect
Information Systems 32 (2007) 886–908
www.elsevier.com/locate/infosys

Combining text and link analysis for focused crawling—An application for vertical search engines

G. Almpanidis, C. Kotropoulos*, I. Pitas

Department of Informatics, Aristotle University of Thessaloniki, Box 451, Thessaloniki GR-54124, Greece

Received 25 November 2005; received in revised form 7 July 2006; accepted 29 September 2006
Recommended by R. Baeza-Yates

Abstract

The number of vertical search engines and portals has rapidly increased over the last years, making the importance of a topic-driven (focused) crawler self-evident. In this paper, we develop a latent semantic indexing classifier that combines link analysis with text content in order to retrieve and index domain-specific web documents. Our implementation presents a different approach to focused crawling and aims to overcome the limitations imposed by the need to provide initial data for training, while maintaining a high recall/precision ratio. We compare its efficiency with other well-known web information retrieval techniques.

Keywords: Focused crawling; Information retrieval; Latent semantic indexing; Text categorisation; Vertical search engines

1. Introduction

Within the last couple of years, search engine technology had to scale up dramatically in order to keep up with the growing amount of information available on the web. Current large-scale search engines such as Google have the ability to handle billions of pages [1]. But as the amount of web sites is growing rapidly, the number and size of stored documents is growing even faster and site contents are getting updated more and more often. Centralised systems cannot grow that fast and therefore they cover an ever-decreasing segment of the web. The large size and general focus of their indices entails a rather low precision. Naturally, specialised search engines and domain-specific web portals have seen an increasing popularity in recent years. These are also called vertical engines and vertical web portals (also *vectories* and *vortals* [2,3]), respectively. Contrary to large-scale engines, topical search engines are inexpensive in storage requirements and more appropriate to services catering for specialty markets and target groups. A search engine with a specialised index has consequently a more structured content and offers higher precision than a generalised search engine, as it has already been intelligently extracted from the web. This allows for a greater search functionality and information extraction from these pages [4]. Furthermore, a user visiting a specialised search

*Corresponding author. Tel.: +30 2310 996361; fax: +30 2310 998419.
E-mail addresses: galba@aiia.csd.auth.gr (G. Almpanidis), costas@aiia.csd.auth.gr (C. Kotropoulos), pitas@aiia.csd.auth.gr (I. Pitas).

0306-4379/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2006.09.004

engine has a prior knowledge to the covered domain, so extra input to disambiguate the query might not be needed.

In this paper, we study existing methods for targeted web information retrieval (IR) in order to construct or expand a vertical search engine. An important limitation in many supervised methods for web resource discovery is the necessity of an existing large hyperlink graph, required for training and for document matching purposes. Ensuring its index quality and freshness is not always feasible for many small-scale vertical search engines, which are often semi-automatically constructed or maintained. The dynamic nature of information on the web makes it difficult, if not impossible, to construct a complete and up-to-date data representation required for an ideal IR system. The main goal of our work is to provide an efficient information resource discovery algorithm in the context of a topical web search engine, when no previous knowledge of link structure is available, except that found in web pages already fetched during a crawling phase. By assuming an existing text document dataset and integrating text with link analysis in a novel way, we attest that IR in such cases can be greatly improved.

The paper is organised as follows. In Section 2, we discuss methods of intelligent IR on the web. A thorough analysis of both text- and link-based approaches is given as it is essential to the comprehension of our proposed method. In the next section, we study briefly the theoretical background and related works in vertical search engines and focused crawling. In Section 4 we propose a new classifier for automatic hypertext categorisation catering for limited training data. Our method leverages text content and link information in a novel way in order to aid the construction of a small-scale vertical search engine. In Section 5 we present the implementation tactics for our experiments. This section also depicts the experimental results and a comparison against existing methods. Finally, Section 6 concludes the paper, gives a justification of the adopted implementation and assertions, and points out future research topics.

2. Web information retrieval

In data retrieval, we are normally looking for an exact query match. That is, we are checking to see whether an item is present or not in the file. In IR on the other hand, we want to find those items which partially match the request and then select from those a few of the best matching ones [5]. The purpose of an automatic IR strategy is to retrieve all the relevant documents while at the same time retrieving as few of the non-relevant ones as possible. In the early days of the web, IR was just an application of traditional data retrieval techniques where the user's information request is usually expressed by a set of keywords and simple query lexical matching versus indexed documents returned the relevant pages. But there is a crucial difference between simple static text and hypertext documents in that the latter have also hyperlink structure besides their text content, making an alternative representation for them a necessity. Naturally, search engine technology has witnessed a paradigm shift; from early-web traditional text-based ranking schemes to improved algorithms that rely on link analysis. A fine example of the previous argument is the popularity and success of Google [1].

Recently, much effort has been directed to combining these two approaches for efficient IR, i.e., induce available text information in link analysis techniques. In this paper, we evaluate the application of these approaches in targeted web resource discovery and we leverage the combination of textual and linking information aiming at improving the construction of a vertical search engine. Vertical search engines are discussed in Section 3. But first, it is essential to look at text- and link-based IR techniques separately.

2.1. Text-based techniques in web IR

Although the physical characteristics of web information are distributed and decentralised, the web can be viewed as one big virtual text document collection. In that regard, the fundamental questions and approaches of traditional IR research (e.g., term weighting, query expansion) are likely to be relevant in web document retrieval [5]. In early search engines, the basic use of the content of web pages was in the form of exact query matching in a document index database. Assuming that there is a topicality in the web [6], meaning that relevant pages are found close connected together in the web graph, we can argue that the probability of a page u citing pages $\{v_i\}$ similar to it is higher than linking to randomly dissimilar pages. Without loss of generality we can say that if u is relevant to a topic T, a number of $\{v_i\}$ also belong to that topic with a high probability. Therefore, judging from the text

content of a web document, we can estimate the relevance of its outlinks.

Two classic models of text IR are the Boolean and the Vector Space Model (VSM). The Boolean model treats a page as a *set* of keywords, while the vector model treats a page as a *bag* of keywords, thus takes into account the frequency information. In its simplest form, the VSM is described with indexing terms that are considered to be coordinates in a multidimensional space where the documents and the queries are represented as binary vectors of terms. Assuming that we have n documents d_j, $1 \leqslant j \leqslant n$, and a total number of terms t_i, $1 \leqslant i \leqslant m$, where m is the size of the vocabulary V ($m = |V|$), a document is described as an m-dimensional vector in the term space. Accordingly, a query q is also represented as a binary vector of dimension m where the occurrence of a term is marked with 1 or 0 otherwise. Alternatively, term frequencies instead of binary values can be used in each document vector.

The language-independent "bag-of-words" representations of documents have proved surprisingly effective for text classification [7]. In this section, we shall present two extensions of the classic VSM that yield better results and, later, we shall use them as classifiers in focused crawling. The first one is based on weighting and probabilistic ranking while the second one utilises the Latent Semantic Indexing (LSI).

2.1.1. Probabilistic ranking

Term Frequency—Inverse Document Frequency (TF-IDF) is a classical global weighting scheme for an indexing system and VSM where terms appearing in documents are weighted proportionally to term frequency and inversely proportional to the document frequency [7]. In a more sophisticated extension of this approach, the documents of a collection are ranked in a decreasing order of their probability of relevance to a user's query, a principle known as probability ranking [8]. This method is based on term reweighting, using the baseline IR algorithm of [8] and yields better results than the simple TF-IDF scheme. Assuming that we have the VSM of Section 2.1, the following measures are used:

The *collection frequency weight* $CFW(t_i)$ of a term t_i implies that terms appearing in few documents are more valuable than those appearing in many, i.e.

$$CFW(t_i) = \log \frac{n}{n_i}, \tag{1}$$

where n_i is the occurrence of term t_i in the collection.

The *average document length*, $NDL(d_j)$, is the normalised document length and is given by

$$NDL(d_j) = \frac{DL(d_j)}{\overline{DL}}, \tag{2}$$

where $DL(d_j)$ is the document length of document d_j and $\overline{DL}$ is the average document length in the corpus of n documents. The *term frequency weight* $TF(t_i, d_j)$ of a term t_i in a document d_j is defined as the number of occurrences of this term in the document and states that a term appearing many times in a document is likely to be more important for that document. The *combined weight* $CW(t_i, d_j)$ of a term t_i in a document d_j is the combination of all the above measures

$$CW(t_i, d_j) = \frac{CFW(t_i) \times TF(t_i, d_j) \times (k_1 + 1)}{k_1 \times ((1 - b) + (b \times (NDL(d_j))) + TF(t_i, d_j))}. \tag{3}$$

The constant k_1 determines the effect of term frequency in combined weight, while the constant b ($0 \leqslant b \leqslant 1$), controls the effect of document length. Suggested values according to [8] are $k_1 = 2$ and $b = 0.75$.

2.1.2. LSI—Singular Value Decomposition updating

Various approaches to VSM depend on the construction of a term–document two-dimensional $m \times n$ matrix where each of the m rows represents the occurrence of a term in all the documents of the collection, while each of the n columns represents a single document in term space. In a term–document matrix A that consists of $m \times n$ elements, where m is the number of terms and n is the number of documents in the collection, the i,jth cell denotes the relation between the ith term and jth document. Typically, if a term exists in the document the value 1 is assigned, while a zero value denotes the absence of a term in a specific document. Alternatively, the value of cell A_{ij} can correspond to the frequency of term i in document j. The term–document matrix is generally sparse with $n \gg m$ for large-scale search engines with sizeable indices. This rule does not apply to small vertical engines covering a few hundred or thousand of documents, where these values are comparable. The problem with the simple term–document matrix approach is that the word contexts can become too large. As a simple solution, we can use smaller contexts (machine-readable dictionaries) or reduce the space using various multidimensional scaling and factor analysis techniques.

LSI, an optimal special case of multidimensional scaling, is a concept-based automatic indexing method that tries to overcome the two fundamental problems which plague traditional lexical-matching indexing schemes: synonymy and polysemy [9]. LSI models the semantics of the domain in order to suggest additional relevant keywords and to reveal the "hidden" concepts of a given corpus while eliminating high-order noise. The attractive point of LSI is that it captures the higher-order "latent" structure of word usage across the documents rather than just surface-level word choice. This is done by modelling the association between terms and documents based on how terms co-occur across documents [10]. The dimensionality reduction is typically computed with the help of Singular Value Decomposition (SVD), as it is discussed below. SVD is similar to principal components analysis, which can only be used with square matrices and has been successfully applied to many problems in other domains such as visual object recognition. The SVD of a given matrix A, is defined as

$$A = USV^{\mathrm{T}}, \tag{4}$$

where A is the term–document matrix, U and V are $m \times k_0$ and $n \times k_0$ matrices, respectively, with orthonormal columns ($U^{\mathrm{T}}U = V^{\mathrm{T}}V = I$) with $k_0 = rank(A)$. The columns of U and V define the orthonormal eigenvectors associated with the non-zero eigenvalues of AA^{T} and $A^{\mathrm{T}}A$. S is a diagonal $k_0 \times k_0$ matrix which contains the singular values of A arranged from the largest to the smallest. In LSI, an approximated version of A is computed by truncating its singular matrices and is denoted by A_k where $k = rank(A_k) < k_0$. Therefore, only the k singular values and their associated left and right eigenvectors are used for retrieval:

$$A_k = U_k S_k V_k^{\mathrm{T}}. \tag{5}$$

The dimensions of U_k, V_k, and S_k are now $m \times k$, $n \times k$, and $k \times k$, respectively, while A_k remains $m \times n$. In SVD the eigenvectors with the largest eigenvalues capture the axes of the largest variation in the data. Therefore, it is logical to assume that by neglecting the smallest values we extent the representation of the model, but retain its main semantic aspects. The dimensionality reduction of LSI offers considerable computational performance improvement over classic VSMs but also achieves noise reduction and yields better results in text IR [11].

