Investigating the effectiveness of client-side search/browse without a network connection

Hussein Suleman

University of Cape Town, South Africa
hussein@cs.uct.ac.za
http://dl.cs.uct.ac.za/

Abstract. Search and browse, incorporating elements of information retrieval and database operations, are core services in most digital repository toolkits. These are often implemented using a server-side index, such as that produced by Apache SOLR. However, sometimes a small collection needs to be static and portable, or stored client-side. It is proposed that, in these instances, browser-based search and browse is possible, using standard facilities within the browser. This was implemented and evaluated for varying behaviours and collection sizes. The results show that it was possible to achieve fast performance for typical queries on small- to medium-sized collections.

Keywords: search · browse · client-side · offline · browser-based.

1 Introduction

Digital library systems are often created by using a toolkit for this purpose, with many popular toolkits currently available as open source software [7]. DSpace [8] is a popular example of a toolkit that is often used for institutional repositories while AtoM [12] is a popular example of a toolkit that is often used for heritage collection management. Many users are satisfied with the functionality provided by such toolkits and need look no further for their repository systems.

However, there are particular circumstances where these systems do not work well. Firstly, in some parts of the world, network access is not as reliable as it is elsewhere, so purely online systems will be inaccessible to some potential users (e.g., judges in remote parts of African countries, trying to access case law on tablets). Secondly, most digital repository tools require technical expertise for installation and configuration and some users do not have this expertise or access to experts or the resources to hire expertise (e.g., historians in most South African universities). Thirdly, in low-resource environments (such as most poor countries), there is a higher risk with more complex software systems as long-term maintenance of systems and data collections is more difficult with less funding and few experts. Fourthly, not everyone needs DSpace - it is clearly overkill to install DSpace to manage your personal photo album.

The Five Hundred Year Archive (FHYA) [1] is a project attempting to create an exemplar of material that relates to pre-colonial history in Southern Africa.

The number of resources contemplated is of the order of 10000, and a prototype AtoM system was configured for this. The project had some difficulty in finding ongoing developer support as its focus was the data and not the continuing maintenance of software systems. Arguably, many owners of small archives face similar problems, as their collections and resources are not large enough to justify investment in typical repository systems.

This raised the question of whether or not a simpler archiving tool would suffice for such projects. The development of such a prototype simple archiving tool began with the central search and browse interface as an experiment to determine if this could be done without the need for a server at all, using all resources hosted locally and using only the browser for all computation. An earlier project [9] included a simple HTML search using Javascript, but performance was not evaluated and the functionality was restricted to Web pages.

This paper reports on the development and evaluation of a general-purpose offline client-side metadata search and browse system, also referred to as a faceted search system. The research focus was on feasibility and efficiency. From a feasibility perspective, the offline tool worked correctly for a small amount of FHYA test data (approximately 100 items), but it was not known how well this would work for larger realistic-sized collections. The key research question therefore addressed in this paper is: how scalable is this alternative off-line client-side architecture for search and browse systems?

2 System Design

The search implementation, using an extended Boolean model, was designed in accordance with typical information retrieval principles, as described in Managing Gigabytes [14], with browse operations supported through the addition of database-style indices of all items matching a given term.

Thus, there are 2 sets of indices, both stored as XML documents so they can be processed using built-in browser facilities. The search index for the term "study" will contain a listing of identifiers of all items that include the term. The browse index for the field "date" with a value of "1977" will contain a listing of identifiers of all items where the date field has the value "1977". In both cases, titles for documents are included to support fast single queries without a separate title listing. Table 1 shows a snippet of a typical browse index. Search indices are similar, with the inclusion of a weight for each document.

The search system was comprised of 2 applications:

- **create_index.pl** is a Perl script to build the indices required for search/browse operations. All metadata is read into memory and processed using various data structures to create inverted indices as well as lists of items for each browsable field.
- **search.js** is a Javascript script to execute a search/browse operation and display results within the browser window. The query is extracted from the browser's search form. The necessary search and browse indices are then

Table 1. Snippet of typical browse index

```
<index>
<bif id="571" file="32000/record303578.xml" title="The wages-prices spiral
 : a study in distributive shares"/>
<bif id="26332" file="32000/record215502.xml" title="Cerebral palsy"/>
<bif id="1423" file="32000/record212306.xml" title="The waveforms of
atmospherics and the propagation of very low frequency radio waves"/>
<bif id="21682" file="32000/record480241.xml" title="The attitude of the
Tractarians to the Roman Catholic Church, 1833-1850"/>
...
```

loaded and processed to generate a list of results in order of probable relevance and filtered by the specified browse fields.

