SimplyCT Online Search

Azhar Desai
University of Cape Town
Rondebosch
Cape Town, South Africa
desai.azhar@gmail.com

ABSTRACT

Based on the SimplyCT framework, an online search is developed to test whether being online improves the effectiveness of the search. The search is built using the Xapian toolkit. It could not be shown that the effectiveness improves, however the users tested found the search satisfactory.

Categories and Subject Descriptors

H.3.7 [**Information Storage and Retrieval**]: Digital Libraries—*Collection*

General Terms

Experimentation

Keywords

SimplyCT, online, search, Xapian

1. INTRODUCTION

Large archives stored in the SimplyCT framework need a convenient way to be searched. With archives that rarely change we can pre-generate indices for fast searching. These indices would rarely need to be updated. This research project investigates the effectiveness of an online search based on the SimplyCT framework.

2.

2.1 The Research Method

A simple online search was developed based on the SimplyCT framework. The search uses Xapian to generate indices and perform queries. The hypothesis is to test whether an online search improves the effectiveness of the search. The effectiveness of the search was evaluated in the following three categories:

- Speed of search (as compared to offline)
- Ease of use (for the searcher)
- Advantage of feedback service

The feedback service was not implemented. However, the supporting code was left in place for any future implementation of it. A comparison of this online search with Marc Bowes's CALJAX offline search was planned. There was not enough time for this comparison.

2.2 A Description of the System

2.2.1 *SimplyCT framework*

The SimplyCT framework consists of the archive data, services and indices for it grouped together in a hierarchical directory. The archive files are described in accompanying XML metadata files. The services follow a server-instance model, with the shared code in the `archive-root/lib` directory. The following is the directory tree for the test search archive:

```
ArchiveRoot
|
|-- archive
|   `-- test
|       `-- archive folders not displayed
|-- indices
|   `-- test
|       `-- search
|           |-- all
|           |-- description
|           |-- word1
|           `-- word2
|-- lib
|   |-- preprocessor
|   |-- search
|   |-- shared
|   `-- www
|       |-- js
|       `-- stylesheets
`-- services
    |-- search
    |   `-- test
    `-- www
        `-- test
            |-- js
            `-- stylesheets
```

The folders in the indices sections of the test search correspond to the tags in the metadata file.

2.2.2 How the Search Works

The user accesses the Web page from services `www/test/index.html`. The JavaScript, CSS and XSL stylesheets are in the corresponding folders. When the user searches the archive, a XMLHTTPRequest submits a GET request from the Python CGI script located at `services/search/test/search.py`.

The CGI scripts in the subdirectories of `services/search` are instances of the search service applied to a particular archive and in this case, the test archive. These scripts create unique external references for the searches on a particular archive. It uses the shared code in the `lib/search` directory to perform the search query.

The code in `lib/search` makes use of shared code in `lib/shared`. The shared directory includes code to work out and create the various paths of the archive and to interface with Xapian. Xapian handles the creation and searching of the indices. The JavaScript on the client side receives the XML formatted results, transforms it into XHTML and then inserts it into the page.

2.2.3 Description of software used

The website is hosted on a computer using:

- Ubuntu 9.10 using the 2.6.31-17-generic Linux kernel
- Apache 2.2.12
- Python 2.6.4
- Xapian in the Ubuntu package: libxapian15 1.0.15-2ubuntu2
- Xapian python bindings in the Ubuntu package: python-xapian 1.0.14-1build1

2.2.4 Limitations

The code presently has a few limitations.

Unicode Handling The Python code was intended to be very Unicode aware since collections such as the Bleek and Lloyd collection includes a wide variety of characters.

Unicode characters inside the metadata files are handled well. The preprocessor script expects Unicode characters and normalises them to be handled by the Xapian database.

Unicode handling does not work with non-ASCII characters as filenames and directory names.

Relative Imports The preprocessor currently needs to be run while the current directory is set to the directory containing it.

2.3 Testing Procedures

An archive was generated with random data to test the search. The test archive used consisted of 720 files and a further 720 metadata files in as many directories.

The speed of the search was tested in three ways: the first is by measuring the time it takes for the system to perform a search independent of displaying the information and network delays. This was done by accessing the cgi script directly and timing the response. On the server the following bash command was executed:

```
 $ time GET -d http://localhost/services/search
/test/search.py?searchtext=searchstuff
```

The second way is to time the search from another computer on the local area network. The same command as above was used, replacing localhost with the domain of name of the server.

The third test is to see if users are satisfied with the response speed. This will be measured by their approval indicated on the user evaluation forms. The search queries used were:

1. (a blank query)
2. hello
3. one mango will make no difference
4. get lots of results master lope the potholes surround your neighbourhood and kill daisies for fun

The ease of use for the user will measured with the evaluation forms. This asks them how easy it is to locate a specific item in the search, how they found responses on the site and about their general impressions.

2.4 Findings and Discussion

2.4.1 Speed of Search

Table 1 suggests a slight increase in search time for longer queries. However, there was a large variation in timings for each measurement so that trend may not continue. As indicated earlier, these results cannot be compared to the CALJAX offline search, so it is not known if this is faster.

2.4.2 User Responses

Generally the users tested found the search to be fast and easy to use. The average rating of the speed of the search was 8.4 and for ease of use 9.5 out of 10.

They made several suggestions including to:

- show less results for long searches
- make layout more customisable
- show the search terms when displaying no results found
- fix the display of result sets, when the number of results are greater than 25

Table 1: Average real times of the performing queries from different computers (in seconds)

	Search Query 1	Search Query 2	Search Query 3	Search Query 4
localhost	0.191	0.202	0.202	0.210
Local Area Network	0.248	0.249	0.252	0.265
Number of Results	0	2	31	49

- break up the display of large result sets into several smaller ones.

Apart from the limitations, it seems the online search works reasonably well. The users find it fast enough. Though not explicitly shown it seems reasonable to suggest the speed varies less across different browsers and operating systems as seen in the offline search of Marc Bowes's (CALJAX 2009). The users find the layout of the searching interface usable.

3. CONCLUSIONS

The online search was found to be effective by users. However it could not be shown whether there was an improvement of effectiveness. It is speculated that some consistency to the speed across the different operating systems and browsers is attained, though this was not tested.