In VSM, the similarity between the query q and the n documents in the database can be found by calculating the cosines between the query vector and the document vectors. In LSI, the cosine is the result of the comparison of the query vector to the columns of matrix A_k. The cosine between a query vector q and a document j (i.e., a column j of A_k) is defined as

$$\cos\theta_j = \frac{(A_k e_j)^{\mathrm{T}} q}{||A_k e_j||_2 ||q||_2} = \frac{e_j^{\mathrm{T}} V_k S_k (U_k^{\mathrm{T}} q)}{||S_k V_k^T e_j||_2 ||q||_2}, \quad j = 1, 2, \ldots, n, \tag{6}$$

where e_j denotes the jth canonical vector of dimension n (i.e., the jth column of the identity matrix $I_{[n \times n]}$, q is $m \times 1$, and $||a||_2$ stands for the L_2 norm of vector a.

In typical text document IR, there is little need for a frequent reorganisation of the training corpus since its size is assumed large enough and the affect of inserting new documents is insignificant. New terms not existing in the vocabulary V are considered out of vocabulary words and new documents are never inserted in the system. But there are situations where either the corpus is relatively small or the already classified by the system documents are used for relevance feedback. This is the problem of targeted web resource discovery where incoming information is extremely important in assessing more documents and, therefore, it should be integrated to the system. As it will be seen in Section 4, frequent and efficient corpus updating in such systems is crucial. Adding new pages to the corpus or modifying existing ones also means that the index has to be regenerated for both the recall and the crawling phase. But once an index is created it might be obsolete in a matter of seconds when new information (terms and documents) need to be inserted to the system [12]. Depending on the indexing technique followed, this can be a computationally intensive procedure. Therefore, the practical application of matrix decomposition such as the SVD in dynamic collections, such as the index of a vertical search engine, is not a trivial problem.

In the case of LSI, there are well-known and relatively inexpensive methods such as *fold-in* and *SVD updating* that avoid the reconstruction of the term–document matrix and the need for a new decomposition. According to [12], updating refers to the general process of adding new terms and/or documents to an existing LSI-generated database.

Updating can mean either folding-in or SVD updating

- *SVD updating* is a method of updating where new terms and documents are folded to matrix A_k [13].
- *Folding-in* terms or documents is a much simpler alternative that uses an existing SVD to represent new information.
- *Recomputing* the SVD is not an updating method, but a way of creating an LSI-generated database with new terms and/or documents by reconstructing the original matrix A and performing SVD to the new term–document matrix.

Recomputing the SVD of a large term–document matrix requires computation time and, for large problems, may be impossible due to memory constraints. In contrast, folding-in is based on the existing latent semantic structure and hence new terms and documents have no effect on the representation of the pre-existing terms and documents. Furthermore, the orthogonality in the reduced k-dimensional basis for the column or row space of A (depending on inserting terms or documents) is corrupted. This can have deteriorating effects on the representation of the new terms and documents. The advantage of folding-in is that it requires less time and memory and is simple to execute. Each new document in this technique is represented as a weighted sum of its component term vectors. Once a new document vector has been computed it is appended to the set of existing document vectors. Similarly, new terms can be represented as a weighted sum of the document vectors in which they appear.

SVD updating, while more complex, maintains the orthogonality and the latent structure of the original matrix [13]. Adding both new documents and terms in an LSI reduced subspace using SVD updating involves three steps: add new documents, add new terms, and make correction changes in term weighting. The new expanded term–document matrix C can be depicted in Fig. 1.

If we want to add t new term vectors then $T_{[t\times n]}$ is a sparse matrix, since each term rarely occurs in every document of the original space. In SVD updating, we append T to the rows of A_k:

$$B_{[(m+t)\times n]} = \begin{pmatrix} A_{[m\times n]} \\ T_{[t\times n]} \end{pmatrix}. \tag{7}$$

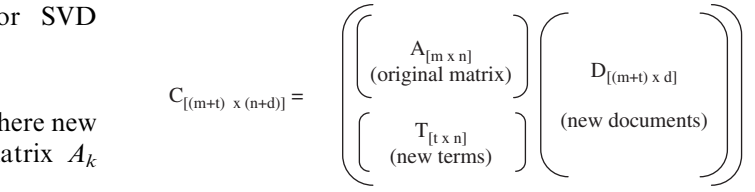

Fig. 1. SVD updating: inserting t new terms and d new documents.

If we define

$$SVD(B) = U_B S_B V_B^{\mathrm{T}}, \quad H_{[(k+t)\times k]} = \begin{pmatrix} S_k \\ TV_k \end{pmatrix} \quad \text{and}$$

$$SVD(H) = U_H S_H V_H^{\mathrm{T}},$$

then, according to [13]:

$$U_B = \begin{pmatrix} U_k & 0 \\ 0 & I_t \end{pmatrix} U_H, \quad S_B = S_H \text{ and}$$

$$V_B = V_K V_H. \tag{8}$$

We observe that, instead of performing an SVD on the $(m+t)\times n$ matrix B, we only have to do it for the $(k+t)\times k$ matrix H. Since $k \ll \min(m,n)$, the performance gain in computation is significant.

If we want to insert additionally d new documents, we now append the sparse matrix $D_{[(m+t)\times d]}$ to the columns of matrix B:

$$C_{[(m+t)\times(n+d)]} = (B|D). \tag{9}$$

If we define $SVD(C) = U_C S_C V_C^{\mathrm{T}}$, $F = (S_H | U_B^{\mathrm{T}} D)$ and $SVD(F) = U_F S_F V_F^{\mathrm{T}}$ then:

$$V_C = \begin{pmatrix} V_B & 0 \\ 0 & I_d \end{pmatrix} V_F, \quad S_C = S_F \quad \text{and}$$

$$U_C = U_B V_F. \tag{10}$$

Again, the benefit in computation reduction is considerable.

2.2. Link analysis techniques in web IR

What really differentiates hypertext from static text documents is the fact that the former, besides the text content, contain additional semantics, such as linking information, citations and metadata. Furthermore, the existence of hypertext structure implies that different features and information units can also be exploited (e.g., tag contexts, Document Object Model (DOM) trees, etc. instead of paragraphs and sentences). Contrary to text-based techniques, link analysis' main target is to identify

the *importance* or *popularity* of documents. This task is clearly derived from earlier work in social network analysis [14] and bibliometrics citation analysis, specifically in bibliographic coupling [15] and co-citation [16]. There, the objective is to study the patterns of how scientific papers make reference to one another and attempt to identify topics in a document collection, as well as influential authors and papers on those topics, based on patterns in citation frequency. The best-known measure of a journal importance is the *impact factor*. This metric essentially judges a publication by the number of citations it receives [17].

Likewise, link and social network analysis have been successfully applied recently to web hyperlink data in order to identify authoritative information sources [18]. The common assumption in most web IR techniques is, as in social networks in academic publications, that hyperlink confers authority. A link from one page u to another v corresponds to u author's belief that v contains valuable information. Here, *valuable* can mean either that v is relevant/similar to the conceptual contents of u or belongs to the same topic and has some information that the user might find useful. Under this hypothesis, pages strongly cited by many are considered to play an influential role in the web IR. Similarly, the impact factor in web IR would correspond to the ranking of a page simply by a tally of the number of links that point to it. This is known as *BackLink count* (BL) or *Link In-Degree* and is defined as the number of the ancestors of a hypertext document in the web graph. But the use of BL is typically not appropriate in the setting of web IR. Since the web is considered an open system for many applications, including crawlers, the BL is never known, but it is approximated based on the web pages retrieved so far. Also, BL can favour universally popular locations, such as the home page of Yahoo, regardless of the specific query topic. It is clear that this metric is sensitive to link spamming. Consequently, BL can only serve as a rough, heuristic-based measure of document importance and quality in social network analysis. More intelligent connectivity-based page quality metrics that take into account the cumulative prestige of the pages that link to it are required.

Obviously, the traditional social network analysis techniques are not always applicable in the web setting. The massive size, the diversity in content, context, format, purpose, and quality, and the dynamic distribution of the web document collection, all make impractical previous IR research findings based on small and homogenous test collections [19]. Contrary to scientific journals, web documents are inherently noisy. They have an unstructured style and an immense authoring variety and their connectivity linking information is an implicit rather than an explicit indication of similarity. For example, the former assumption about the relation between hyperlinking and authority does not apply for all links in a web document. The link from one page u to another v can serve multiple tasks, such as navigation, expansion, and resource, besides citation [20]. Also in recent years, the number of automatically generated dynamic links or explicitly placed advertisements and banners has dramatically increased. Another fact attributing to the high noise of the web is that the contents of page v might have changed since page u and hyperlink (u,v) have been created, thus making the authors original intentions invalid.

Two of the most promising web IR tools, namely *Google* [1] and *Clever* [21] seem to use combinations of retrieval techniques identified above. Clever combines topic-dependent link analysis techniques called *HITS* with term similarity techniques of text retrieval, while Google seems to employ a number of text retrieval techniques to obtain high-performance text retrieval ranked heavily by a universal link analysis score called *PageRank*. Both Google and Clever not only leverage heavily off the implicit human judgment embedded in hyperlinks, but also improve on link analysis results by combining it with text retrieval techniques [19].

2.2.1. Pagerank (PR)

According to Brin and Page [22], a link from page u to page v is a vote by u's author to v, meaning that u's author cites v because it is popular and/or relevant to u. The intuition behind PageRank is that not all "votes" count equally, but citations from "better" (important/popular) pages count more. In other words, the significance of a page depends on the significance of those referencing it. Because of this cumulative prestige of the pages that link to a page, PageRank is a better measure of its importance or authoritativeness than BackLink count [22]. If page p has n pages $q_1, q_2, \ldots, q_n$ which point to it (i.e., they are citations), d is a damping factor with $0<d<1$ and $C(q_i)$ is the number of links going out of page q_i (i.e., the out-degree) then the PageRank of the page p is

given as follows:

$$PR(p) = (1-d) + d\sum_{i=1}^{n} \frac{PR(q_i)}{C(q_i)}. \quad (11)$$

As illustrated in Fig. 2, the PageRank algorithm recursively defines the importance of a page to be the weighted sum of its backlinks' importance values. PageRank scores form a probability distribution over web pages, so the sum of all web page PageRanks will be one.

The rank of a page is divided evenly among its outlinks to contribute to the ranks of the pages they point to. The equation is recursive, but starting with any set of ranks and iterating the computation until it converges we are able to compute it. PageRank can be calculated using a simple iterative algorithm and yields the principal eigenvector of a specially normalised link matrix of the web. The main intuition behind PageRank is that it simulates the actions of a random surfer. With a probability of d, a user that is currently at a web page p can either click one of p outlinks or, with a probability d, can jump to a random page of the web. The parameter d can be estimated by statistical analysis of user surf tactics. It is usually selected between 0.8 and 0.9.

2.2.2. HITS

An alternative but equally influential algorithm of modern hypertext IR is Kleinberg's HITS [23], first implemented in the Clever [21] search engine from IBM. According to Kleinberg [23], web pages can be categorised to two different classes. *Target* pages or *authorities* are rich and relevant in text content to the user's query and these only need to be indexed in a vertical search engine. On the other hand, pages that might not have relevant textual information, but can lead to relevant documents are called *hubs*. The interesting part is that discovering the good hubs is not a straightforward task, since a hub might not reference directly authorities, but other hubs which in return link to relevant target pages. Pages of this kind (hubs) do not need to be indexed in a vertical engine as they are of little interest to the end user but they play a significant role in the crawling process. They have to be stored in a different manner than authorities. However, both kinds of pages can collaborate in a combined analysis procedure in order to determine the path of a focused crawler. With reference to Fig. 2, the authority and hub scores for each document in the HITS algorithm are calculated iteratively as follows:

$$AuthorityScore(p) = \sum_{\text{all } q \overset{\text{linking}}{\underset{\text{to}}{}} p}^{n} HubScore(q), \quad (12)$$

$$HubScore(p) = \sum_{\text{all } r \overset{\text{linking}}{\underset{\text{from}}{}} p}^{m} AuthorityScore(r). \quad (13)$$

Unlike PageRank, HITS offers a query-dependent connectivity-based ranking which means that it measures the quality and the relevance of a page to a given user query [24]. As a result, no universal prestige ranking can or needs to be calculated, making the algorithm more appropriate to tasks such as web crawling. It must be stated that HITS can be extended trivially to be query-independent too, by considering the root set as the full Web graph.