The system is configurable in that fields for searching and browsing can be specified in a configuration file. Nominally, search operations collapse all fields into a single unstructured piece of text during indexing. The system also includes the ability to specify/index individual fields and subsets of individual fields, allowing queries such as "name:Jak" where the source metadata might have fields named "creator" and "contributor". This field-based search was not evaluated in this paper as the specification of a field simply creates a separate index for that field and this will not affect performance differently from non-fielded queries.

Figure 1 shows a screen snapshot of the search/browse interface used for these experiments.

3 Evaluation

A multi-factor design was adopted to evaluate the performance of the system, considering: size of collection; complexity of search query; complexity of browse query; and variability in queries.

3.1 Dataset

A dataset with human-assigned metadata fields covering a wide range of topics, and with some fields having controlled vocabularies, was deemed necessary.

Many heritage collections have somewhat skewed datasets with more items on some topics than on others.

Electronic thesis metadata, however, has a large amount of variety, as every record is, to some degree, created by a different individual and the records span

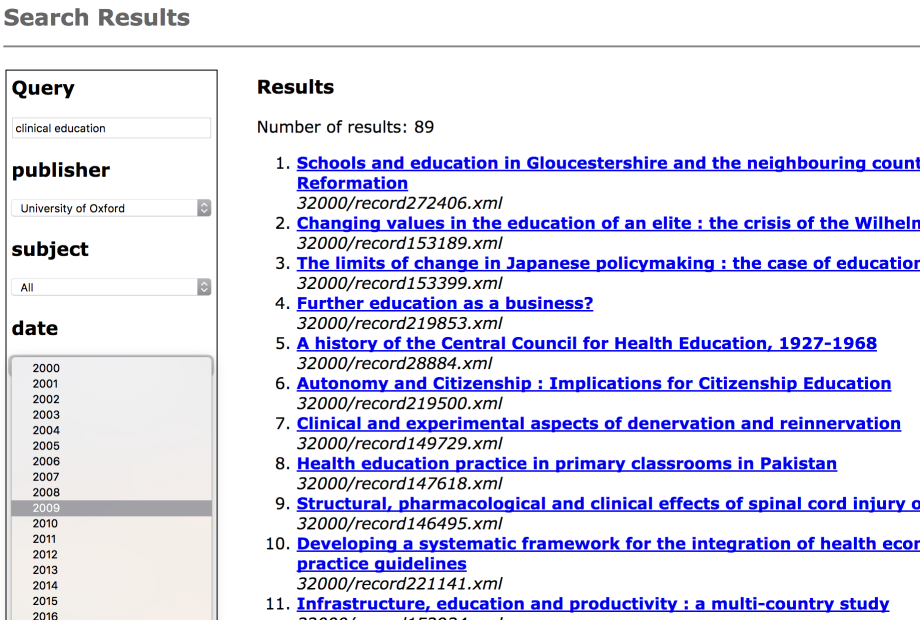

Fig. 1. Screen snapshot of search/browse interface.

many disciplines. ETD metadata was harvested using the Open Archive Initiative Protocol for Metadata Harvesting (OAI-PMH)[4] from the NDLTD Union Archive[1].

517854 records from UK universities were collected – the UK collection was used because it is a complete record due to a national mandate – in the Dublin Core format. Records were then randomly extracted to create experimental collection with sizes of 32000, 16000, 8000, 4000 and 2000. For comparability, each subset was a proper superset of the next smaller subset - so all records in the 2000 item set were contained in the 4000 item set.

Data was stored in individual XML files and each subset was separately indexed.

3.2 Experimental Design

Table 2 lists the queries used for the search and browse aspects of the experiments. These queries were manually selected such that, in each case, the first 2 returned were moderately popular (at about the 30th percentile), the second 2 were highly popular (at about the 80th percentile) and the final query was among the most popular terms within the index. These were all chosen by in-

[1] NDLTD Union Archive, http://union.ndltd.org/OAI-PMH/

specting the index for the 2000 item collection, thus ensuring that the results would also appear in every other collection.

Table 2. Queries used in experiments. S1-S5 are search terms used. B1-B5 are browse fields used.

