A Simple Focused Crawler

Ah Chung Tsoi
Office of Pro Vice-Chancellor (IT)
University of Wollongong
Wollongong, NSW 2522,
Australia

Daniele Forsali
Dipartimento di Ingegneria
dell'Informazione
Universita' degli studi di Siena
Siena, Italy

Marco Gori
Dipartimento di Ingegneria
dell'Informazione
Universita' degli studi di Siena
Siena, Italy

Markus Hagenbuchner
Office of Pro Vice-Chancellor (IT)
University of Wollongong
Wollongong, NSW 2522,
Australia

Franco Scarselli
Dipartimento di Ingegneria
dell'Informazione
Universita' degli studi di Siena
Siena, Italy

ABSTRACT

A focused crawler may be described as a crawler which returns relevant web pages on a given topic in traversing the web. There are a number of issues related to existing focused crawlers, in particular the ability to "tunnel" through lowly ranked pages in the search path to highly ranked pages related to a topic which might re-occur further down the search path. We will introduce a simple focused crawler, which is described by two parameters, viz., degree of relatedness, and depth. Both provide an opportunity for the crawler to "tunnel" through lowly ranked pages. Results from initial experiments are promising and motivate for further research.

Keywords

Focused web crawler, best first search, topic specific search

1. INTRODUCTION

There are two types of web crawling strategies deployed by web search engines, viz., breadth first search strategy and "best" first search strategy. Breadth first search strategy endeavors to build a general index of the web covering any conceivable topic by endeavoring to search a significant portion of the web. The "best" first search focuses on the retrieval of pages which are relevant to a particular given topic. A crawler using a "best" first search strategy is known as a "focused crawler".

A focused crawler has the following main components: (a) A way to determine if a particular web page is relevant to the given topic, and (b) a way to determine how to proceed from a known set of pages. An early search engine which deployed the focused crawling strategy was proposed in [1] based on the intuition that relevant pages often contain relevant links. It searches deeper when relevant pages are found, and stops searching at pages not as relevant to the topic. Unfortunately, the above crawlers show an important drawback when the pages about a topic are not directly connected in which case the crawling might stop pre-maturely.

WWW2003, May 20–24, 2003, Budapest, Hungary.
ACM xxx.

This problem is tackled in [3] where reinforcement learning permits credit assignment during the search process, and hence, allowing off-topic pages to be included in the search path. However, this approach requires a large number of training examples, and the method can only be trained offline. In [2], a set of classifiers are trained on examples to estimate the distance of the current page from the closest on–topic page. But the training procedure is quite complex.

Our focused crawler aims at providing a simpler alternative for overcoming the issue that immediate pages which are lowly ranked related to the topic at hand. The idea is to recursively execute an exhaustive search up to a given **depth** d, starting from the "relatives" of a highly ranked page. Hence, a **set of candidate pages** is obtained by retrieving pages reachable within a given perimeter from a **set of initial seeds**. From the set of candidate pages, we look for the page which has the best score with respect to the topic at hand. This page and its "relatives" are inserted into the set of pages from which to proceed the crawling process. Our assumption is that an "ancestor" with a good reference is likely to have other useful references in its descendants further down the lineage even if immediate scores of web pages closer to the ancestor are low. We define a **degree of relatedness** r with respect to the page with the best score. If r is large, we will include more distant "cousins" into the set of seeds which are further and further away from the highest scored page. This device overcomes the difficulties of using reinforcement learning in assigning credits, without the burden of dynamic programming. These ideas may be considered as an extension to [1, 2], as the use of a degree of relatedness extends the concept of child pages in [1] while avoiding the complex issue of inherence of scores, and the use of a perimeter is similar to the "layer" concept used in [2].

2. DEFINITIONS

The **web** is a directed graph $G = (P, L)$ where $p \in P$ is a node (i.e. a page) and $(p, q) \in L$ is a **directed** link from node p to node q. The **crawler**, given a seed node s, explores the web G and builds the **visit tree** $T = (V, L_c)$, where V collects the **visited pages**, and $L_c = \{(p, q) \in L |\ q$ is visited by the crawler from $p\}$. The **border** B are

the nodes in T that have outlinks to nodes outside T. The **depth**, d, is the maximum distance from the seed node, and the **degree of relatedness** r is defined as follows: an r ancestor of q is a node a for which there exists a path $a \rightarrow_r q$ in T with $r-1$ arcs to q such that $a \rightarrow_r q$. Then, if $a \rightarrow_r q$, the r degree relatives of p are the nodes q_i for which $a \rightarrow_r q_i$ holds. Finally, the **score** of a page $S(p)$ is the relative importance of page p about a particular topic. Important pages are scored high.

3. THE PROPOSED ALGORITHM

Given d and r, the algorithm is as follows:

Step 1 Initialize the algorithm with a seed page s and set $T = (\{s\}, \emptyset)$, $B = \{s\}$.