Bharat and Henzinger [25] argue that HITS does not work satisfactorily in cases where there is a mutually reinforcing relationship between hosts (nepotism), or there is high noise due to automatically generated links, often by using web authoring tools, or when nodes in the neighbourhood graph, non-relevant to the topic, tend to drift the followed path (topic drift). In their proposed algorithm, nodes have additional properties and they make use

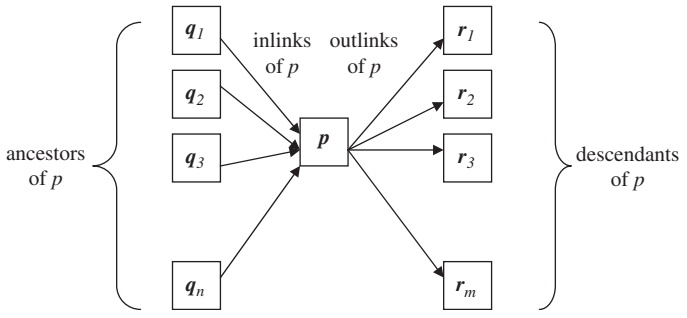

Fig. 2. PageRank and HITS algorithms.

of web page content in addition to its graph structure.

2.3. Combining text and link information for web IR

Numerous techniques try to combine textual and linking information for efficient URL ordering in the literature. Many of these are extensions to either PageRank or HITS. Cohn and Hoffman [26] describe a joint probabilistic model extending previous work at Probabilistic LSI/Analysis (PLSI) and Probabilistic HITS (PHITS). PLSI is a variant of LSI that builds a factored multinomial model based on the assumption of an underlying document generation process [26]. Like LSI, PLSI is a latent variable model for general co-occurrence data, but instead of using SVD, PLSI performs a probabilistic decomposition and which associates an unobserved class variable $z_k \in Z = \{z_1, z_2, \ldots, z_K\}$ with each observation or occurrence of term t_i and citation c_i in document d_j. The document d_j is then represented as a convex combination of factors with mixing weights $P(z_k|d_j)$. The predictive probabilities for terms in this document are non-negative:

$$P(t_i|d_j) = \sum_{k=1}^{K} P(t_i|z_k)P(z_k|d_j) \tag{14}$$

with the normalisation constraints $\sum_{i=1}^{M} P(t_i|z_k) = 1$ for all k and $\sum_{k=1}^{K} P(z_k|d_j) = 1$ for all j. Using the Expectation-Maximisation (EM) algorithm, PLSI aims at finding the local maximum of the likelihood of the observed term frequencies:

$$L = \sum_{j=1}^{N}\sum_{i=1}^{M} TF(t_i, d_j) \log\left[\sum_{k=1}^{K} P(t_i|z_k)P(z_k|d_j)\right], \tag{15}$$

where $TF(t_i,d_j)$ is the term frequency of t_i in document d_j.

In analogy with HITS, PHITS performs a probabilistic factoring of document citations used for bibliometric analysis. It replaces the eigenvector analysis of HITS with a probabilistic so that the resulting model has clear statistical interpretations.

The PLSI + PHITS model is a principled probabilistic model for document–term and document–citation pairs and has been shown to provide improved classification performance [26]. Subsequently, the following convex combination needs to be maximised:

$$L = a\sum_{j=1}^{N}\sum_{i=1}^{M} TF(t_i, d_j) \log P(t_i, d_j) + (1-a)\sum_{j=1}^{N}\sum_{l=1}^{K} LF(d_j, c_l) \log P(d_j, c_l), \tag{16}$$

where $LF(d_j,c_l)$ corresponds to the frequency occurrence of citation c_l in document d_j. The parameter a sets the relative weight of text and connectivity information. If a is 1 the model considers only text, while if it is 0 it takes into consideration only citations. A second parameter, b, controls the EM algorithm in order to avoid overfitting [27]. This tempered EM is usually implemented by continuously decreasing b until it reaches a set lower limit or until that does not yield further improvements.

3. Vertical search engines and focused web crawlers

3.1. Search engines

Extracting useful knowledge from large amounts of hypertext has become a topic of great interest, in part because of the huge volume of data that are now available on the web. But this enormity and rapid growth of the web also indicates that it becomes more and more difficult to extract relevant information from it. A search engine is a system that collects and organises web documents, and presents a way to select documents based on certain words, phrases or patterns within documents [28]. The recall of even the largest commercial search engines is rather low, covering 50–70% of the web today [29]. This means that existing large-scale search techniques are not always able to retrieve topic-related information from pages stored deeply in a hyperlink tree and they are not ideal for providing topic-oriented information. Subsequently, the plethora of available information led to the introduction of human-maintained directories, web portals, and vertical search engines that cover a small portion of the web with a high precision and organise information by topic.

The creation and maintenance of a vertical search engine is usually seen as task of *classification* requiring supervised automatic categorisation of text documents into specific and predefined topic areas. In clustering, the document collections are processed and grouped into clusters that are

dynamically generated by an algorithm. On the other hand, in classification large volumes of data break apart into several discrete classes that are a priori determined on the basis of a training dataset and a user-provided taxonomy [30]. Here, a corpus is constructed manually or semi-automatically and documents are labelled to predefined categories. During the training phase the agent knows the correct answer for each input state and builds a model from the set of preclassified documents. Using this knowledge, the agent/crawler guesses the label for unlabelled cases in training and expands the system with new documents. Consequently, text categorisation is more suitable than clustering for vertical search engine design, since a conceptual preunderstanding of the domain and knowledge of each topic's attributes is required and assumed. These topics are therefore well defined even though they might contain no data at first. Each one can be described with a set of keywords or a few text examples manually selected which help the construction of the search engine at early stages.

Human ability to judge whether a web page is relevant to a specified topic and distinguish provocative misuse of web authoring ethics is still exploited in web portals and vertical search engines. The maintenance and document classification tasks are usually executed by human indexers because of their very high precision. Human indexers have still an important role in the creation and maintenance of vertical portals and search engines mainly because they implement common tactics that are difficult for a machine to emulate or outperform. Therefore, a system should assist but not replace the work of human indexing. It is evident that this work can be greatly enhanced by providing documents with high probability of being relevant and/or important than having to deal with a plethora of unrelated documents. Our goal is to provide a semi-automatic methodology of adding relevant pages to the portal, assisting but not replacing completely human effort.

3.2. Web crawlers

Alternatively to human indexing, the construction process of the index of a search engine can be done with the help of a *crawler*. A crawler is an agent that traverses the hypertext structure of the web automatically, starting from an initial hyperdocument or a set of starting points (seeds) and recursively retrieving all documents referenced by that document. Web crawlers are also referred to as web wanderers, web robots or spiders. Generally, crawlers can be used for a number of purposes such as indexing, HTML validation, link validation, and mirroring [31,32]. The visiting strategy of new web pages usually characterises the purpose of the search engine. Generalised search engines that seek to cover as much proportion of the web as possible usually implement a BReadth-First Search (BRFS) or Depth-First Search (DFS) algorithm. The BRFS policy is implemented by using a simple FIFO queue for the unvisited documents. BRFS order provides a fairly good bias towards high-quality pages without the computational cost of keeping the queue ordered [33].

Systems on the other hand that require high precision and targeted information must seek new unvisited pages in a more intelligent way. The crawler of a vertical search engine is assigned the task to automatically classify crawled web pages to the existing category structures and simultaneously have the ability to further discover web information related to the specified domain. Due to its size and dynamic nature, the web can be described as an open set of hyperlinked documents. Consequently, the optimum path to relevant documents is always unknown; an efficient crawler has to guess promising links by taking decisions that are based on predefined rules/metrics each influencing its performance differently. A *focused* or *topic-driven crawler* is a specific type of crawler that analyses its crawl boundary to find the links that are likely to be most relevant for the crawl while avoiding irrelevant regions of the web. Focused (thematic) crawling [34] is a relatively new, promising approach to improving the recall of expert search on the web. It involves the automatic classification of visited documents into a user or community-specific topic hierarchy (ontology).

A popular approach for focused resource discovery on the web is the BeSt-First Search (BSFS) algorithm where unvisited pages are stored in a priority queue, known as *frontier*, and they are reordered periodically based on importance or similarity criteria. A typical topic-oriented crawler performs a BSFS strategy by keeping two queues of URLs. One containing the already visited links (from here on **AF**) and another having the references of the first queue called *Crawl Frontier* (from here on **CF**). The pseudocode of a typical BSFS crawler is shown at Fig. 3. The challenging task of this crawler is to order the links in the CF

```
1:  enqueue seed URL into Crawl Frontier CF
2:  while CF is not empty
3:     dequeue URL u from CF
4:     download page u
5:     parse u and extract its outlinks {u,v} plus other features
6:     for each outlink URL v found:
7:       add (u,v) to link database
8:       if v is not already visited (does not exist in AF) then:
9:            enqueue v to AF
10:    if (reorder criteria, e.g. every N documents fetched) then:
11       reorganise CF according to priority ordering policy
11:  end
```

Fig. 3. Typical BSFS-based crawler pseudocode (CF = Crawl Frontier, AF = queue of Already Fetched documents).

efficiently. Various techniques have been researched and implemented [34].

Chakrabarti et al. [34] discriminate hypertext mining in two separate modules that constitute a web crawler: a *classifier* that evaluates the content relevance of a region of the web to the user's interest and a *distiller* that evaluates a page as an access point for a large neighbourhood of relevant pages. The importance metrics for the crawling can be either interest driven where the classifier for document similarity checks the text content and popularity or location driven where the importance of a page depends on the hyperlink structure of the crawled document [36]. The inherited problem in focused crawlers is that the system does not have any textual information about the referenced documents in the CF since these have not been downloaded yet. Even then, content alone does not always guide efficiently a focused crawler to topic-related regions of the web. On the other hand, a document, relevant to the trained dataset, can be referenced by non-relevant documents and vice versa. This is actually an important assumption in the HITS algorithm where hubs can lead to authoritative pages while being unrelated to the initial query themselves [23]. Nevertheless, it is strongly argued that the hypertext structure yields better results in focused crawling than interest-driven algorithms [34]. Ideally, the convergence of these two approaches would alleviate many of the limitations discussed earlier. Some well-known methods of focused resource discovery are presented in the next subsection.

3.3. Related work on focused crawling

The ubiquity of search engines in our lives as an information discovery tool has led to many recent advances in web crawling technology. Various methods utilising text or content analysis have been researched. Two commonly used topic distillation algorithms exploiting the hyperlink structure of the web are PageRank and HITS. Kleinberg [23] has an in-depth discussion on their applicability to web crawling. On the other hand, content similarity remains an essential objective for assessing the classification of a crawled document. Here, well-known IR techniques and text categorisation algorithms can be applied.

Numerous techniques that try to combine textual and linking information for efficient URL ordering exist in the bibliography. Many of these are extensions to either PageRank or HITS. The search engine Google [1] is based on PageRank but also combines the ranking with sophisticated feature-matching techniques. These are not necessarily only term-based but also take into account additional page features and heuristic-based algorithms. BSFS crawlers using PageRank as their heuristic are discussed in [35,37]. An application of PageRank to target seeking crawlers can improve the original method by employing a combination of PageRank and similarity to the topic keywords [38]. The URLs at the frontier are first sorted by the number of topic keywords present in their parent pages, then they are sorted by their estimated PageRanks. A set (precomputed) biased PageRank vectors instead of a single generic vector is used in [39] in order to generate query-specific importance scores for pages at query time and to yield more accurate rankings. Baeza-Yates et al. [40] surveys on the usefulness of historical information from previous crawls several page ordering strategies including PageRank. An interesting work discussing how rapidly the computation of PageRank over a visited subgraph yields relative rank is [41]. It is deduced that PageRank locality results to poor approximation of a PageRank on a partial crawl when high-quality pages are accumulated in short time and vice versa.

