NUROP CONGRESS PAPER
Music Lyrics Spider

Wang Litan[1] and Kan Min Yen[2]
School of Computing, National University of Singapore
3 Science Drive 2, Singapore 117543

ABSTRACT

The World Wide Web's dynamic and unstructured nature and exponential growth rate causes difficulty in locating relevant information. Constrained searching within a particular website is also difficult. Focused web crawlers are widely used in collecting domain-specific documents from the web to build locally-stored repositories, in order to tackle these problems. Many domain-specific websites make uses of listings of hyperlinks to guide the users in locating desired information. In this project, both web searching algorithms and webpage analysis algorithms are examined for their possible uses in this focused web crawling. This paper presents a way to formalize the use of the link listings properties and proposes a context-based model in improving focused web crawling, both in precision and recall rates. Using lyrics websites as the example, a prototype system has been developed to crawl and index lyrics pages from these websites. Testing done on six lyrics websites listed in Open Directory Project has shown promising results.

1 INTRODUCTION

The Internet has become the largest information database with billions of web pages. However its dynamic and unstructured nature and exponential growth rate causes difficulty in locating relevant information. Constrained searching within a particular website is also difficult. These problems hinder the progress of scientific research relying on web resources. One of the most effective solutions has been to collect domain-specific documents from the web to build local databases using *focused web crawler*.

Lyrics' textual nature enables them to be easily distributed and stored using relatively fewer resources on the Internet. Lyrics websites share some crucial commonalities: they undertake similar page content organization and use a similar linking system (such as an alphabetical or listing of hyperlinks on each index page). These characteristics define lyric websites and other e-commerce sites with database backends which require a specialization of the focused crawling scheme.

2 DESIGN AND IMPLEMENTATION

[1] Student

[2] Supervisor

2.1 Algorithm Design

Different algorithms are used in focused crawling to determine the order in which pages are visited. Breadth-first search algorithm can be used to collect topic-specific pages when the target website has limited and known number of levels to be crawled. However, even when one can provide a base URL of high relevance and hyperlink concentration, after several levels of crawling, it starts to lose focus and may retrieve irrelevant and repeated pages, resulting in a noisy final collection. Best-first search algorithm makes use of heuristics to rank the URLs maintained in the frontier. Promising ones are reordered to be visited sooner. Less promising ones are shifted to the end of the queue. However pages corresponding to less promising URLs determined by rank functions but yet still yielding ones will be missed by the crawler when the harvesting number of pages from the given website has a ceiling value. Lyrics website crawling requires a simple strategy, similar to breadth-first search, yet needs to be effective. As the final goal is exhaustive local collection of lyrics from a lyric website, breadth-first search with suggesting rank functions based on specific and highly efficient webpage analysis algorithms dealing with lyrics website pages is used.

Webpage analysis algorithms can be roughly divided into two approaches: content-based and link-based. Content-based approaches examine the page content to determine relevance. In the scenario of focused crawling, relevance can be equated with whether the page is in the target domain: in my case, whether or not a page contains song lyrics. On the other hand, link-based approaches, exploit the human effort made in constructing hyperlinks to and from a target page. Google's PageRank algorithm is the most-widely known instance of this approach.

In the present work, I have applied both. One method to model the content of target pages is to build a language model to represent the typical words found on those pages. Supervised machine learning can used to create such a model. In my work, I use BoosTexter, a machine learner that uses boosting to create a decision list for classification (Schapire 2000).

During the research we observed that several lyrics websites indexed by Google had different PageRank values for different web pages levels. PageRank is the aggregate figure that takes into account global structure of the web graph. However, in many sites that only have incoming links at the root page or have these incoming links uniformly distributed in all parts of the website, relative PageRank between pages within a site can indicate the level of pages. This observation can be leveraged used in determining the categories of the web pages.

2.2 State-Transition Model

We develop a general model that includes all the possible states and transitions particular to our chosen application of spidering lyrics. We utilize this model for focused crawling at run-time. The following states between transitions are identified after a thorough survey of many lyrics websites with different site sizes and structures:

Root: the root page of lyrics website, given by the *Root URL*. It usually provides alphabetical index or categorized hyperlinks to artist list pages (*Artist*).

Artist: the list page of artists in the lyrics website, given by hyperlink from the root page (*Root*) or the base URL. The *Artist URL* usually indicates a particular indexing letter and/or a particular

range of number of artists. *Artist* mainly provides hyperlinks to song list pages (*Song*) of the indexed artists.

Song: the list page of song titles by a particular artist or several closely named artists in the lyrics website, given by hyperlink from the artist list page (*Artist*) or the base URL. *Song* mainly provides hyperlinks to lyric pages (*Lyric*) of the songs listed.

Lyric: the lyric page of a given song title and artist in the lyrics website. The *Lyric URL* usually indicates the artist name and the song title.

