Judge a Book by its Cover: Conservative Focused Crawling under Resource Constraints

Shehzaad Dhuliawala Arjun Atreya V
Ravi Kumar Yadav Pushpak Bhattacharyya
Center for Indian Language Technology, CSE Department
IIT Bombay, Mumbai, India,
{shehzaadzd,arjun,ravi,pb}@cse.iitb.ac.in

Abstract

In this paper, we propose a domain specific crawler that decides the domain relevance of a URL without downloading the page. In contrast, a focused crawler relies on the content of the page to make the same decision. To achieve this, we use a classifier model which harnesses features such as the page's URL and its parents' information to score a page. The classifier model is incrementally trained at each depth in order to learn the facets of the domain. Our approach modifies the focused crawler by circumventing the need for extra resource usage in terms of bandwidth. We test the performance of our approach on *Wikipedia* data. Our Conservative Focused Crawler (CFC) shows a performance equivalent to that of a focused crawler (skyline system) with an average resource usage reduction of $\approx$30% across two domains *viz*., *tourism* and *sports*.

1 Introduction

Crawling is a process of fetching documents iteratively from the web. While generic web search engines need to crawl and index a huge number of documents, there exist other search systems like enterprise search, domain search, patent search *etc*., that concentrate on a particular section of the web. The crawling infrastructure required to process the entire web is huge and most small scale and academic organizations cannot afford them. Even though open source crawling frameworks help in crawling the web, resource constraints limit the number of documents being crawled. Most generic web crawlers employ a breadth-first approach for fetching documents from the web. A set of seed URLs which are manually fed to the crawler are processed in the first depth and then the outlinks are processed for subsequent depths. This process continues for multiple depths to crawl more documents. Every document crawled needs to be processed before retrieval. Processing a document involves the following sequence of steps:

Fetching The process of downloading the document from the web. This is a bandwidth consuming task.

Parsing The process of extracting clean content from a web document. Parsing would also involve extracting outlinks, language, domain information and other meta information from the document. This is a CPU intensive task.

Indexing The process of storing the document in a searchable format. This is a memory intensive task.

In the context of crawling under resource constraints, it is important to carefully choose the appropriate outlinks to be crawled at each depth. Choosing the outlinks before actually fetching them involves taking a decision based on the outlink and its parents' characteristics (which are already fetched in the previous depths).

The organization of the paper is follows; in Section 2, we talk about related work. Section 3 describes the problem statement. Our approach is discussed in Section 4 while Section 5 details the experimental setup, 6 describes our skyline system. The analysis of the results obtained are discussed section 7. We conclude our work in section 8.

2 Related Work

(Chakrabarti et al., 1999) is the first work which talks about crawling a topic specific set of webpages. Classifying a document based on the URL and its content is discussed in (Kan, 2004). For the problem of classifying a document based on

D S Sharma, R Sangal and E Sherly. Proc. of the 12th Intl. Conference on Natural Language Processing, pages 166–171,
Trivandrum, India. December 2015.

URL only, some of the features mentioned in this work is found to be useful. The closest work to our approach is by Aggarwal et al. (2001), which provides an idea about categorizing the domain of a URL while crawling but uses a simple weighted addition for scoring. This paper does not address the adaptability of the weights across domains.

Kan and Thi (2005) added URL features, component length, content, orthography, token sequence and precedence to model URL. The resulting features, used in supervised maximum entropy modeling, significantly improve over existing URL features. Baykan et al. (2009) showed that machine learning approach outperforms dictionary based approach for URL classification with n-grams as features.

Jamali et al. (2006) discusses use of link structure of web and content for focused crawler. They use a classifier to compute similarity of given web page to the topic. Qi and Davison (2009) describes multiple content based features that can be used for page classification in focused crawling. Priyatam et al. (2013) worked with URL based approach and tokenized them using n-grams to achieve high precision for Indian languages. The other works that use URL tokens as features for document classification are Shih and Karger (2004), Hernandez et al. (2012) and Anastacio et al. (2009).

3 Problem Statement

In this work, we address the problem of excess resource usage while classifying the document as in-domain or out-domain during crawling.

The formal problem statement is as follows:

Rank the URLs to be crawled in a given depth based on the URL and its parents' information gathered over previous depths.