An application of PLSI in web IR is introduced in [26]. There, existing links between the documents are used as features in addition to word terms. The

hypothesis is that the hyperlinks contribute to the semantic context of the documents and thereby enhance the chance of successful applications. Two documents having a similar citation pattern are more likely to share the same context than documents with different citation patterns. An intelligent web crawler is suggested based on a principle of following links in those documents that are most likely to have links leading to the topic at interest. The topic is represented by a query in the latent semantic factor space.

Diligenti et al. [42] propose supervised learning on the structure of paths leading to relevant pages to enhance target seeking crawling. A link-based ontology is required in the training phase. Another similar technique is Reinforcement Learning where a focused crawler is trained using paths leading to relevant goal nodes. This crawler seeks to predict the total benefit from following a link (u,v) starting from page u [43]. The effect of exploiting other hypertext features such as segmenting DOM tag-trees that characterise a web document and propose a fine-grained topic distillation technique that combines this information with the HITS algorithm is studied in [44]. Keyword-sensitive crawling strategies such as URL string analysis and other location metrics (URLs with fewer slashes are more useful than URLs with many slashes; The importance of a page p is a function of its location and not of its contents) is investigated in [37]. In [45], there is a discussion of an intelligent crawler that can adapt the queue link-extraction strategy during crawling to reduce the likelihood of starvation. A noteworthy characteristic of this design is that it does not depend on the existence of topical, pretrained examples but on the crawler *auto-focus*, after a number of link visits. The Bingo! engine is a well-known focused crawling system using the HITS algorithm for link analysis, Support Vector Machines as a document classification method and the Kullback–Leibler divergence for feature space construction [46]. A fully scalable, fault-tolerant and distributed web crawler is described in [47]. Varlamis et al. [48] also incorporates linking semantics in addition to the textual concepts for the task of web page classification into topic ontologies. Bergmark et al. [49] uses tunnelling to overcome some of the limitations of a pure BSFS approach such as the dependence in on-topic seeds. Links are not only prioritised according to the page's relevance score, but also according to the estimated value of each link.

Work on evaluating different crawling strategies is described in [50]. Classifiers were built for each of 100 topics, to be used to evaluate the crawled pages. The authors argue that a good focused crawler should remain in the vicinity of the topic in vector space. They plot a trajectory over time and assess the crawlers based on their ability to remain on topic. Three different focused crawling strategies were evaluated:

- *BestFirst crawler*: it uses a priority queue ordered by similarity between topic and page where link was found.
- *PageRank crawler*: it crawls in pagerank order, and recomputes ranks every 25th page.
- *InfoSpiders* [51]: it uses neural net, back propagation, and considers text around links.

It was found that BestFirst performed best, followed by Infospiders, and then PageRank.

The quality of the training data for the classifier is the most critical issue and potential bottleneck for the effectiveness and scale of a focused crawler. Our implementation, discussed in the following section, presents an approach to focused crawling that aims to overcome the limitations of the initial training data.

4. Hypertext combined latent analysis (HCLA)

A common problem in web IR is the availability quality connectivity link corpora. For research and evaluation purposes available corpora such as the WT2 g collection have been criticised for their lack of cross-hosts links [52]. Regarding search engine systems, even if a hyperlink corpus is given, its freshness is always an issue. With the breathtaking evolution of the web it is a matter of time for the corpus linking data to become obsolete. Recent studies indicate that about 80 percent of all links in the link structure will have changed or be new within a year, 50 percent of all contents will be changed within the same period and about 20 percent of the web pages of today will disappear within a year [53]. Although, others argue that the web in fact is in fact becoming denser [54]. Nevertheless, large-scale search engines might be able to recompute their index (or even a part of it) in a relatively short period of time and guide a focused crawling process, but for smaller-scale vertical engines the need of updating the index continuously is vital [55]. On the contrary, text-based corpora are

in abundance and can be easily constructed from sample documents. As stated earlier, in this paper we evaluate various crawling algorithms that can be applied to the case when only an unlabelled (but known to contain sufficiently enough relevant sample documents) text corpus is available, without any linking information and we seek web documents relevant to the given topics. The topics are described by a set of keywords. In other words, the problem raised is the implementation of a focused crawler for target topic discovery, given unlabelled textual data, a set of keywords, and no other data resources. Taking into account these limitations many sophisticated algorithms of Section 3.3, such as HITS and context graphs, cannot be applied.

The main contribution of our work is an algorithm (called *Hypertext Content Latent Analysis* or **HCLA** from now onwards) that tries to combine text with link analysis using the VSM paradigm. Unlike PageRank, where simple eigen-analysis on globally weighted adjacency matrix is applied and principal eigenvectors are used, we choose to work with a technique more comparable with HITS. While the effectiveness of LSI has been demonstrated empirically in several text collections yielding an increased average retrieval precision, its success in web connectivity analysis has not been straightforward. There is a close connection between HITS and LSI/SVD multidimensional scaling [55]. The HITS algorithm is equivalent to running SVD on the hyperlink relation (source, target) rather than the (term, document) relation to which SVD is usually applied. As a consequence of this equivalence, a HITS procedure that finds multiple hub and authority vectors also finds a multidimensional representation of the nodes in a web graph and corresponds to finding many singular values for AA^{T} or $A^{\mathrm{T}}A$, where A is the adjacency matrix.

Our main assumption is that terms and links are both considered representative of document relevance in an expanded matrix. They are seen as *relationships*. In the new space introduced, each document is represented by both the terms it contains and the similar text and hypertext (e.g., outlinks) documents. This is an extension of the traditional 'bag-of-words' document representation of the traditional VSM described in Section 2.1.

Unlike [26], simple LSI instead of a PLSI is used in our work. The proposed representation offers a number of benefits. First, text only queries can be applied to the enriched relationships space so that documents having only linking information, such as those in CF, can be ordered. Secondly, the method can be easily extended for the case where we also have estimated content information for the documents in CF. This can be done using the anchor text and the neighbour textual context of the link tag in the parent HTML source code, following heuristics to remedy for problem of context boundaries identification [56,57]. Moreover, we can easily apply local weights to the terms/rows of the matrix, a common technique in IR that can augment LSI efficiency. While term weighting in classic text IR is a kind of linguistic favouritism, here it can also been seen as a method of emphasising either the use of linking information or text content. The main disadvantage of HCLA is the complexity of updating the weights in the expanded matrix, especially when a global weighting scheme is used for LSI. For simplicity, we only used a simple weighting scheme during the tests.

The application of this algorithm in a BSFS crawler is depicted in Fig. 4. Matrix A is the original text-only term–document representation while $\begin{pmatrix} L_{[m\times a]} \\ G_{[a\times a]} \end{pmatrix}$ and $\begin{pmatrix} O_{[m\times b]} \\ R_{[a\times b]} \end{pmatrix}$ are the new document vectors projected in the expanded term-space having both textual (term–document) components (submatrices $L_{[m\times a]}$ and $O_{[m\times b]}$) and linking (document–document) connectivity components (submatrices $G_{[a\times a]}$ and $R_{[a\times b]}$).

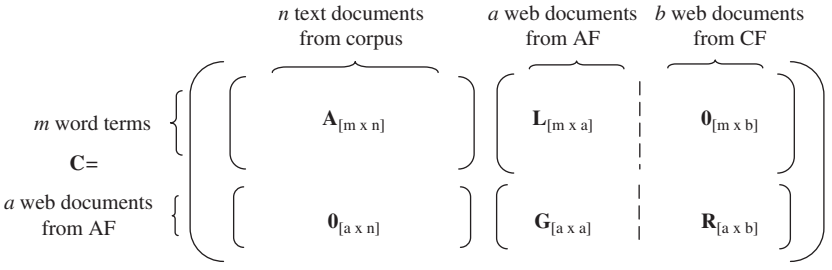

Fig. 4. Expanded connectivity matrix in HCLA. Matrix C is $(m+a)\times(n+a+b)$.

The steps of the proposed method are:

- With a given text-only corpus of m documents and a vocabulary of n terms we first construct offline a text–document matrix $A_{m\times n}$ and perform a truncated SVD $A_k = \mathrm{SVD}(A,k)$.
- Starting the crawling process and after a sufficient user-defined number of pages (a) have been fetched, we analyse the connectivity information of the crawler current web graph and we insert $a = |AF|$ new rows as "terms" (i.e., documents from the AF queue) and $a+b = |AF|+|CF|$ web pages from both AF and CF as "documents" to the matrix. Using the mathematics of Section 2.1.2 we perform the SVD updating technique to avoid the reconstruction of the expanded index matrix. Because the matrices G and R in Fig. 4 are sparse, the SVD updating procedure is simplified and the computation is reduced. In accordance with Section 2.1.2, we want to insert $t = a$ terms and $d = a+b$ documents, meaning that we have to append the submatrix

$$D_{[(m+a)\times(a+b)]} = \begin{pmatrix} L_{[m\times a]} & 0_{[m\times b]} \\ G_{[a\times a]} & R_{[a\times b]} \end{pmatrix} \text{ to } B_{[(m+a)\times n]} = \begin{pmatrix} A_{[m\times n]} \\ 0_{[a\times n]} \end{pmatrix}$$

  which is the new space after inserting terms from the AF.
- Because we do not have any information of direct relationship between any of these web pages and the text documents $\{d_i\}$ of the original corpus, we simply add a terms/rows at the bottom of the matrix A_k with zero elements. This allows the recomputation of SVD with minimum effort, by reconstructing the term–document matrix. If $SVD(A,k) = U_k S_k V_k^{\mathrm{T}}$ is the truncated SVD (k -SVD) of the original text–document matrix A, and $SVD(B) = U_B S_B V_B^{\mathrm{T}}$ the k-SVD of the matrix after having inserted a documents, then we have:

$$U_B = \begin{pmatrix} U_{[m\times k]} \\ 0_{[a\times k]} \end{pmatrix}, \quad S_B = S_k, \quad V_B = V_k. \tag{17}$$

- In order to insert fetched and unvisited documents from the AF and CF queues as columns in the expanded matrix, we use an SVD updating technique to calculate the semantic differences introduced in the column and row space. Matrices U_c , S_c and V_c are calculated using Eq. (10). If we define $SVD(C) = U_C S_C V_C^{\mathrm{T}}$, $F = (S_k | U_B^{\mathrm{T}} D)$, and $SVD(F) = U_F S_F V_F^{\mathrm{T}}$ then, according to [13] :

$$V_C = \begin{pmatrix} V_B & 0 \\ 0 & I_{[a+b]} \end{pmatrix} V_F, \quad S_C = S_F, \quad U_C = U_B V_F. \tag{18}$$

- Accordingly, we project the driving original query q to the new space defined by the expanded connectivity matrix C represents. This is done by simply appending a rows of zeroes to the bottom of the query vector: $q_C = \begin{pmatrix} q_{[m\times 1]} \\ 0_{[a\times 1]} \end{pmatrix}$. By applying the driving query q_C of the test topic, we are able to compute a total ranking of the expanded matrix C. Looking at Fig. 4, we deduce that we only need ranking the last $b = |CF|$ columns. The ranking scores of each document in CF are calculated using the cosine similarity measure of Eq. (6):

$$\cos\theta_j = \frac{e_j^{\mathrm{T}} V_C S_C (U_C^{\mathrm{T}} q_C)}{||S_C V_C^{\mathrm{T}} e_j||_2 ||q_C||_2}. \tag{19}$$

- Once similarity scores are attributed to documents in the frontier, we are able to reorder the priority queue CF, select the most promising candidate and iterate the above steps.

5. Implementation—experimental results—analysis

For the evaluation of the proposed algorithm, two sets of experiments have been carried out in this paper. The first experiment compares HCLA against well-known text- and link-based algorithms on the WebKB corpus while the second one compares HCLA performance against PLSI-based classification on the Cora corpus.

5.1. Experiment on WebKB dataset

In the first experiment, we evaluate six different algorithms. The BRFS was only used as a baseline method since it does not offer any focused resource discovery. The other four algorithms are cases of BSFS with different CF reordering policies. More

specifically, the second algorithm is a simple BL as proposed in [37]. Here, the BL of a web document v in CF is the current number of documents in AF that have v as an outlink.