Search/Browse	**Query terms**
Search (single term)	S1: comparative S2: simple S3: study S4: london S5: university
Search (multiple term)	S1: comparative study S2: simple relationship S3: clinical education S4: disease multiple S5: london university
Browse (single field)	B1: date=1954 B2: date=1959 B3: date=1977 B4: date=1986 B5: date=2011
Browse (multiple field)	B1: date=1954 and univ=University of Wolverhampton B2: date=1959 and univ=University of the West of Scotland B3: date=1977 and univ=University of Southampton B4: date=1986 and univ=University College London B5: date=2011 and univ=University of Oxford

For each collection size, 8 experiments were conducted. For each of the experiments, 5 queries appropriate to the experiment were used. Every query was submitted 5 times, and an average measurement was recorded.

The 8 experiments resulted from varying the complexity of search – either no term, one term or two terms – and the complexity of the browse – either no fields constrained, one field constrained or 2 fields constrained. As no experiment was conducted with neither search nor browse terms, this yielded 8 combinations.

The search system code was instrumented to execute experiments without user involvement but with a simulation of user selection (by manipulating the HTML DOM to set form values) and unchanged result display within the browser. Measurement error was minimised by recording times automatically (using the Javascript Date object) and determining the complete time for all repititions, followed by finding the average. Results were reported in a browser window at the end of the process and recorded.

3.3 Results and Discussion

All experiments were run on a Macbook with a 1,3 GHz Intel Core i5 processor, 8 GB of 1867 MHz LPDDR3 RAM, and a solid-state hard drive. Waterfox v56.2.10 was used as a browser - this is a derivative of Mozilla's Firefox, which allows for greater developer control. All background applications were kept constant during the experiments.

After a brief indexing discussion, the detailed results for one collection are presented and discussed, followed by a comparison of performance across collections.

Indexing Table 3 shows the time taken to index the different collections. The indexing time is roughly proportional to the collection size. This is not a major issue as, in practice, a collection is only indexed after changes are made and many collections, such as historical collections, are not changed often.

Table 3. Index creation time

Collection Size	time (in seconds)
2000	23.57
4000	55.82
8000	68.51
16000	134.99
32000	254.13

Collection Size 16000 Figure 2 illustrates the performance results for the single/multiple term search and single/multiple field browse for the 16000 item collection.

For the search queries, the highly popular term "university" appears in almost every result, hence takes a long time to process, given the complexity of the search algorithm. The moderately-popular search terms return results much faster. Browse operations all take a similar amount of time because the browsing operation filters the complete list so the work performed is similar in every case. All operations, including those that process most/all the items in the collection, return results in less than 500ms.

Table 4 displays the full set of results with all measurements corresponding to the 16000 item collection. The first 2 columns indicate the complexity of the search and browse respectively, and the individual entries are for specific query combinations.

The final column in Table 4 indicates average times for operations with different complexities. These range from 114-213 milliseconds, which is typically

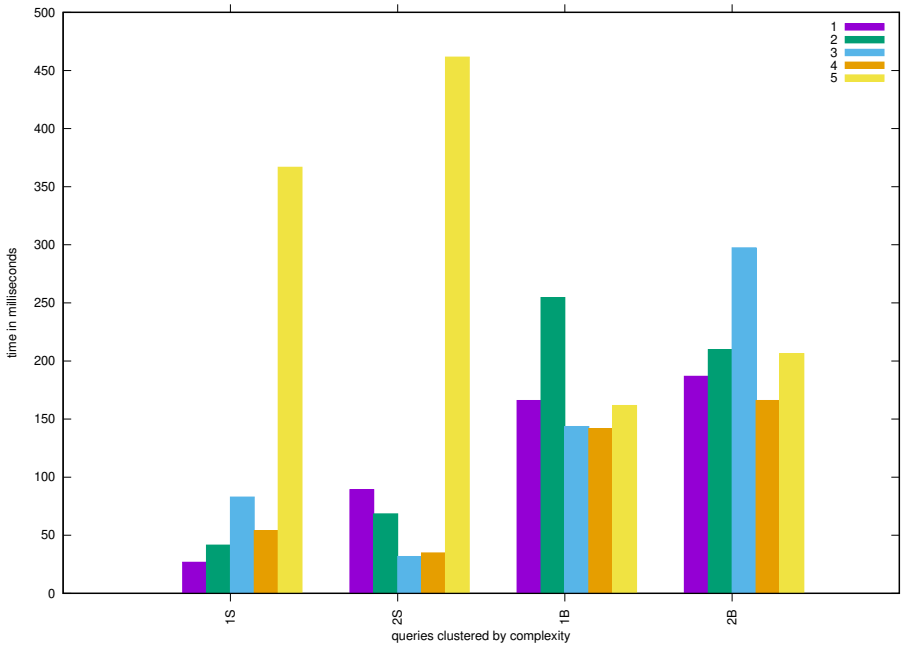

Fig. 2. Average times for queries of difference complexities. 1S/2S/1B/2B are the query complexities while the 5 data points within each cluster are the different queries tested.