In the modified strict model, we allow only transitions between two consecutive levels. As such, we can simplify the multi-class classification task into a binary one. This binary classification only seeks to distinguish whether the page's state is the same as its parent (a self-loop in the state diagram) or a state deeper (a transition to a more detailed state, e.g., *Song* $\Rightarrow$ *Lyric*). The key observation is that this differentiation is sufficient. We can thus limit our search for useful features to features that help distinguish between adjacent states in the state transition graph.

Root versus Artist. Assuming the root URL provided by the user is the root URL of the target lyrics website, a strong indicator to identify a *Root* page is an alphabetical listing. Alphabetical listing refers to the set of hyperlinks with "A, B, C…Z" or similar features as their anchor texts. A pilot evaluation shows that this feature can produce 100% accurate differentiation of *Root* pages from *Artist*.

Artist versus Song. In order to differentiate *Artist* from *Song*, we use a similar feature for list differentiation based on the distribution of part-of-speech (POS) tags in the anchor text. We note that *Artist* pages contain more proper nouns than *Song* pages since *Artist* pages mention specific people's names. To use this feature, we have to first parse the page to extract and tokenize all of the anchor text. Next, we tag these tokens with POS tags conforming to the widely-accepted Penn Treebank standard. We observe an optimal ratio of 0.275 generates the highest accuracy, and thus set the threshold accordingly.

Song versus Lyric. Here, number of links to document size ratio is used. We observed that *Song* pages tend to have more hyperlinks embedded for the same document size than *Lyric* pages due to their hyperlink listing property. Moreover, *Lyric* pages are typical of leaf (content) pages in a site. Such pages do not list many hyperlinks, due to their low link centrality. In our development set, an optimal ratio of 1.33 generates the highest decision accuracy.

2.3 Implementation and Refinements

In order to fetch a page, we build an HTTP client which sends HTTP request for a web page and reads the response. In the actual implementation of the spider, it is simulated to be a web UserAgent. Once the page is fetched, we parse its content to extract information that guides the future path of the crawler. Most importantly, the web page needs to be parsed to extract hyperlinks to build the URL frontier. URLs in relative form (e.g., /~litan/) are converted to absolute form (e.g., http://www.comp.nus.edu.sg/~litan/) by using Perl's URI module.

URL filtering is essential to this research as it restricts the crawler's direction so that it focuses on spidering the desired path and retrieves the target lyrics pages. The transitions reduced from the original state-transition model to the modified version are basically eliminated through URL

filtering. The method proposed, defined as horizontal elimination, compares URLs from two pages known to be at the same level (e.g., two *Artist* pages). Any URLs that are in common between the pages are removed from the URL frontier (**even if they were formerly queued**), as these are likely to be navigation links for the whole website rather than the content hyperlinks that mark a transition to a state closer to the leaf. This comparison and elimination is made between pages of the same state, i.e. the same horizontal level.

Many large lyrics websites maintain large number of leaf pages and intermediate nodes in their collections. It is often impractical to list all these pages (e.g., artists of names beginning with the same alphabet letter) in a single page. A common approach in websites is to chunk the list into several pages, connected by "Next Page" and "Previous Page" hyperlinks. These pages which survive the horizontal and vertical URL eliminations should be classified all as the *Artist* state rather than the lower ordered states. However, normal crawlers, including focused crawlers, will not be able to detect this problem. It accounts for the low recall rate of these crawlers, as they miss a significant part of the entire website due to this single problem. Explicit recursion methods are implemented in my system to resolve this problem. Pages that are likely self-loops (*Artist* $\Rightarrow$ *Artist*) identified as new *Artist* and reprocessed as *Artist* rather than the next state *Song*. A similar procedure is used in the case of *Song* versus *Lyric*.

3 EVALUATION AND ANALYSIS

To assess the performance of music lyrics spider, six lyric websites were randomly selected from ODP. The six selected websites are allocated to three categories based on the state-transition model. Sites 1, 2 and 3 are identified as examples of category R, which has *Root* as the starting page. Sites 4 and 5 are classified to category A with *Artist* as the base page. Site 6 belongs to the last category S, since their base pages were identified to be of *Song* state. We tested four different music lyric crawler configurations. Common to all configurations, different rounds of crawling were carried out on these websites based on their respective site levels. The level of crawling is manually set for different categories. Category R, A, and S will be crawled to the 4^{th}, 3^{rd} and 2^{nd} level correspondingly with the starting page as the 1^{st} level.

1. **Breadth-First search**. The crawler fetches the website on a level by level basis. Domain restriction is applied, i.e. only pages with URLs within the target website are crawled.
2. **Breadth-First search with crawling of alphabetical listings enabled.** Instead of crawling every page linked by the *Root* page, only pages linked through alphabetical hyperlinks (links with anchor text of "A", "B", "#" and so on) will be crawled as the 2^{nd} level.
3. **BFS with crawling of alphabetical listing and page analysis**. In addition to configuration #2, pages crawled will analyzed by the ratio threshold. This determines their level and distance to the final lyrics pages, and overrides the crawling levels, as the crawling is automatically carried according to the level decided by the classifier.
4. **Full configuration**. In addition to the configuration #3, horizontal and vertical URL eliminations will be used instead of general URL filtering.