The third algorithm (SS1) is based on the Shark-Search algorithm [56] a more aggressive variant of Fish-Search [58]. The intuition behind this algorithm is that relevant documents often have relevant neighbours. Unlike the binary relevance metric used by the Fish-Search algorithm, Shark-Search uses a similarity engine which returns a fuzzy score between 0 and 1. The relevance score of each document in AF is measured using pair-wise document similarity that is done by calculating the cosine lexical similarity between its document vector and the topic keywords of the driving query. Then, with a decay factor of $0<d<1$, each unvisited URL v in CF inherits d times the relevance of its parents (nodes that are direct inlinks if v), d^2 times the relevance of its grandparents (nodes with a length-2 path to v) and so on. The fourth algorithm (SS2) is based again on the Shark-Search but this time the relevance scores are calculated using a pretrained VSM that uses the probabilistic ranking scheme of Section 2.1.1. Since we work with an unlabelled text corpus, we use the topic query to extract the most relevant documents and use them as sample examples to train the system.

The fifth algorithm we implement is based on PageRank. According to [37] the crawler visits the candidate pages in the order of Pagerank. Here, no textual information is available at all, only the connectivity between the documents fetched so far and their outlinks. A known problem is that pages in CF do not have known outlinks since they have not been fetched and parsed yet. In order to achieve convergence of the PageRank we assume that from nodes with no outlinks we can jump with probability one to all other pages in the current web graph. In this application, the exact pagerank values $PR(p)$ in Eq. (11) are not as important as the ranking they induce on the pages. This means that we can stop the iterations fairly quickly even when the full convergence has not been attained. In practice we found that no more than 10 iterations were needed. A typical value of 0.8 was selected for the damping factor.

The sixth algorithm is the one this paper proposes. SVD calculations were done using the implicitly restarted Arnoldi/Lanczos bidiagonalisation method. Simple document normalisation was applied to each vector of the expanded matrix in every reordering step [7].

All algorithms have been tested on the WebKB dataset. This corpus has 8275 (after eliminating duplicates) web documents manually classified to 7 categories. 4113 documents are miscellaneous pages collected from other universities while the rest were collected from four universities in 1997 by the World Wide Web Knowledge Base project [59]. Table 1 illustrates the distribution of corpus documents into different categories. In our experiments, the preprocessing of WebKB corpus involved fixing HTML errors, converting text encoding and filtering out all external links (outlinks that are not found inside the WebKB corpus). Also, for algorithms SS1, SS2 and HCLA that needed text similarity measures, we used stemmed contents through an implementation of Porter stemmer [60] and a word stoplist for both the train and term documents. Only the 1000 most frequent terms were considered for training of the VSM. For the algorithms SS1, SS2, HCLA, where text analysis is used, we select each time three universities for training the classifier and the fourth for testing using a leave-one-out technique. Documents from the "misc" university were also used for HCLA, since a larger size of the initial text corpus can enhance the efficiency of LSI, while this can have deteriorating results for the remaining algorithms. The effective dimension of matrix A, used for LSI, is approximately 1000×7000. Although the WebKB documents have link/connectivity information we disregarded this fact in the training phase and chose to treat them only as textual data. Naturally, for the testing phase we did take into account both textual and linking information. The keyword-based queries that drive the crawl were an indicative

Table 1
WebKB Corpus

	Cornell	Misc	Texas	Washington	Wisconsin	Total
Course	44	685	38	77	85	929
Depart	1	178	1	1	1	182
Faculty	34	969	46	31	42	1122
Other	619	692	571	939	942	3763
Project	20	418	20	21	25	504
Staff	21	91	3	10	12	137
Student	128	1080	148	126	156	1638
Total	867	4113	827	1205	1263	8275

description of each category. These were formed by assigning 10 people the task of retrieving relevant documents for each category using Google and recording their queries. The queries are summarised in Table 2. Seeds in each case were considered the root documents in the *department* category. This entails the possibility of some documents being unreachable nodes in the vicinity tree by any path starting with that seed, something that explains the smaller than 100% final recall values in Figs. 5, 7, and 8. Categories having relatively limited number of documents (e.g., *staff*) were not tested. We repeated the experiments for each category and for every university.

Evaluation tests measuring the overall performance of the techniques in Section 2 were performed by calculating the *harvest rate*, a measure that reflects the current health of the focused crawl [34]. Harvest rate is the percentage of the web pages fetched by the crawler that are relevant to a given predicate:

$$H = \frac{N_c}{N_t}, \tag{20}$$

where N_t is the total number of URLs crawled so far and N_c is the number of URLs crawled so far which satisfy the predicate. Fig. 5 depicts the average Harvest-Rate for categories *course*, *faculty*, *project*, and *student* indicating also the 0.95 confidence intervals for the best two algorithms (HCLA and SS2).

The performance evaluation of the six algorithms for the category *student* in the WebKB dataset is done using the recall and precision measures, as demonstrated in Fig. 6. Recall R is the percentage of the total relevant documents in the collection retrieved by the system, while precision P is the percentage of relevant documents in relation to the number of documents retrieved:

$$P = \frac{\text{number of relevant documents retrieved}}{\text{number of documents retrieved}}, \tag{21}$$

$$R = \frac{\text{number of relevant documents retrieved}}{\text{number of relevant documents}}. \tag{22}$$

The overall performance of the system is assessed using the F_1 measure, which is the weighted harmonic mean of precision and recall:

$$F_1 = \frac{2PR}{P+R}. \tag{23}$$

A recall-precision graph for the category *student* is depicted in Fig. 6, while Table 3 demonstrates the precision and F_1-measure statistics at three different recall values (0.1, 0.5, and 1).

It must be stated that due to the complexity of PR and HCLA algorithms we choose to follow an adapted BSFS strategy, applying the CF reordering policy every N documents fetched (and added to AF queue) for all algorithms (except BRFS). We call this a BSFSN[1] strategy. This is supported by the results of [61] which indicates that explorative crawlers outperform their more exploitive counterparts. We experimented with values of $N = 10$, 25, 50 and present representative results in Fig. 7 for category *project* and university *washington*.

The results shown in the harvest rate plot (Fig. 5) depict the superiority of our method especially at high recall range. We must also consider that in our implementation we did not use a sophisticated term weighting scheme such as Okapi [62], which is argued to boost the performance of LSI [63]. BRFS performance was surprisingly good, matching or exceeding in some cases SS1 and BL. This can be attributed to the hypertext structure of the WebKB corpus and the quality of the seed documents. On the other hand, the unimpressive, considering its complexity, results of PageRank justify the argument that it is too general for use in topic-driven tasks due to its minimal exploitation of the topic context [50].

What is not depicted in Fig. 5 is the fact that while HCLA and PR methods may yield good results, they require considerably more processor power and memory resources and thus they are significantly slower. The dynamic nature of the crawler means that computational complexity

Table 2
WebKB Corpus topic queries

Category	Topic keywords
Course	Course, university, homework, lesson, exams, assignment, lecture, tutorial, book, schedule, notes, grading, handout, teaching, solutions
Faculty	Faculty, university, professor, publications, papers, research, office
Project	Project, university, demonstration, objective, overview, research, laboratory
Student	Student, university, interests, favourite, activities, graduate, home

[1]BSFS where CF is reorganised every N documents downloaded and added to AF.

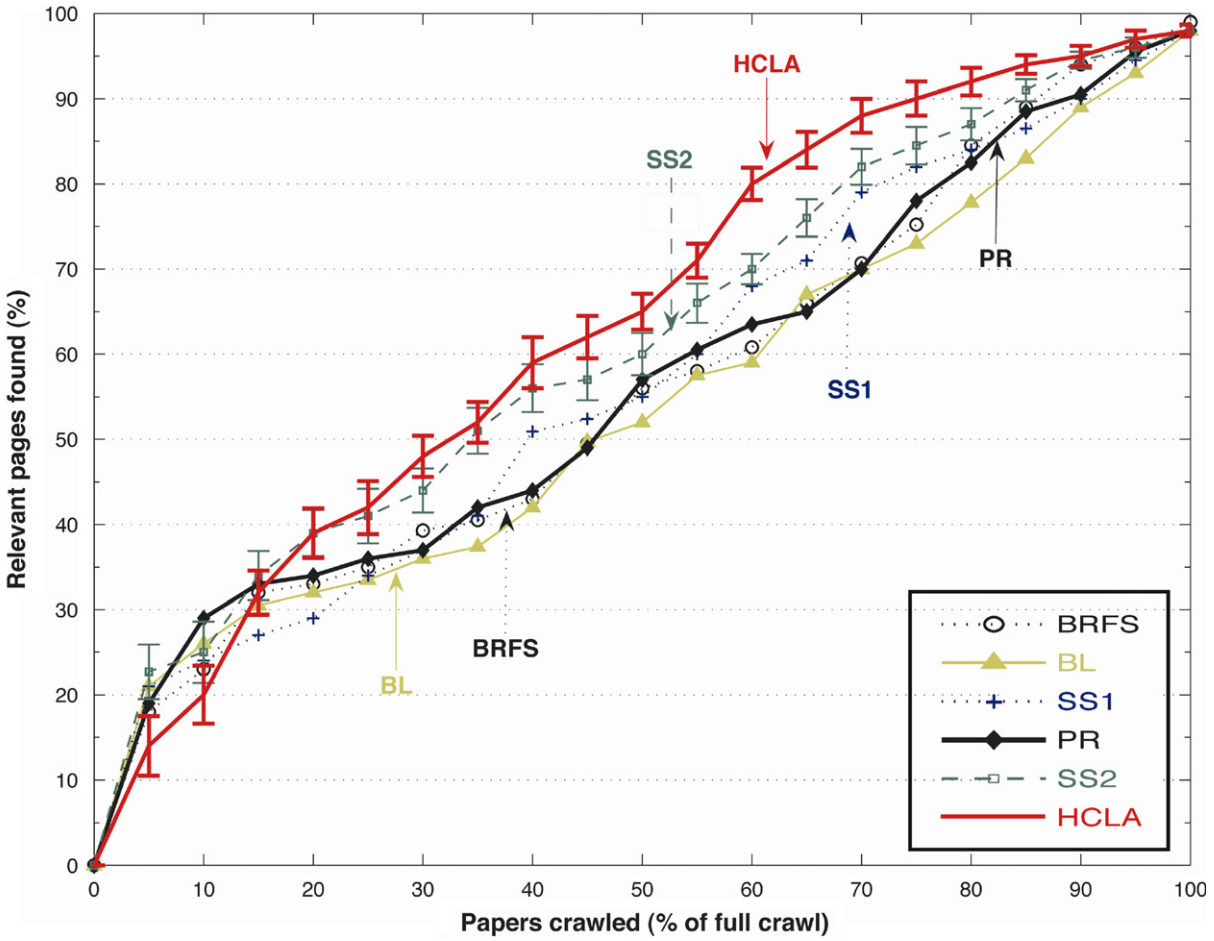

Fig. 5. Average harvest rate for WebKB.

increases as more documents are inserted in AF and CF queues. An obvious solution to the problem is to limit the size of both queues and discard less authoritative or relevant documents at the bottom of the queues during the reordering phase.

In a BSFS focused crawler it is crucial that the time needed for reorganising the CF is kept at a minimum. It is pointless to perform sophisticated but time-costly analysis when at the same interval we could have simply downloaded all the documents in the queue. According to [64], the best algorithms for SVD computation of a $m \times n$ matrix take time that is proportional to $O(\lambda m^2 n + \lambda' n^3)$ (λ and λ' are constants which are 4 and 22 for a Riemannian SVD algorithm (R-SVD)). If there is only need for the set of singular values and not the U and V matrices the above is reduced to $O(mn^2)$. This hinders the performance of a LSI-based BSFS crawler since new documents and terms are inserted in each iteration and both values of m and n increase. Of course, in our work, we do not need to recompute the SVD of the highly dimensional matrix C, but perform SVD for the reduced matrices H and F in Eqs. (8) and (10). Moreover, we follow a BSFSN algorithm where the reordering of the CF, and consequently the term–document matrix expansion and SVD computation, are performed every N documents are fetched. Naturally, the larger value of N has a significant influence in the processing time of the algorithm but can harm the efficiency of the reordering analysis [61]. For the results presented here it is $N = 50$. As we can deduce from Fig. 7, reordering the CF in higher frequency (i.e., lower values of N) does not necessarily yield better results.