Table 4. Detailed performance measurements (in milliseconds) for all query combinations for 16000 record test size. S=Number of search query terms. B=Number of browse fields. S1-S5 are search terms used. B1-B5 are browse fields used. Avg=Average time across all terms/fields for a given complexity.

S	B	S1					S2					S3					S4					S5					Avg
		B1	B2	B3	B4	B5	B1	B2	B3	B4	B5	B1	B2	B3	B4	B5	B1	B2	B3	B4	B5	B1	B2	B3	B4	B5	
0	1	142	137	133	139	142	155	190	148	143	218	241	285	172	142	160	138	489	137	155	135	154	172	129	130	154	174
0	2	171	343	545	118	158	155	193	306	175	190	259	169	157	139	314	184	156	179	264	193	166	187	300	133	177	213
1	0	24	20	25	44	21	29	38	35	30	76	134	67	63	68	83	58	51	52	58	51	442	366	344	332	349	114
1	1	26	20	20	36	26	21	31	38	24	40	88	81	202	146	223	162	72	56	90	83	410	372	398	394	361	137
1	2	16	13	18	19	25	14	15	19	20	27	101	58	54	57	68	60	67	87	77	70	772	357	462	403	503	135
2	0	119	80	89	80	78	67	56	59	124	36	32	31	30	32	34	34	30	45	33	32	392	480	470	513	454	137
2	1	154	119	85	99	127	53	56	77	63	41	28	22	29	31	40	29	44	34	39	49	480	431	459	507	390	139
2	2	68	76	72	117	94	95	39	85	150	182	25	23	27	29	36	29	28	32	47	50	498	422	527	410	553	149

adequate as a response time for users. The highest values in the table are related to the S5 ("university") columns , and the first 2 browse operation rows with no preceding search filter; these all process the largest number of items before returning results to the user so the performance is understandable.

All Collections Figure 3 illustrates the performance results for the single/multiple term search for all collections, for the specific terms "study" and "clinical education". The "study" broader term appears in most documents in larger collections and therefore there is a clear increase in processing time. The more specific query appears in fewer documents and therefore has a less predictable outcome. Overall, all results are processed within 150ms.

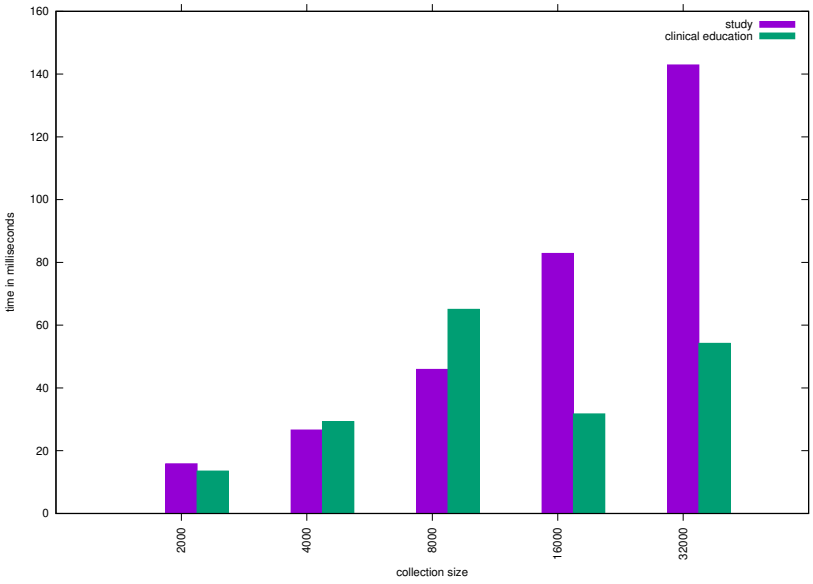

Fig. 3. Average times for search queries of difference complexities across all collection sizes.

Figure 4 illustrates the performance results for the single/multiple field browse for all collections, for the queries "year=1977" and "year=1977 and univ=University of Southampton". There is a general trend towards increasing times for larger collections, with some specific variation because of the random sample. Overall, all responses are processed within 350ms. This longer time is due to the entire list of results being filtered, with no pre-filtering from a search query.