4 Our Approach

We propose a light-weight focused crawler which selectively discards the out-domain URLs without fetching them. This is done by scoring and prioritizing the URLs to be crawled in each iteration. Our approach gives a set of in-domain URLs to be fetched at a given depth. As several links get rejected before the fetching stage, the cost of fetching them is averted.

Figure 1 describes the architecture of our proposed crawler. In a generic crawling pipeline, we integrate a classification module that assigns a confidence score to each outlink. The classification module uses the outlink's URL along with some features of the outlink's parents to decide the domain of the outlink. We prioritize the URLs based on these scores that would be fetched in the subsequent depths. The initial classification module is trained on a small set of outlinks. At every depth, the model is incrementally trained using the in-domain and out-domain outlinks encountered in that depth. As shown in figure 1, the initial model, M_0 is trained using a small set of initial *positive* and *negative* outlinks, P_0 and N_0. In depth $i+1$, the model of the initial depth M_i is incrementally trained using $P_i + 1$ and $N_i + 1$.

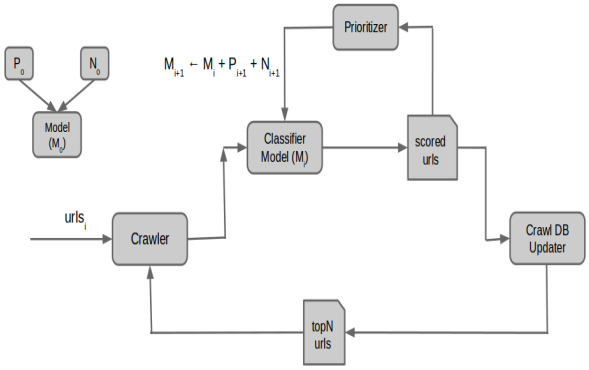

Figure 1: The system architecture

4.1 Scoring

The process of assessing the confidence of an outlink pointing to an in-domain page involves the creation of a feature vector for every outlink. This feature vector is then fed to a binary classifier which then assigns a class(in-domain or out-domain) to the outlink along with a confidence score.

The scoring model incorporates four sets of tokens. These sets are called token pools. Two pools collate tokens of URLs and anchor texts linking to all the in-domain pages crawled till the current depth, while the other two pools comprise of tokens of URLs and anchor texts relating to out-domain pages. The pools also hold a weight for each token based on its frequency of occurrence.

In order to score every outlink the following features are used:

- The weighted overlap between the tokens in the outlink's URL with the *positive URL token pool*
- The weighted overlap between the tokens in

the outlink's anchor text with the *positive anchor text token pool*

- The weighted overlap between the tokens in the outlink's URL with the *negative URL token pool*
- The weighted overlap between the tokens in the outlink's anchor text with the *negative anchor text token pool*
- The weighted overlap between the tokens in the outlink's parents' URL with the *positive URL token pool*
- The weighted overlap between the tokens in the outlink's parents' URL with the *negative URL token pool*
- The average score of the outlink's parents
- The total number of in-domain parents

We use the concept of weighted overlap with pools over a *bag-of-words* approach is because the list of tokens relating to a domain is not exhaustive. These pools are updated at each depth with newer tokens discovered. A weighted overlap is chosen to accommodate for non-domain specific words. Tokens with no domain information would be present in equal amounts in both the negative and positive pools. Words which provide information about the domain and indicate that the outlink is in-domain would have a higher weight in the positive token pool.

4.2 Facets of a domain

A domain cannot be only categorized using a small exhaustive list of words. A domain often has several facets. Hence, trying to train a complete classifier for domain identification is often a huge task. For example, the domain *sports* may have subdomains such as:

- Football
- Cricket
- Hockey

Quite obviously, the terms associated with the three above sub-domains of the domain of *sports* will not be the same. A classifier model which is trained using the terms of *Football* may be unable to recognize the terms of another sub-domain like *Cricket*. To overcome this we use an online classification model. The classification model is incrementally trained at every depth as newer outlinks are discovered. The online training is described in section 4.3.

4.3 Online Learning

Over multiple iterations the performance of the classifier may decrease as new tokens are encountered; online learning, in such a scenario, seems like a viable option (Priyatam et al., 2013). The incremental training aspect of the model manifests two major benefits. First, it allows for the initial set of training data to be relatively small (Zheng et al., 2013). Secondly, it prevents the accuracy from dipping over depths as new tokens are encountered.