As we have seen in Section 4, the LSI analysis for the original text corpus is done offline during the training phase. This permits us to determine an

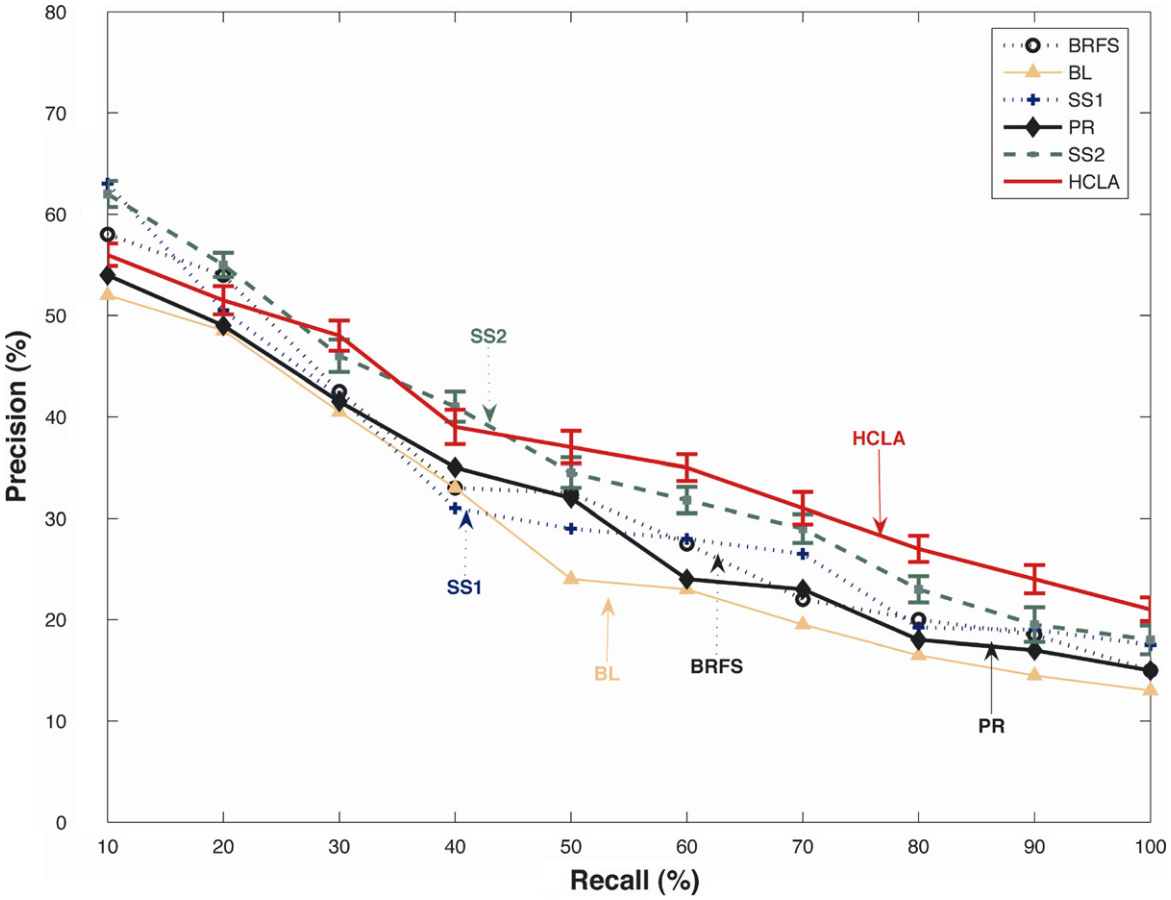

Fig. 6. Recall-precision graph for category *student* in WebKB dataset.

Table 3
Comparison of precision and F_1 measure for different algorithms using the category *student* in WebKB

Algorithm	Precision			F_1 measure		
	$R = 0.1$	$R = 0.5$	$R = 1.0$	$R = 0.1$	$R = 0.5$	$R = 1.0$
BRFS	0.58	0.32	0.15	0.17	0.39	0.26
BL	0.52	0.24	0.13	0.17	0.32	0.23
SS1	0.63	0.29	0.17	0.17	0.37	0.30
SS2	0.62	0.34	0.18	0.18	0.41	0.31
PR	0.54	0.32	0.15	0.16	0.39	0.26
HCLA	0.56	0.37	0.21	0.18	0.43	0.35

Values are at recall rates $R = 0.1$, 0.5, and 1.0, respectively.

optimum value for the number of the important factors k in term–document matrix A by using the F_1 measure but on the basis of assuming an a priori document classification knowledge of the dataset. Nevertheless, for most cases, the optimal k in corpora of a few thousand documents is typically between 10 and 100. A recent study on LSI performance is given in [65]. Selecting a small value for k has important benefits for computational cost. For our work, we found that choosing $k = 50$ for the LSI of the text corpus (matrix A) in the training phase of HCLA, yields the best results. In Fig. 8 we see that selecting too many features ($k = 80$) can have in fact deteriorating results.

5.2. Experiment on Cora dataset

For the comparisons between HCLA and PLSI, we chose the Cora dataset [66]. This consists of over 37 000 academic papers, automatically classified into hierarchical categories. Cora has a higher link density and higher quality information than the WebKB collection. External links (outlinks not existing in the collection) account only for 20% of the collection while this is over 60% for WebKB [67]. Intra-class links percentage is higher in Cora, too. Also, academics papers in Cora have more quality text content than web pages in WebKB.

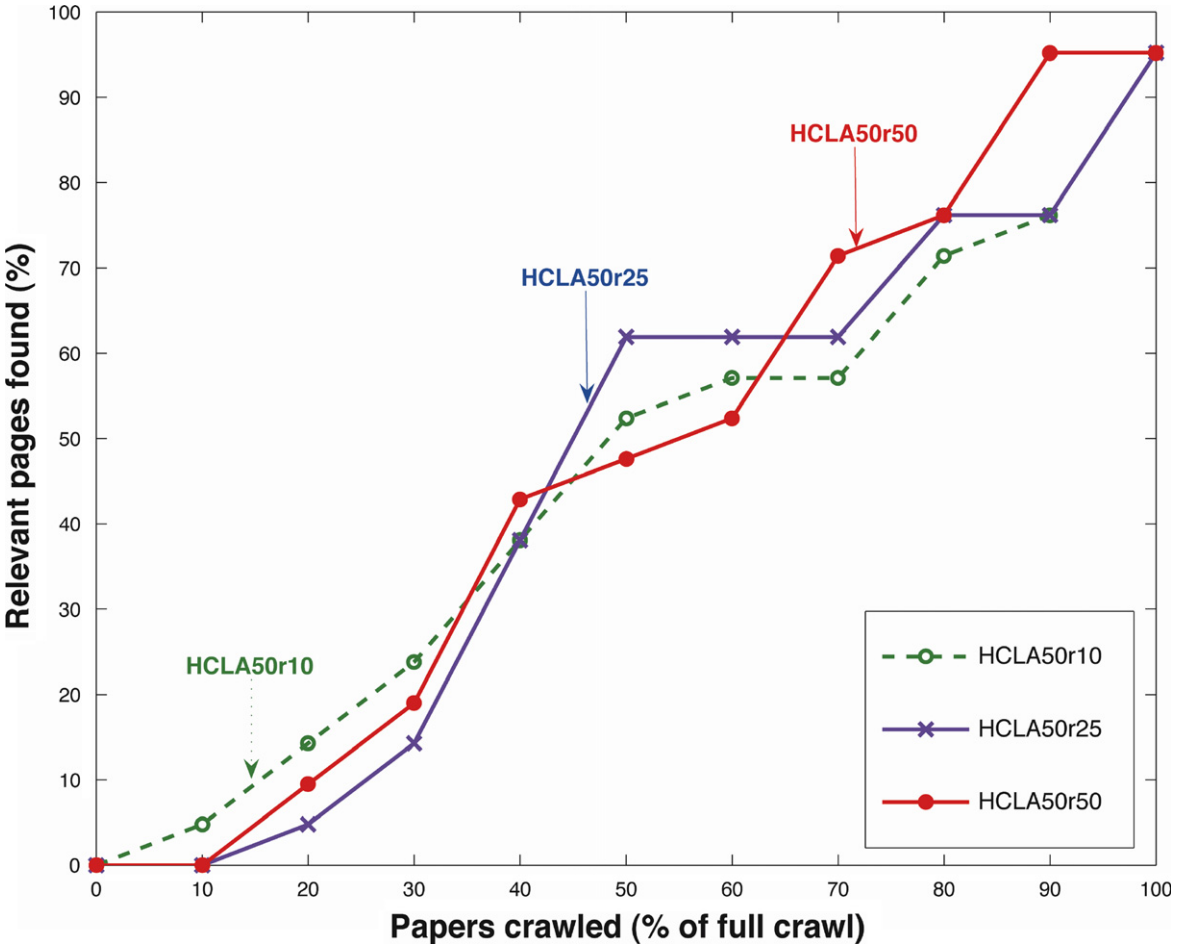

Fig. 7. Harvest rate for category *project* and university *washington* in WebKB, for different reordering frequencies (N) in BSFSN strategy. HCLA50r10 stands for the algorithm HCLA, with $k = 50$ features selected for LSI analysis and reordering the crawl frontier every $N = 10$ documents.

The same document preprocessing rules and algorithm parameters with the first experiment were applied here too. This time we considered Cora reference contexts as analogous to anchor text. Frequent stemmed terms from the abstracts, the affiliation and paper title were used to train the VSM and formulate the driving queries. In our experiments we followed a BSFSN ($N = 25$) visiting policy for both algorithms. For PLSI, we chose default values for parameters a and η and b ($a = 0.5$, initial value of $b = 1$, lower limit $b = 0.8$, update parameter $\eta = 0.9$). Using a leave-one-out technique, we averaged results over all possible test/train splits of *Information_Retrieval* category and its subcategories *Digital_Library*, *Extraction*, *Filtering* and *Retrieval* (see Table 4). Additional unlabelled documents were added randomly from other categories as needed, so that the training matrix A used for LSI is approximately 1000×7000, matching the dimensions of A in the WebKB experiment.

Table 5 demonstrates the precision and F_1-measure statistics for the two algorithms at three different recall values (0.1, 0.5, and 1). The overall evaluation of efficiency of the two algorithms is depicted in the average recall-precision graph in Fig. 9, where the confidence intervals are at 0.95. We deduce that HCLA offers higher performance in high recall rates than PLSI + PHITS. From additional undocumented experiments, where anchor text was not used, performance, depending on the available connectivity information, was marginally worse than PLSI + PHITS (under 5% at most recall levels) but still, comparable.