Figure 5 illustrates the performance results for combinations of single/multiple term search and single/multiple field browse for all collections, for the same queries as in the prior figures. The single generic search term "study", when

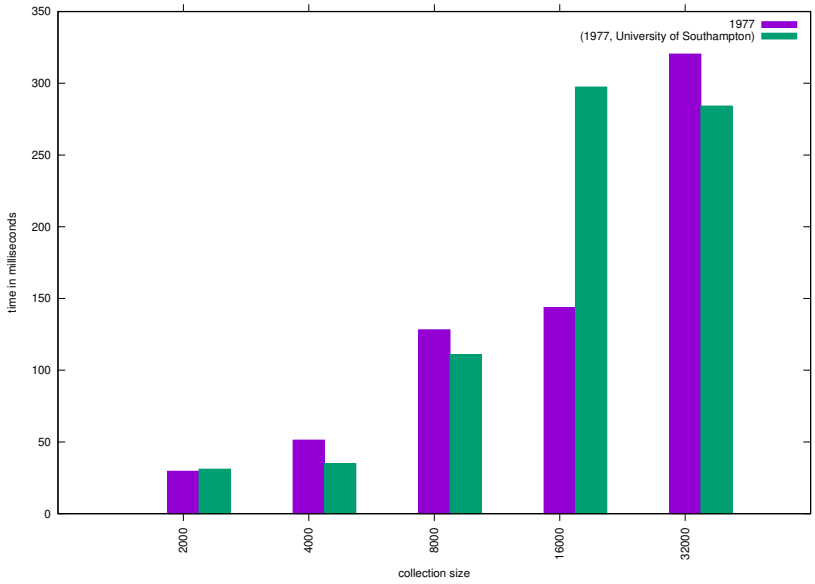

Fig. 4. Average times for browse queries of difference complexities across all collection sizes.

combined with one or more browse fields, yields longer processing times in many cases. The specific search query "clinical education" is more constrained, with far lower processing times, irrespective of the browse fields specified. All operations are, however, completed within 300ms.

On average, even for collections with 32000 items, complex search and browse combination queries were successfully processed in less than half a second. Clearly, the query outliers will result in longer times, but this performance is likely to be adequate for most users interacting with a faceted search system.

4 Related Work

One of the earliest works that demonstrates the notion of simple and static repositories is Project Gutenberg [2], which is an online repository to distribute out-of-print and free ebooks. The core system that manages the data is simple and does not rely on typical archiving tools. All data is required to be in simple formats, even though more advanced formats may be included as well.

The Greenstone project [13] pioneered the idea of a digital library system that could be used offline, where collections were indexed and distributed on CDROM. The technology used to create this experience required the installation of a specific operating system's Web server on the user's computer. While the Greenstone approach was innovative and addressed the needs of low-resource

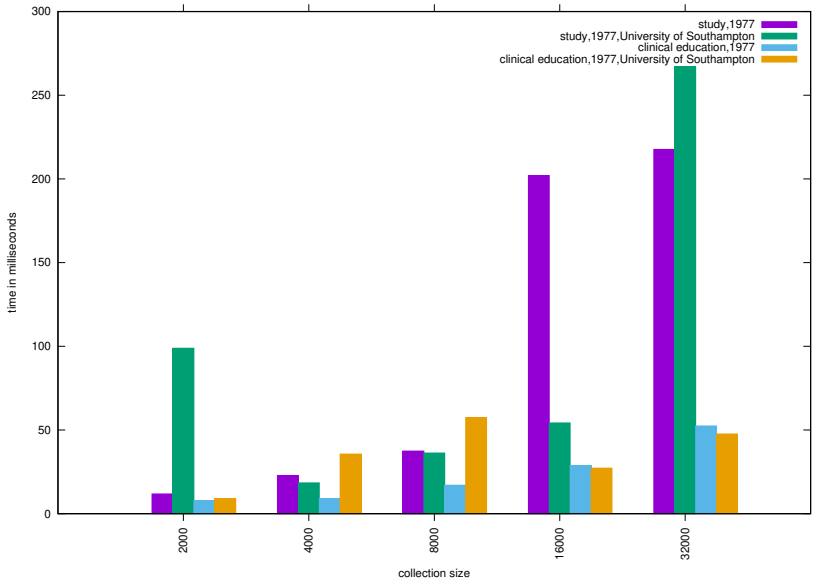

Fig. 5. Average times for faceted search/browse queries of difference complexities across all collection sizes.

environments, it predated the advanced browser technology currently available and exploited in this project.