Over the iterations, the size of the pools grow. The classifier uses the new training data to incrementally learn. There can be multiple combinations to create this training set. We selectively train our classifier with only a small fraction of the pages.

The risk of an incremental classifier, however, comes in the form of topic drift. A small amount of corruption in the training pool, can cause the classifier to accept several out-domain examples for training and hence decreasing the accuracy. As the model is further trained, it begins to assert its belief.

5 Experimental Setup

We customize the open source crawler, *Nutch 1.7* for our experimental setup.

We replace the scoring mechanism by our model. For the classifier we use an online Naive Bayes classifier. Given that we begin our crawl with a very small set of initial training data, we employ a Naive Bayes classifier.

5.1 Generating initial pools and training data

CFC requires a very small data set to initially train itself. The crawler starts with a single seed URL (The content page of the category). This crawler initially runs until it crawls 400 in-domain and 400 out-domain pages. This information is used to populate the initial set of pools and create the initial training file.

5.2 Crawling

We use Apache-Nutch-1.7 as the basic crawl frame work. The single seed URL for each of the domains is listed in table 2. The crawlers (CFC and Skyline) are instructed to fetch 100 in-domain URLs in every depth and the crawl is run for a total of 20 depths.

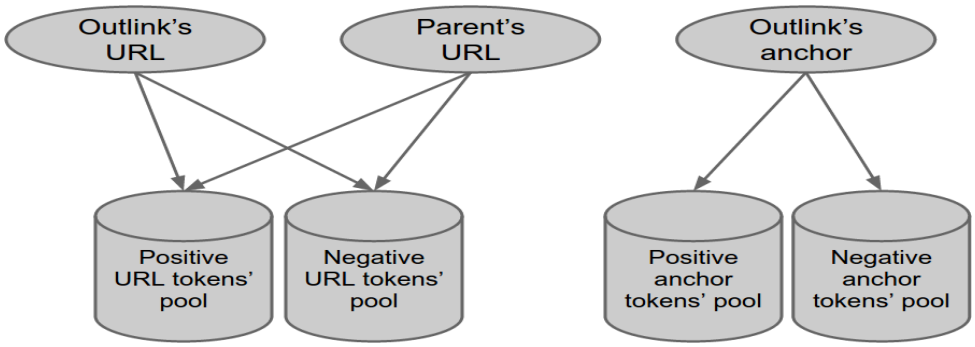

Figure 2: Scoring the token with the token pools

5.3 Data

We use English Wikipedia as our data set for experiments. Wikipedia was specifically chosen as pages are highly linked, so offering a highly connected web graph. Secondly, Wikipedia pages are category tagged. This allows for gold data to be effectively identified. In our experiments we aim to crawl over two topic domains: Indian Tourism and sports.

5.3.1 Preparing Gold Data

Each document in Wikipedia is associated with a set of categories. Each category represents a collection of documents belonging to a particular topic. A category may have one or more subcategories resulting in a hierarchical structure.

For tourism we choose the category **Tourism in India**[1] and for sports we choose **Sports**[2]. Along with the chosen topic root categories, we choose the sub categories in their respective category trees.

The category lists obtained are further manually pruned to refine our gold category set.

6 Skyline

For a skyline, we use the idea behind a generic focused crawler. Here every page is downloaded before the crawler verifies its domain. The Skyline Crawler is built by inserting a domain identifier 6.1 into the pipeline of a generic crawler. The domain identifier downloads each page and asserts its domain. This is a bandwidth intensive process.

6.1 Domain Identifier

For the creation of our Skyline Crawler, we create a binary class, bag of words text classifier.

[1]https://en.wikipedia.org/wiki/Category:Tourism
[2]https://en.wikipedia.org/wiki/Category:Sports

Domain	Accuracy
Indian Tourism	95.4
Sports	94.8

Table 1: Domain Identifier accuracies

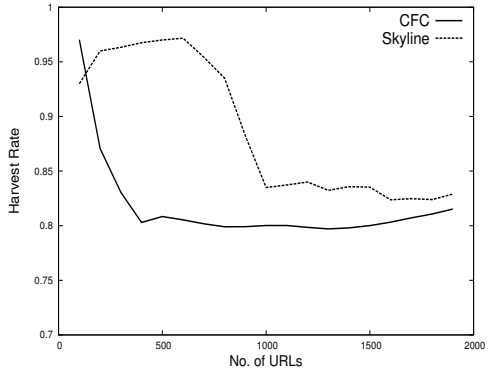

Figure 3: Harvest Rate: Indian Tourism

This classifier was trained over pre-crawled gold topic data. The classifier was trained using 1000 Wikipedia pages belonging to the required domain along with 1000 randomly selected general category documents. The classifier used was a C-SVC type SVM. The five-fold accuracies obtained by the classifiers for the domains of Indian Tourism and sports are described in table 1.