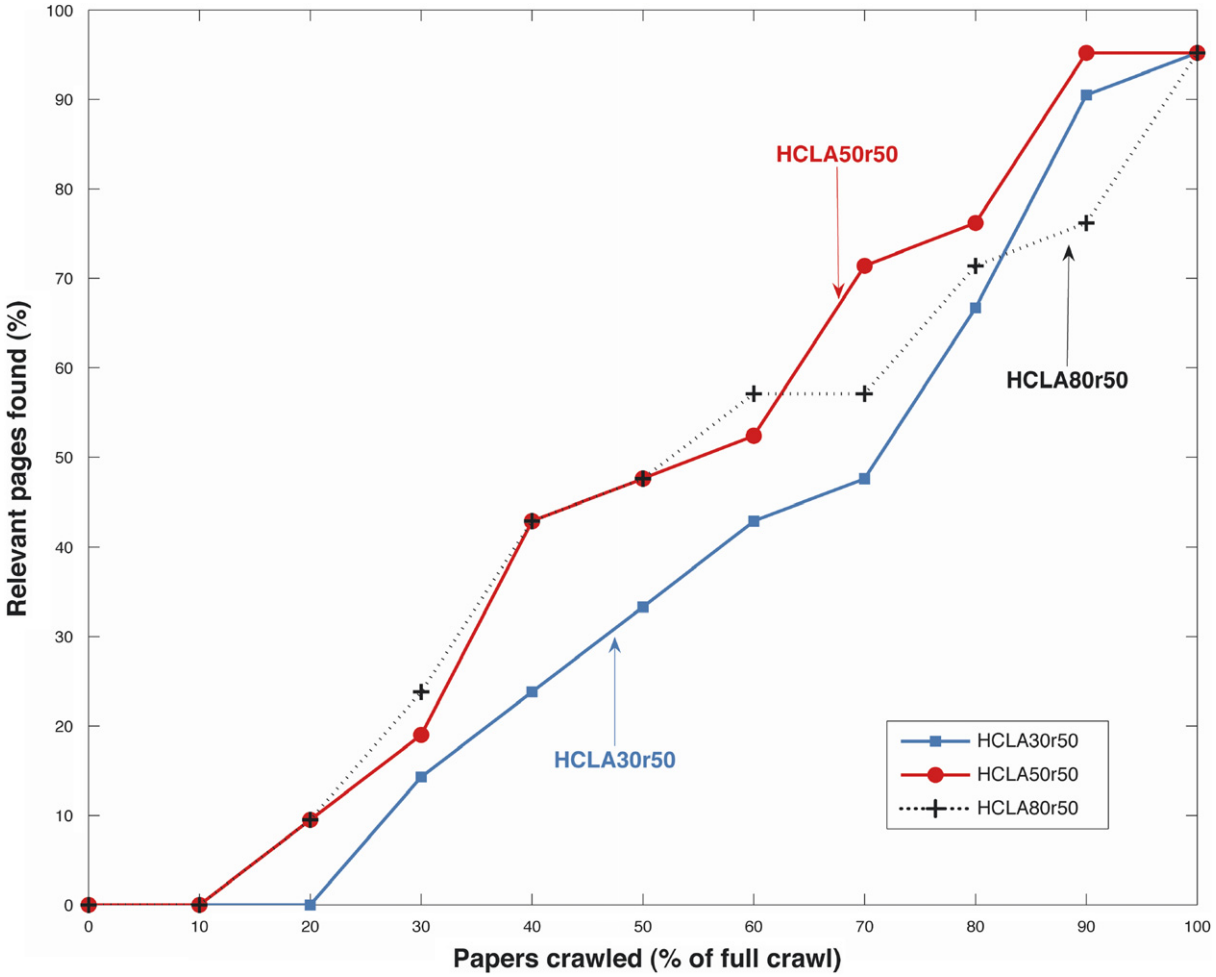

Fig. 8. Harvest rate for category *project* and university *washington* in WebKB, for different number of important factors (k). HCLA30r50 stands for the algorithm HCLA, with $k = 30$ features selected for LSI analysis and reordering the crawl frontier every $N = 50$ documents.

Table 4
Sub-categories of *Information_Retrieval* in Cora dataset

Category	# Documents
Digital_Library	133
Extraction	77
Filtering	57
Retrieval	315
Information_Retrieval (Total)	582

Table 5
Comparison of precision and F_1 measure for HCLA and PLSI + PHITS algorithms using the category *Information_Retrieval* in Cora dataset

Algorithm	Precision			F_1 measure		
	$R = 0.1$	$R = 0.5$	$R = 1.0$	$R = 0.1$	$R = 0.5$	$R = 1.0$
HCLA	0.88	0.72	0.54	0.18	0.59	0.70
PLSI + PHITS	0.86	0.74	0.47	0.18	0.60	0.64

Values are at recall rates $R = 0.1$, 0.5, and 1, respectively.

6. Conclusions

This work has been concerned with a statistical approach to text and link processing. This ignores any understanding of language semantics, but it has historically proved efficient for dealing with a topic-oriented problem such as a vertical search engine design rather than a knowledge-based logic representation system and moreover it requires less human effort. We argue that content- and link-based features can be used for both the classifier and the distiller of a focused crawler. As expected, sophisticated algorithms such as HCLA, PR, and

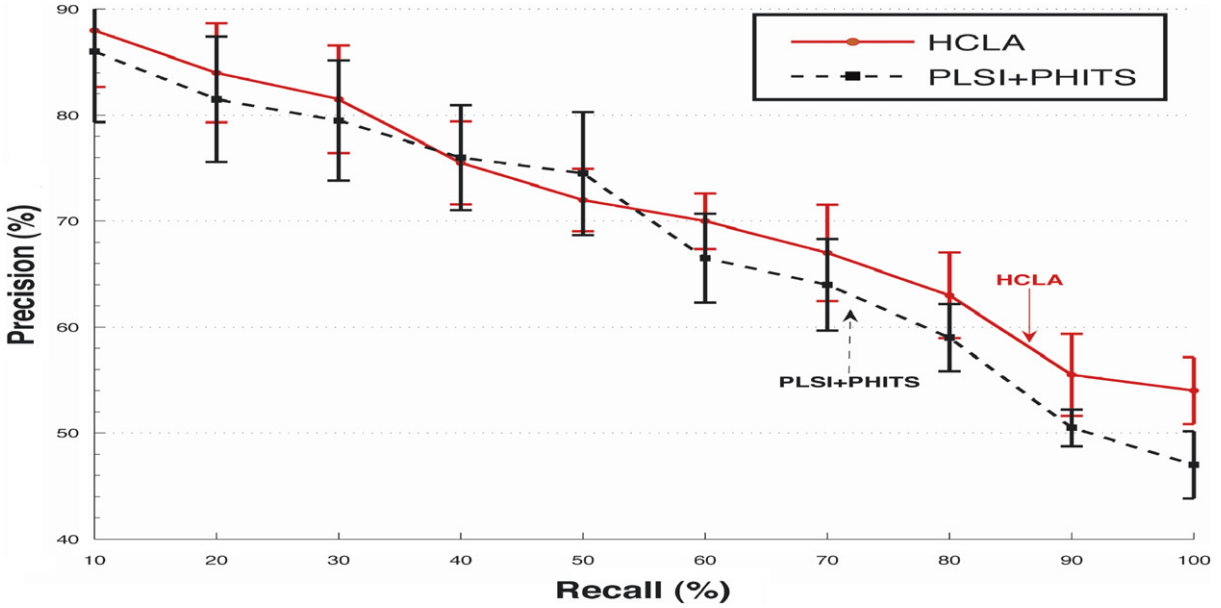

Fig. 9. Recall-precision graph for category *Information_Retrieval* in Cora dataset.

SS2 generally yield better results than the simplistic BL and BRFS algorithms. We have also demonstrated that HCLA performance is comparable and in many cases better than PLSI + PHITS. The question remains whether the extra performance gain justifies the complexity of the development of an HCLA-based focused web crawler. Both HCLA and PR methods proved significantly slower requiring more processor power and memory resources. More specifically, HCLA was up to 100 times slower than the simple BRFS on some tests and PR performed similarly, something that has been attested by [35].

A positive point in our method is that it does not have any dependence on historical information such as a previous crawl and index of the web, or an existing generalised search service. It must also be mentioned that the text corpus used to train the classifier was considered unlabeled reducing the problem to an unsupervised machine learning one. Complete knowledge of the text topic taxonomy does not invalidate our work since the proposed document representation can be retained. The extra knowledge can be used as a weighting scheme for the documents in the expanded matrix or simply can replace the role of the driving queries. In the latter case, documents of the crawling frontier do not need to be inserted in the term–document matrix A, thus reducing the level of complexity of SVD updating. Instead, they are considered as queries and their relevance to the topic can be calculated using, for example, a noise-or measure [57]. The individual similarity cosine distances (19) from n indexed documents of each category against a query document of the CF are combined to provide a final score for each category, $S_{LSI} = 1 - \prod_{i=1}^{n}(1 - \cos\theta_n)$.

LSI performance is sensitive to the size of the trained corpus and can suffer severely when little data are available. Therefore, starting a crawl with a small term–document matrix A is not recommended since at early stages the extra linking-text information from the crawl is minimal. This can be alleviated by using unlabelled data and other forms of available background text as well as labelled training data in the classification process. This *background knowledge* assists the modelling of LSI and offers significant performance gains [57]. While the quality of the training data is a key aspect of the system, having sufficient but noisy text information is preferred to having highly relevant but few training documents. In our case, we append extra text documents in the training phase that can even have less relevance with the topics of the current corpus, for instance documents from *misc* university or *other* category. At later stages when more information is available to the system these documents can be removed and the model can be retrained.

There are a number of limitations of this work that we still have to address. Link-based algorithms, such as HCLA, may suffer at early stages of the retrieval process. This is attributed to the fact that at early stages not sufficient link information is available and the density of the current graph is rather sparse. We believe that a hybrid queue link-extraction strategy where HCLA is facilitated in the early stages of the crawl by a more explorative algorithm should yield better results. Also, it is self-evident that by using additional text information such as reference contexts and anchor text we can further improve the performance of the algorithm. However in that case, the $O_{m \times b}$ submatrix would be non-zero meaning that the computational cost of SVD analysis would increase.

A parameter not well documented is the choice of k (number of important factors) in LSI. While offline experiments can reveal an optimum initial value for the text corpus (matrix A), there is no guarantee this will remain optimal for the expanded matrix C, especially since the latter increases dynamically. An idea worth exploring in the future is an adaptive scheme that can support the dynamic information increase as the crawling process evolves. As the collection grows, there is more textual evidence and a richer graph structure, which means that the number of important factors (k) and the "frequency" of CF reranking (N) need to increase accordingly.

As stated earlier (Section 4), some theoretical assumptions of the VSM and LSI have been relaxed, mainly the document representation. Extra work identifying the implications of orthogonality in the new space needs to be done in the future. Another important restriction is the weighting scheme of the expanded adjacency matrix. Moreover, the existence of queues in the crawling process means that there is always the danger of stagnation. Davison [68] also proposes an expanded adjacency matrix that allows for different weighting schemes in different directions and explores the use of eigen-analysis in the augmented matrix. There, not only term–document similarity is modelled but also term–term and document–document. It will be interesting to apply the assumption of word-link semantic equivalence in the representation of web documents in that proposal. As a first step, we can expand the original term–document matrix $A_{m \times n}$ during training by considering the documents as terms, i.e., add n rows to the bottom of A. In the new column vector space, a document is represented as a bag of both terms and citations (outlinks). The significance of this representation will be realised when we there is previous knowledge available on link connectivity between documents, for example, when deploying an incremental crawler. This can lead to a semantically richer query definition. Another generalised way of viewing data objects and their intra- and inter-relationships from multiple and heterogeneous sources has been studied in [69]. Using the Unified Relationship Matrix, their algorithm (SimFusion) demonstrates improved similarity measurements of web objects over traditional content-based algorithms.

It should be stated that the size of the experiments in this work is limited for the web setting. Extra work on a larger scale, against other algorithms is a challenge for future work. It would be interesting to compare HCLA against link-based strategies that use historical information from a previous crawl or that can query an "oracle" which knows the complete web graph and has calculated the actual Pagerank of each page (*Omniscient* strategy) [70].

Information discovery on the web is a challenging task with a great potential that is yet to be realised. The diversity and complexity of web information, as well as its richness, call for approaches that go beyond conventional IR. One such approach is to leverage and combine various information sources on the web. More specifically, investigation of combining content, hyperlinks, and other features of web documents with human categorisation and classification-based techniques would be an interesting and valuable endeavour to undertake. Proceeding so, this work has explored the combination of link analysis and content analysis in order to improve retrieval performance.

Acknowledgements

G. Almpanidis was granted a basic research fellowship co-funded by the European Union and the Hellenic Ministry of Education under the framework of the programme "HERAKLEITOS" of the Operational programme for Education and Initial Vocational Training within the 3rd Community Support Framework.

We would like to thank Mr. Athanasios Papaioannou for his contributions in the implementation of the PLSI algorithm.

References

[1] Google Search Technology. Online at ⟨http://www.google.-com/technology/index.html⟩.