The experiments were carried out on two servers. The number of pages crawled for each level was recorded for assessment. There are many indicators of the performance of a focused crawler.

Relevance (precision), coverage (recall) and time consumption are usually included. *Precision* is calculated as the ratio of number of correct *Lyric* pages identified and fetched by the crawler to the number of total identified and fetched pages. *Recall* is calculated as the ratio of number of correct *Lyric* pages identified and fetched to the number of *Lyric* pages maintained in the crawled lyrics website. Fig 1 shows the results obtained for the six selected websites.

	Site	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6
Round	Level	4	4	4	3	3	2
1st	Precision	0.760	0.920	0.863	0.887	0.944	0.827
	Recall	0.253	1.00	0.742	1.00	1.00	1.00
	Duration(h:m)	5:12	20:13	9:55	NA	NA	NA
	URLs fetched	26756	128508	47291	211	295	29
2nd	Precision	0.772	0.952	0.874	0.887	0.944	0.827
	Recall	0.255	1.00	0.757	1.00	1.00	1.00
	Duration(h:m)	4:57	20:10	9:14	NA	NA	NA
	URLs fetched	25347	128009	44588	211	295	29
3rd	Precision	0.733	0.696	0.808	0.793	0.985	0.968
	Recall	0.634	0.709	0.726	1.00	1.00	1.00
	Duration(h:m)	16:33	22:49	10:53	NA	NA	NA
	URLs fetched	70238	106692	41127	230	299	29
4th	Precision	0.849	0.840	0.830	0.858	0.985	0.968
	Recall	0.761	0.835	0.774	1.00	1.00	1.00
	Duration(h:m)	15:10	19:57	9:29	NA	NA	NA
	URLs fetched	91382	113036	42019	208	295	29

Fig 1: Statistics results of websites crawled

From the results, we see that performance gains are most notable when crawling is done on larger websites. However, there are limited difference in performance between the breadth-first search and the fully modified configuration in crawling smaller websites. Accurate observation of the depth needed in a site is sufficient for most webpage retrieval.

As lyrics websites normally preserve the high concentration of lyrics pages within the domain, it is common to obtain high relevance ratio from crawling. With the slight modifications to the crawler which add alphabetical listing crawling, an average improvement of 1.8% is observed. With effective elimination of noisy links on the intermediate level pages, i.e. *Artist* and *Song*, the precision ratio rose back by an average of 9.4% for the final round of full configuration.

Breadth-first search algorithm failed to perform for Site 1 in recall, which has many separate pages of listings of artist names and song titles under the same initial alphabetical letter. The fixed 4-level crawling lead to a high miss rate as the crawling stops at many 2nd or 3rd level pages. This accounts for the considerably low recall rate of 25.3% only. Page classification attempts have managed to resolve this problem as we can see the recall rate were pulled up to 63.4% and 76.1% respectively in the rounds of BFS with crawling of alphabetical listing and page analysis and full configuration. Moreover, we can see that there is general reduction in crawling time consumed between the rounds of BFS with crawling of alphabetical listing and page analysis and full configuration, indicating that horizontal and vertical eliminations of URLs save time comparing with general URL filtering, especially for site crawling of over 100,000 pages. However, considerable amount of time has to be spent on page classification process, which cancels part of time improvement made in the last round of full configuration of the spider.

4 CONCLUSION

Although issues associated with the crawling process are handled in this project, there are still several areas to be tackled in the future. First, many websites maintain similar collections of the same information; work is needed to develop a distinct database of the specific domain. Moreover, the current page classification features can be improved. Last but not least, if the language model developed using BoosTexter could be trained with larger sample of pages to attain higher classification precision within a comparably shorter period of time; it could be added onto the crawler for higher automation work.

5 REFERENCE

Chau M., Zeng D. and Chen H. (2001). Personalized Spiders for Web Search and Analysis. In *Proceedings of the 1st ACM/IEEE-CS joint Conference on Digital Libraries*, Roanoke, Virginia, United States, 2001, pp79-87.

Cho J. (2001). Crawling the Web: Discovery and Maintenance of Large-Scale Web Data. Ph.D. Thesis, Stanford University, November 2001.

Qin J., Zhou Y. and Chau M. (2004). Building Domain-Specific Web Collections for Scientific Digital Libraries: A Meta-Search Enhanced Focused Crawling Method. *In Proceedings of the 2004 joint ACM/IEEE Conference on Digital Libraries*, Tuscon, AZ, USA, 2004.

Schapire Robert E. and Singer Yoram (2000). BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, 39(2/3), 2000, pp135-168.

Seymore K., McCallum A. and Rosenfeld R. (1999). Learning Hidden Markov Model Structure for Information Extraction. *AAAI 99 Workshop on Machine Learning for Information Extraction*, 1999.