7 Results

7.1 Harvest-rate

This section discusses the performance of CFC when pitted against the Skyline crawler 6. We observe that CFC's performance is close to skyline accuracy.

The metric used to compare the crawlers performance is the *harvest-rate* obtained. (Li et al., 2008) defines *harvest rate* as:

Domain	Initial seed URL
Indian Tourism	https://en.wikipedia.org/wiki/Tourism_in_India
Sports	https://en.wikipedia.org/wiki/Sport

Table 2: Seed URLs

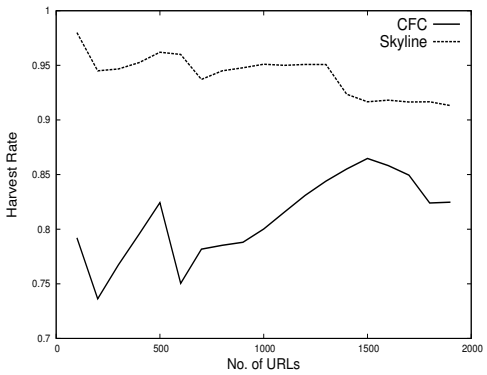

Figure 4: Harvest Rate: Sports

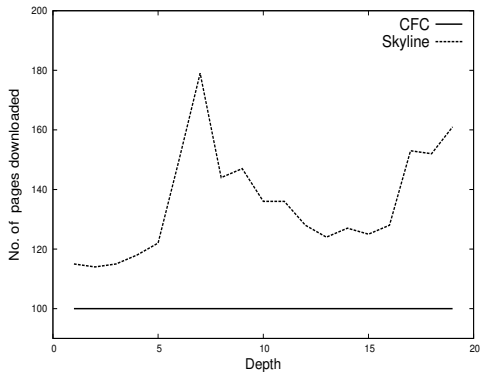

Figure 5: Resources: Indian Tourism

Crawler	Indian Tourism	Sports
CFC	0.82	0.82
Skyline	0.83	0.91

Table 3: harvest rate comparison

> The harvest rate represents the fraction of web pages crawled that satisfy the crawling target R among the crawled pages P. If the harvest ratio is high, it means the focused crawler can crawl the relevant web pages effectively; otherwise, it means the focused crawler spends a lot of time eliminating irrelevant pages, and it may be better to use another crawler instead. Hence, a high harvest rate is a sign of a good crawling run."

The harvest rate for both *Indian tourism* and *sports* domain is depicted in figures 3 and 4. In figure 3, CFC starts with a very high harvest rate. However, this doesn't seem to be mirrored in figure 4. We feel this is just by chance as the harvest rate seems to soon stabilize. The graphs (figures 3 and 4) show that the Skyline outperforms CFC. However, as the number of URLs increase, the domain identifier is unable to maintain its high accuracy. A likely reason for this is that, over time, the domain identifier gets several pages of different facets of the domain which it hasn't been trained on. CFC, however, doesn't suffer from this dip as the online classifier model is constantly updated with new terms and tokens. Towards the end of the graphs, we notice that CFC and Skyline reach very close.

7.2 Resources

In this section we aim to compare CFC with the Skyline crawlers based on the amounts of resources required. Bandwidth (for fetching) is a major resource involved in crawling. All other resources like processing power (for parsing), memory (for indexing) depends on the number of pages fetched. Hence, we evaluate our crawling performance in terms of bandwidth usage.

The Skyline crawler requires an additional amount of internet bandwidth owing to the fact that it needs to download a page to determine the domain of the page. While CFC scores an outlink before the page is downloaded, only a fixed number of pages are downloaded at every depth. Downloading of extra pages increases the usage of bandwidth and also increases the time required for crawling.

