A Framework for adaptive focused web crawling and information retrieval using genetic algorithms

Kevin Sebastian
Dept of Computer Science, BITS Pilani
kevseb1993@gmail.com

1 Abstract

The web is undeniably the largest repository of information today. Containing uncountable web pages it is a herculean task to traverse the entire web and doing so requires enormous amount of resources. This paper focuses on a particular aspect of web crawling namely focused web crawling when crawler(s) are only focused on extracting information pertaining to certain subject(s).

While this is not practical in the case of commercial web search engines where the number of topics or subjects are incredibly vast, this helps when only information pertaining to a certain topic is required by an individual or an organization. Since this crawling strategy is adaptive, it can learn to pick out the relevant data from relevant web pages and thus save a lot of computing power and memory by not processing pages or information that is of little relevance to the desired topics.

2 Introduction

The web contains billions of web pages and is growing rapidly. A conventional search engine allows users to enter a query and almost immediately fetches a list of pages containing content relevant to the terms of the query. These pages are fetched using crawlers which utilizes the hyperlink structure between pages to traverse the web and extracts information from each page.

Focused crawlers are able to seek out and collect subsets of Web pages that satisfy some specific requirements. It performs much better than traditional web crawlers when a search is needed to be done at query time and is also effective in crawling the deep web or dynamic web pages not accessible to traditional search engines.

An adaptive focused crawler incorporates a feedback system where the crawling agents learn the relevance or the need to traverse any page.
To ensure that an adaptive focused crawler is efficient and doesn't stray outside its domain we need to achieve two objectives:

1. Find pages which have a high degree of relevancy to the search topic(s).
2. Obtain these pages using minimal resources, i.e., the number of redundant hops should be minimized.

To achieve these objectives we use a combination of genetic algorithm and reinforcement learning-based approaches. We achieve this by using an adaptive crawler which learns the relevancy of a link to the topic at hand. While this work doesn't go too deep on the heuristics or parameters to judge the relevancy of a hyperlink, it presents a framework which demonstrates how we can model an adaptive crawler using genetic algorithms.

### 3 Related work

There has been quite a bit of research done in the direction of adaptive focused crawling. A detailed study on the use of InfoSpiders for crawling and using a mixture of evolutionary algorithms and reinforcement learning for information retrieval has been discussed by F. Menczer et al [1].

Micarelli et al [2] provides a detailed study on focused crawling which explores various methods of topical or focused crawling available and collaborates studies by several sources.

### 4 Architecture

#### 4.1 Crawler Components

A crawling agent contains the following components:

1. **Vector of search keywords**: These keywords denote the relevant topics at hand. We can use natural language processing techniques to add words to the
2. **Set of weights**: These are randomly assigned. It is required later for the learning algorithm.

3. **Lifetime**: This essentially denotes the maximum number of pages that a crawler can visit.

4. **Energy**: This is determined by the fitness function. This is a measure of how successful a particular agent is.

5. **URL Frontier**: A queue of URLs which need to be traversed. Initially they contain a set of seed URLs.

6. **Bias parameter**: This provides a bias and is defined as a measure of the reward or penalty given to the agent based on its performance. It gives more freedom for agents to explore 'bad URLs.

7. **Genotype**: The set of search terms, the learning weights as well as the Bias parameter constitutes the genotype of the crawler agent.

8. **Energy Threshold**: A threshold for the energy. This determines the out-come of the crawler after its lifetime is over i.e. it crawls the maximum number of pages it is allowed to.

9. **Network Threshold**: A threshold for the output of the neural network. If the result of the activation function is greater than Network Threshold it returns a 1 else 0.

10. **Index-map**: It is a hash-map which stores the list of URLs already visited by the crawler to prevent visiting the same URL multiple times.

Figure 2 shows the genotype of a crawler. B is the bias parameter, k1,k2.. are keywords in search set and w1,w2.. are the learning weights.

### 4.2 Learning component - Artificial Neural Network

This is a feed-forward neural network with a sigmoid activation function which is easily configurable. For sake of simplicity a two layer neural network can be used with just one input and one output layer. In order to capture more abstractions one or more hidden layers can be added to the neural network. However increase the number of layers can lead to increase in computation power required. For good results, one hidden layer is usually included.

The input to the neural network are functions of each keyword in the set of search terms and the output is a real value. Thus, the number of input neurons is number of terms in the set of search terms . The set of weights which are part of the Genotype gets updated continuously while the neural network is trained.
5 Initialization

This is the first step. We can start with a single agent crawler or multi-agent crawlers. Multi-agent crawlers have a distinct advantage of being able to run in parallel to each other and also reinforce the feedback system used by other agents and leverage their inference on relevance of web pages.

The set of search keywords are defined and the set of weights are assigned random values. All other components are defined as per user requirements. The URL frontier is populated with certain seed URLs. It should be noted that Bias parameter is to be valued accordingly so that initially a web crawler has freedom to explore a large set of pages while it is learning parameters.

6 Crawling

6.1 Term extraction and weighing

This section explains how we retrieve information from web page which would be used as input to the neural network which would in turn evaluate a link. Each keyword within the search term vector is assigned a unique index. This index corresponds to the input to the neural network and is common across all the crawling agents.

For each search keyword we need to weigh it according to how relevant that particular keyword is to a particular link in the document. We use the following formula.

\[ KW = \sum (1.0/D) \]

D = Measure of proximity of particular instance of keyword from the link i.e the word distance.

This is sum for every instance of the keyword in the body of the web page. Keywords located within the anchor text will have the value of D as 1. We will refer to KW as the keyword weight. This would be the input to the neural network along with the weights which are part of the crawler's genotype.