The SimplyCT project [5][10] tried to define an alternative design approach for digital repositories in low resource environments, such as most African countries. This was exemplified in the Bleek and Lloyd archive, which was completely static and contained a simple HTML-based search service that worked within the browser [9]. Additional experiments with different repositories showed that static file stores could work for a wide range of repository requirements [11][6].

The Open Archives Initiative encouraged the use of simple solutions. While its flagship OAI-PMH standard is considered to embody many of the simplicity principles, the OAI also showed that this could be taken a step further to create static repository snapshots, enabling interoperability among even participants with little infrastructure [3].

5 Conclusions and Future Work

This experiment sought to investigate the performance of a search and browse system based completely within a browser, with no network connection and no software installation necessary. Using Javascript, such a system is clearly possible with data that is pre-indexed and stored in XML files. The results from extensive experimentation show that such a system has reasonable performance for typical operations on a small- to medium-sized collection.

Most operations completed within half a second, even on a 32000 item collection. This suggests that this offline approach may even work on collections of up to 100000 items in an acceptable amount of time. Given its advantages of portability, simplicity and network-independence, this can provide a useful alternative to indexing systems like Apache SOLR in specific situations.

On reflection, this experiment exploits advances in hardware and browser software in order to overcome other environmental limitations, like expertise, funding and Internet connectivity. It is an attempt to use the latest technological advances to address the limitations of low-resource environments.

Future work may include the use of JSON instead of XML for data storage. This may result in a constant factor improvement, due to small data sizes and faster parsing. Also, partitioning of the index files is being considered to support faster loading of initial results for simpler queries; this may help to address some of the slow browse operations that filter large result sets.

Acknowledgements

This research was partially funded by the National Research Foundation of South Africa (Grant numbers: 85470 and 105862) and University of Cape Town. The authors acknowledge that opinions, findings and conclusions or recommendations expressed in this publication are that of the authors, and that the NRF accepts no liability whatsoever in this regard.

References

1. Culture, A..P.: The five hundred year archive, www.apc.uct.ac.za/apc/research/projects/five-hundred-year-archive
2. Hart, M.: The history and philosophy of project gutenberg. Project Gutenberg **3**, 1–11 (1992)
3. Hochstenbach, P., Jerez, H., Van de Sompel, H.: The oai-pmh static repository and static repository gateway. In: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries. pp. 210–217. IEEE Computer Society (2003)
4. Lagoze, C., Van de Sompel, H.: The open archives initiative: Building a low-barrier interoperability framework. In: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries. pp. 54–62. ACM (2001)
5. Phiri, L.: Simple digital libraries. Ph.D. thesis, University of Cape Town (2013)
6. Phiri, L., Williams, K., Robinson, M., Hammar, S., Suleman, H.: Bonolo: A general digital library system for file-based collections. In: International Conference on Asian Digital Libraries. pp. 49–58. Springer (2012)
7. Pyrounakis, G., Nikolaidou, M., Hatzopoulos, M.: Building digital collections using open source digital repository software: A comparative study. International Journal of Digital Library Systems (IJDLS) **4**(1), 10–24 (2014)
8. Smith, M., Barton, M., Bass, M., Branschofsky, M., McClellan, G., Stuve, D., Tansley, R., Walker, J.H.: Dspace: An open source dynamic digital repository (2003)
9. Suleman, H.: Digital libraries without databases: The bleek and lloyd collection. In: International Conference on Theory and Practice of Digital Libraries. pp. 392–403. Springer (2007)

10. Suleman, H.: An african perspective on digital preservation. In: Multimedia Information Extraction And Digital Heritage Preservation, pp. 295–306. World Scientific (2011)
11. Suleman, H., Bowes, M., Hirst, M., Subrun, S.: Hybrid online-offline digital collections. In: Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists. pp. 421–425. ACM (2010)
12. Van Garderen, P.: The ica-atom project and technology. In: Third Meeting on Archival Information Databases.(16-17 julio 2009: Rio de Janeiro). Trabajos presentados. Rio de Janeiro: Association of Brazilian Archivists (2009)
13. Witten, I.H., McNab, R.J., Boddie, S.J., Bainbridge, D.: Greenstone: a comprehensive open-source digital library software system (2000)
14. Witten, I.H., Witten, I.H., Moffat, A., Bell, T.C., Bell, T.C., Bell, T.C.: Managing gigabytes: compressing and indexing documents and images. Morgan Kaufmann (1999)