Figures 5 and 6 show that the number of pages which need to judge the domain is much higher. The figures don't indicate any specific trend, however we notice that the average number of pages needed to be downloaded for **sports** is relatively higher than that for tourism. The maximum pages which needed to be downloaded in any depth is around 180 for each of the domains.

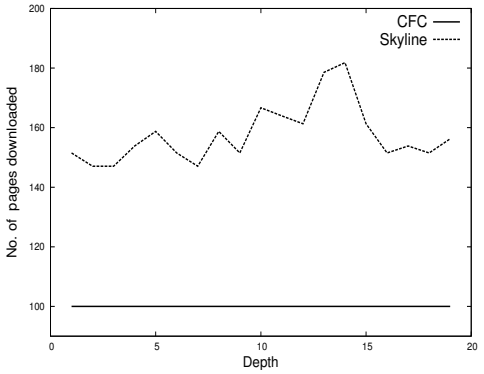

Figure 6: Resources: Sports

Crawler	Indian Tourism	Sports
CFC	137.4	157.5
Skyline	100	100

Table 4: Average pages downloaded

The metric indicating *number of pages downloaded* does not do complete justice when trying to judge the crawler on its resource usage. Several offline factors also do come into play. The Skyline crawler needs to train an SVM classifier on a huge number of documents. This clearly uses more computing power than training a Naive Bayes classifier on a few set of numeric features. The Skyline crawler also needs the huge set of in-domain documents which it can be trained upon.

8 Conclusion

In this paper, we developed a preemptive crawler that selectively picks a set of in-domain URLs to be crawled in each depth. Using the outlink's URL and its parent's information to determine a page's domain, yields results comparable to that of a fully fledged focused crawler. Our experimental results on *tourism* and *sports* domains validate reduced bandwidth usage of $\approx 30\%$. As a future work, we aim to evaluate our system on the open web which has a more complex web-graph as compared to *Wikipedia*.

References

Charu C Aggarwal, Fatima Al-Garawi, and Philip S Yu. 2001. Intelligent crawling on the world wide web with arbitrary predicates. In *Proceedings of the 10th international conference on World Wide Web*, pages 96–105. ACM.

Ivo Anastácio, Bruno Martins, and Pável Calado. 2009. Classifying documents according to locational relevance. In *Progress in Artificial Intelligence*, pages 598–609. Springer.

Eda Baykan, Monika Henzinger, Ludmila Marian, and Ingmar Weber. 2009. Purely url-based topic classification. In *Proceedings of the 18th international conference on World wide web*, pages 1109–1110. ACM.

Soumen Chakrabarti, Martin Van den Berg, and Byron Dom. 1999. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11):1623–1640.

Inma Hernández, Carlos R Rivero, David Ruiz, and Rafael Corchuelo. 2012. A statistical approach to url-based web page clustering. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 525–526. ACM.

Mohsen Jamali, Hassan Sayyadi, Babak Bagheri Hariri, and Hassan Abolhassani. 2006. A method for focused crawling using combination of link structure and content similarity. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 753–756. IEEE Computer Society.

Min-Yen Kan and Hoang Oanh Nguyen Thi. 2005. Fast webpage classification using url features. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 325–326. ACM.

Min-Yen Kan. 2004. Web page classification without the web page. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 262–263. ACM.

Hang Li, Ting Liu, Wei-Ying Ma, Tetsuya Sakai, Kam-Fai Wong, and Guodong Zhou. 2008. *Information Retrieval Technology: 4th Asia Information Retrieval Symposium, AIRS 2008, Harbin, China, January 15-18, 2008, Revised Selected Papers*, volume 4993. Springer.

Pattisapu Nikhil Priyatam, Srinivasan Iyengar, Krish Perumal, and Vasudeva Varma. 2013. Dont use a lot when little will do: Genre identification using urls. *Research in Computing Science*, 70:207–218.

Xiaoguang Qi and Brian D Davison. 2009. Web page classification: Features and algorithms. *ACM Computing Surveys (CSUR)*, 41(2):12.

Lawrence Kai Shih and David R Karger. 2004. Using urls and table layout for web classification tasks. In *Proceedings of the 13th international conference on World Wide Web*, pages 193–202. ACM.

Jun Zheng, Furao Shen, Hongjun Fan, and Jinxi Zhao. 2013. An online incremental learning support vector machine for large-scale data. *Neural Computing and Applications*, 22(5):1023–1035.