It should be noted that there are several other heuristics to evaluate the relevancy of a link. Also for a large web page to save on resources we only take into account keyword occurrences within the same paragraph of the link. From the above equation we can easily deduce that keywords which are located closer to the url anchor text have a higher relevance towards the link.

6.2 Fitness function

The fitness function describes the change in energy as a result of choosing a link.

\[ \text{FITNESS} = \text{benefit()} - \text{cost()} \]

(2)
Benefit shows how relevant a new page is to the search keywords. Benefit is calculated as
\[
\text{benefit}() = \sum \left( \frac{\text{IN}}{\text{TN}} \right)
\]  
(3)

IN : Total number of occurrences of the keyword in the new page
TN : Total number of terms in the new page

Cost shows the expense or rather how relevant the previous page is compared to the new page.
\[
\text{cost}() = \sum \left( \frac{\text{IO}}{\text{TO}} \right)
\]  
(4)

IO : Total number of occurrences of the keyword in the old or parent page
TO : Total number of terms in the old or parent page

Thus the overall fitness function calculates in essence the relevance of a newly fetched page in relation to the parent page based on number of instances of occurrence of each keyword in the search vector as well as the total number of terms in the document. Based on the FITNESS value, a crawler’s energy can increase or decrease in value.

6.3 Learning

For each link on the web page we obtain the KW as calculated in equation (1) above for every search keyword and we feed it into the neural network for the crawler. The output layer is a single neuron which gives the expected benefit of traversing the particular link. If the output is 1 assuming sigmoid activation function, then the network predicts that the URL might be relevant.

During training phase, the actual benefit of new page is extracted as a label which can be used to train the neural network and update the weights accordingly by backpropogation using stochastic gradient descent algorithm.
Once the crawler has been sufficiently trained, we make a call on whether to traverse the link or not based on the output of the neural network. If the output is greater than a particular threshold we can traverse the link and update the energy by recalculating fitness function and incrementing the bias parameter. However if the neural network output is less than the threshold, it is likely than we would be traversing a relatively irrelevant link. But, to allow for experimentation if the bias parameter is greater than 0 we allow the crawler to traverse the link in the hope that it might be relevant to our search and decrement the bias parameter accordingly. This is a technique called tunneling[8] which refers to going through few 'bad' pages to get to a 'good' page.

During the testing phase an output of 1 would lead to exploring the particular link. The keyword ratio described in equation (3) is stored for each URL visited. While fetching each new page the energy of the crawler is updated as the difference between keyword ratio of the present page from that of its parent page. Once a crawlers energy falls to 0 or below or it has crawled the maximum number of pages as defined in its LIFETIME component, the crawler enters the termination stage.

7. Termination

A crawling agent is terminated if any of the below conditions are satisfied -
1. It has explored the maximum number of pages it can i.e. it has completed its lifetime.
2. It's energy as denoted by the FITNESS function has been reduced to zero or negative.

The former condition indicates that the crawler has crawled enough pages and we need to make a decision regarding its usefulness. The latter condition indicates that the crawler has been performing poorly and that it needs to be terminated.

Figure 3 is an illustration of the neural network learning component for the crawler.
Figure 4 demonstrates how a fully-trained crawling agent works.
The success of a multi-agent crawling system can be defined by the fraction of agents that are recalled because of loss of energy. Once a crawler has completed its lifetime we must make a decision whether it should be allowed to reproduce i.e. spawn new crawler(s) or whether it should be terminated. If its final energy is greater than the Energy Threshold which is part of crawler component then it is allowed to spawn a new agent. The child agent inherits all the components of the parent agent excluding the energy which is set to a predetermined value which is common for all crawlers.

7.1 Mutations

To include some randomness we can introduce mutations for all child agents. The intention behind a mutation is to allow a crawling agent to work independently so its performance is not too similar to that of its parent. Mutations are made in the hope that it could result in more optimal solutions. We can introduce mutation by slightly tweaking the crawler's neural network weights W such as adding 0.5 to each of the weights of the offspring. There is no set or correct method to cause a mutation and it is more often than not a result of trial and error.

8. Evaluation and Conclusion

This approach was evaluated on a small self-made data set resembling Wikipedia content. I used around 200 pages for the experiment. I divided the whole set as follows: 80 percent as training set and remaining 20 percent as the testing set. I used both termination rate or the fraction of agents I had to terminate because of low energy as a metric to calculate efficiency of the system. The other metric I used was the precision or ratio of pages with benefit above 0.1 i.e. number of relevant terms should be greater than 0.1 times the total number of terms across all pages extracted. A low termination rate is an indication of the success of the learning algorithm. Using one crawler initially and expanding it to three I found out that termination rate and precision rate does not change much with multiple agents. However with more agents you would be able to crawl more pages in less amount of time. Thus, it is essentially a trade-off between time and memory. There are works on parallel or multi-threaded implementation of genetic algorithm based crawlers[3].

While there are larger datasets available, I had to restrict myself to a few due to resource constraints. However, compared to a generic crawler which fetched data from every page, a focused adaptive crawler is able to fetch desired information at a much faster rate. While implementing several crawlers in parallel we would be needed a central control mechanism to coordinate them and pre-vent redundancy.

The essence of this approach is that as a crawler traverses more and more pages it learns more and is ability to predict with higher accuracy the relevance of a link without traversing it. It also uses a mechanism similar to that of reinforcement learning to correct itself and the whole process can be left largely unsupervised.
9. References

2. Alessandro Micarelli et al, Adaptive Focused Crawling, Roma Tre University
6. Focused web crawling for e-learning content, seminarsonly.com