Step 2 Select from B the best scored page $p \in B$ s.t. $S(q) > S(p)\ \forall q \in B$.

Step 3 Find the set E of relatives of p of a given degree of relatedness r.

Step 4 Use every node of the set E to perform an exhaustive exploration of the web up to the depth d, and update T and B accordingly.

Steps 2 to 4 are repeated until $B = \emptyset$ or a maximum number of operations is reached. Note that the algorithm halts when all the pages reachable from s were visited. In order to guarantee that most of the web is reachable, the method can be straightforwardly extended using a set of seed pages.

4. AN EXPERIMENT

In order to validate our algorithm, we carried out some preliminary experiments. The purpose was to understand the behaviors of the algorithm and to clarify whether the method can be used to speedup a basic focused crawler (a crawler without sophisticated mechanisms for jumping over lowly ranked pages). The data set we used was WT10g, distributed by CSIRO in Australia, that contains a snap shot of a portion of the world wide web consisting of 1,692,096 documents from 11,680 servers. We used a page classifier based on naive Bayes method [4] to assign a score to each page with respect to some randomly chosen topic.

The measures used to characterize the behaviour of the crawler is the *precision* $\eta = \frac{n_q}{n_c}$, where n_q is the number of relevant pages that the crawler has visited, and n_c is the total number of pages visited. The experiment illustrated in Figure 1 obtains η as a function of d, and r respectively. Plotted is the number of web pages visited against the number of relevant pages retrieved. The topic used is "Linux" which has 2600 relevant pages in the dataset.

Note that for $d = 1, r = 1$ our crawler becomes a basic focused crawler. This experiment shows that, at the beginning of the crawling, the basic focused crawler is more efficient than our crawlers. But, as n_c increases the crawlers with $d > 1, r > 1$ give better results. More precisely, Figure 1 shows that choosing either $d = 2$ and/or $r = 2$ produce the best results in the later stage of the search process. This observation was confirmed by other experiments (not shown due to lack of space) on other general and special topics.

The results have an interesting interpretation. The seeds, which are on-topic pages, are probably directly connected to other on-topic pages. Those pages can be easily collected at the beginning by a basic focused crawler. Then, the crawling task becomes more difficult making crawlers with $d = 2$ and/or $r = 2$ more efficient. However, large values of d and r may force the retrieval of more and more pages which are not immediately relevant to a topic at hand.

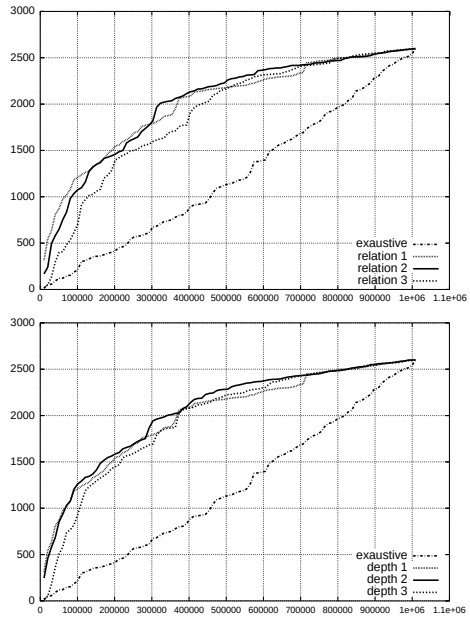

Figure 1: Comparison of the behaviour of crawlers for the topic "Linux". Shown respectively are η for $d = 1$, $r = 1, 2, 3$ (top), and $r = 1$, $d = 1, 2, 3$ (below).

5. CONCLUSIONS

The experimentation proves that the proposed algorithm is able to jump over unrelated pages in order to access relevant pages. Thus, it can be used to speedup a basic focused crawler. Of course, our method can be improved by the use of adaptive values for r and d, such that $d = r = 1$ holds at the beginning of the crawling process, and then to increase d, r as the crawling performance decreases.

Finally, note that even if the presented results are preliminary and experiments are needed to establish a direct comparison with other focused crawlers, the algorithm is considerably less computationally demanding than the ideas expressed in [1, 2]. In addition, we do not require a large training set, and offline training is not necessarily required.

6. REFERENCES

[1] De Bra, P., Houben, G., Kornatzky, Y., Post, R. "Information retrieval in distributed hypertexts". *Proc. 4th RIAO Conference*, 1994.

[2] Diligenti, M., Coetzee, F., Lawrence, S., Giles, L., Gori, M. "Focused crawling using context graphs". *Proc. VLDB 2000*, Cairo, Egypt, 2000.

[3] McCallum, A., Nigam, K., Rennie, J., Seymore, K. "Automating the Construction of Internet Portals with Machine Learning". *Information Retrieval*. Vol 3, 127-163, 2000.

[4] McCallum, A., Nigam, K. "Employing EM in pool-based active learning for text classification". *Proceedings of ICML-98*, San Francisco, US, pp.350–358, 1998.