[2] D. Sullivan, The vortals are coming! The vortals are coming! From The Search Engine Report. Online at 〈http://searchenginewatch.com/sereport/00/04-vortals.html〉.
[3] D. Sullivan, Now, it's the "vectories" that are coming!. From The Search Engine Report. Online at 〈http://searchenginewatch.com/sereport/00/08-vectories.html〉.
[4] R. Steele, Techniques for specialized search engines, in: Proceedings of the Internet Computing '01, Las Vegas, June 25–28, 2001.
[5] K. Rijsbergen, Information Retrieval. Online 〈http://www.dcs.gla.ac.uk/Keith/Preface.html〉.
[6] B. Davison, Topical locality in the Web, in: Proceedings of the 23rd ACM International Conference on Research and Development in Information Retrieval (SIGIR 2000), Athens, Greece, 2000, pp. 272–279.
[7] G. Salton, A. Wong, C.S. Yang, A vector space model for automatic indexing, Commun. ACM 18 (11) (1975) 613–620.
[8] K.S. Jones, Search term relevance weighting given little relevance information, J. Doc. 35 (1) (1979) 30–48.
[9] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, R. Harshman, Indexing by latent semantic analysis, J. Am. Soc. Inform. Sci. 41 (1990) 391–407.
[10] P. Foltz, Improving human-proceedings interaction: indexing the CHI index, in: Proceedings of the Conference on Human Factors in Computing Systems, Denver, 1995, pp. 101–102.
[11] M. Berry, S. Dumais, G. O' Brien, Using linear algebra for intelligent information retrieval, SIAM Rev. 37 (4) (1995) 573–595.
[12] M. Berry, M. Browne, Understanding Search Engines: Mathematical Modeling and Text Retrieval, Society of Industrial and Applied Mathematics, Philadelphia, PA, 1999.
[13] G. O'Brien, Information management tools for updating an SVD-encoded indexing scheme, Master's thesis, University of Tennessee, Knoxville, TN, 1994.
[14] S. Wasserman, K. Faust, Social Network Analysis, University Press, Cambridge, UK, 1994.
[15] M. Kessler, Bibliographic coupling between scientific papers, Am. Doc. 14 (1963) 10–25.
[16] H. Small, Co-citation in scientific literature: a new measure of the relationship between two documents, J. Am. Soc. Inform. Sci. 24 (1973) 265–269.
[17] E. Garfield (Ed.), Journal impact vs. influence: a need for five-year impact factors—Letter to the Editor, Inform. Process. Manage. 22 (5) (1986) 445.
[18] A. Ng, A. Zheng, M. Jordan, Stable algorithms for link analysis, in: Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR 2001), 2001, pp. 258–266.
[19] K. Yang, Combining text- and link-based methods for Web IR, in: Proceedings of the 10th Text Rerieval Conference (TREC-10), US Government Printing Office, Washington, DC, 2002.
[20] S. Haas, E. Grams, Page and link classifications: connecting diverse resources, in: Proceedings of the Third ACM Conference, Digital libraries (DL 1998), Pittsburgh, 1998, pp. 99–107.
[21] CLEVER Project. IBM Almaden Research Center, Online at 〈http://www.almaden.ibm.com/projects/clever.shtml〉.
[22] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, WWW7/Computer Networks 30 (1–7) (1998) 107–117.
[23] J. Kleinberg, Authoritative sources in a hyperlinked environment, in: Proceedings of the Ninth Annual ACM-SIAM Symposium, Discrete Algorithms, January 1998, pp. 668–677.
[24] M. Henzinger, Link analysis in web information retrieval, IEEE Data Eng. Bull. 23 (3) (2000) 3–8.
[25] K. Bharat, M. Henzinger, Improved algorithms for topic distillation in hyperlinked environments, in: Proceedings of the ACM International Conference on Research and Development Information Retrieval (SIGIR 1998), Melbourne (Australia), August 1998, pp. 104–111.
[26] D. Cohn, T. Hoffman, The missing link—a probabilistic model of document content and hypertext connectivity, in: Advances in Neural Information Processing Systems, vol. 13, MIT Press, Boston, MA, 2001, pp. 430–436.
[27] D. Cohn, H. Chang, Learning to probabilistically identify authoritative documents. In: Proceedings of the 17th International Conference on Machine Learning, Stanford University, 2000, pp. 167–174.
[28] R. Baeza-Yates, Modern Information Retrieval, ACM Press Series/Addison Wesley, New York, 1999.
[29] A. Gulli, A. Signorini, The indexable Web is more than 11.5 billion pages, in: Proceedings of the 14th International Conference on WWW (WWW05), 2005, pp. 902–903.
[30] J. Hynek, K. Jezek, Document classification using itemsets, in: Proceedings of the 34th International Conference on MOSIS 2000, pp. 97–102.
[31] D. Zeinalipou-Yatzi, M. Dikaiakos, High-performance crawling and filtering in Java, in: Proceedings of the Eighth Panhellenic Conference on Informatics, vol. 2, November 2001, pp. 377–386.
[32] The Web Robots FAQ. Online at 〈http://www.robotstxt.org/wc/faq.html〉.
[33] M. Najork, J. Wiener, Breadth-first search crawling yields high-quality pages, in: Proceedings of the 10th International Conference on World Wide Web (WWW01), 2001, pp. 114–118.
[34] S. Chakrabarti, M. Berg, B. Dom, Focused crawling: a new approach to topic-specific Web resource discovery, Comput. Networks 31 (1999) 1623–1640.
[35] M. Chau, H. Chen, Comparison of three vertical search spiders, IEEE Comput. Mag. 36 (5) (2003) 55–62.
[36] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, S. Raghavan, Searching the web, ACM Trans. Internet Technol. (1) (2001) 2–43.
[37] J. Cho, H.G. Molina, L. Page, Efficient crawling through URL ordering. In: Proceedings of the Seventh International Conference on World Wide Web (WWW98), Brisbane, Australia, 1998, pp. 161–172.
[38] P. Srinivasan, G. Pant, F. Menczer, Target seeking crawlers and their topical performance. In: Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR 2002), August 2002.
[39] T. Haveliwala, Topic-sensitive PageRank, in: Proceedings of the 11th International Conference on World Wide Web (WWW02), Honolulu, Hawaii, May 2002, pp. 517–526.
[40] R. Baeza-Yates, C. Castillo, M. Marin, A. Rodriguez, Crawling a country: better strategies than breadth-first for web page ordering, in: Proceedings of the International

Conference on World Wide Web (WWW05), Chiba, Japan, 2005, pp. 864–872.
[41] P. Boldi, M. Santini, S. Vigna, Do your worst to make the best: paradoxical effects in PageRank incremental computations, in: Proceedings of the Algorithms and Models for the Web-Graph: Third International Workshop (WAW 2004), Rome, Italy, October 2004, pp. 168–180.
[42] M. Diligenti, F. Coetzee, S. Lawrence, C.L. Giles, M. Gori, Focused crawling using context graphs, in: Proceedings of the 26th International Conference on Very Large Databases (VLDB 2000), Cairo, Egypt, 2000, pp. 527–534.
[43] J. Rennie, A. McCallum, Using reinforcement learning to spider the web efficiently, in: Proceedings of the 16th International Conference on Machine Learning (ICML99), 1999, pp. 335–343.
[44] S. Chakrabarti, Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction, in: Proceedings of the 10th International Conference on World Wide Web (WWW10), Hong Kong, 2001, pp. 211–220.
[45] C. Aggarwal, F. Al-Garawi, P. Yu, Intelligent crawling on the world wide web with arbitrary predicates, in: Proceedings of the 10th International World Wide Web Conference (WWW10), Hong Kong, 2001, pp. 96–105.
[46] S. Sizov, M. Theobald, S. Siersdorfer, G. Weikum, J. Graupmann, M. Biwer, P. Zimmer, The BINGO! system for information portal generation and expert web search, in: Proceedings of the First Conference on Innovative Data Systems Research (CIDR), 2003.
[47] P. Boldi, B. Codenotti, M. Santini, S. Vigna. Ubicrawler, A scalable fully distributed web crawler, Software: Pract. Experience 34 (8) (2004) 711–726.
[48] I. Varlamis, M. Vazirgiannis, M. Halkidi, B. Nguyen, THESUS: effective thematic selection and organization of web document collections based on link semantics, IEEE Trans. Knowledge Data Eng. 16 (6) (2004) 585–600.
[49] D. Bergmark, C. Lagoze, A. Sbityakov, Focused crawls, tunneling, and digital libraries, in: Proceedings of the Sixth European Conference on Research and Advanced Technology for Digital Libraries, 2002, pp. 91–106.
[50] F. Menczer, G. Pant, M. Ruiz, P. Srinivasan, Evaluating topic-driven web crawlers, in: Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR 2001), New Orleans, 2001, pp. 241–249.
[51] F. Menczer, ARACHNID: adaptive retrieval agents choosing heuristic neighborhoods for information discovery, in: Proceedings of the 14th International Conference on Machine Learning, 1997, pp. 227–235.
[52] Text REtrieval Conference (TREC). Online at ⟨http://www.trec.nist.gov⟩.
[53] D. Lewandowski, H. Wahlig, G. Meyer-Beautor, The freshness of Web search engines' databases, Inform. Sci. 32 (2) (2006) 131–148.
[54] J. Leskovec, J. Kleinberg, C. Faloutsos, Graphs over time: densification laws, shrinking diameters and possible explanations, in: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD05), 2005, pp. 177–187.
[55] S. Chakrabarti, Mining the Web: Discovering Knowledge from Hypertext Data, Morgan Kaufmann Publishers, San Francisco, CA, 2002.
[56] M. Hersovici, M. Jacovi, Y.S. Maarek, D. Pelleg, M. Shtalhaim, S. Ur, The shark-search algorithm. An application: tailored Web site mapping, Comput. Networks ISDN Syst. 30 (1998) 317–326.
[57] S. Zelikovitz, H. Hirsh, Improving text classification with LSI using background knowledge, Workshop Notes Text Learning: Beyond Supervision (IJCAI01).
[58] P. De Bra, R. Post, Information retrieval in the world-wide web: making client-based searching feasible, J. Comput. Networks ISDN Syst. (27) (1994) 183–192.
[59] CMU World Wide Knowledge Base and WebKB dataset. Online at ⟨http://www-2.cs.cmu.edu/~webkb/⟩.
[60] M. Porter, An algorithm for suffix stripping, Program 14 (3) (1980) 130–137.
[61] G. Pant, P. Srinivasan, F. Menczer, Exploration versus exploitation in topic driven crawlers, in: Proceedings of the Second International Workshop Web Dynamics, Honolulu, May, 2002.
[62] S. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, M. Lau, Okapi at TREC, in: Proceedings of the First Text REtrieval Conference (TREC-1), Gaitherburg, Maryland, 1992, pp. 21–30.
[63] C. Tang, S. Dwarkadas, Z. Xu, On scaling latent semantic indexing for large peer-to-peer systems, in: Proceedings of the 27th ACM International Conference on Research and Development in Information Retrieval (SIGIR 2004), Sheffield, UK, July 25–29, 2004, pp. 112–121.
[64] G. Golub, C. Van Loan, Matrix Computations, third ed., Johns Hopkins University Press, Baltimore, 1996.
[65] A. Kontostathis, W. Pottenger, A framework for understanding LSI performance, Inform. Process. Manage. 42 (1) (2006) 56–73.
[66] A. McCallum, K. Nigam, J. Rennie, K. Seymore, Automating the construction of internet portals with machine learning, Inform. Retrieval 3 (2000) 127–163.
[67] M. Fisher, R. Everson, When are links useful? Experiments in text classification, in: Proceedings of the 25th European Conference on Advances Information Retrieval (ECIR), Pisa, April 2003, pp. 41–56.
[68] B. Davison, Unifying text and link analysis, in: Proceedings of the Workshop Text-Mining & Link-Analysis (TextLink) (IJCAI), Acapulco, August 9, 2003.
[69] W. Xi, E. Fox, W. Fan, B. Zhang, Z. Chen, J. Yan, D. Zhuang, SimFusion: measuring similarity using unified relationship matrix, in: Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR 2005), 2005, pp. 130–137.
[70] R. Baeza-Yates, C. Castillo, M. Marin, A. Rodriguez, Crawling a country: better strategies than breadth-first for web page ordering, in: Proceedings of the 14th International Conference on World Wide Web (WWW 05), 2005, pp. 864–872.