SIGCSE: U: Focused Retrieval of University Course Descriptions from Highly Variable Sources

Thomas Effland[*]

SUNY, University at Buffalo

tom.effland@gmail.com

Abstract

Finding topically relevant content from sparse disparate sources on the Web requires robust techniques. A focused web crawler is a type of crawler that attempts to make predictions about page relevance and traverse the web efficiently to retrieve relevant information. In this work, we design and test a novel framework of focused crawling tailored to extracting semantically relevant information from disparate seed domains with highly variant structure that do not reference each other. We utilize machine learning techniques by employing two separate random forest regressors that rank the current page and potential relevance gain of hyper-links to navigate sites. We use a novel reformulation of page relevance as the normalized link distance to efficiently tunnel through irrelevant pages and reach target pages with close to optimal paths for domains with large inter- and intra-site variability. We evaluate this system on a concrete problem: retrieving relevant course description and information pages from many university websites with little training data. We find that we are able to train accurate regressors using self-training. We achieve an 80% *harvest rate within* 50 *pages for* 4 *of the* 5 *sites tested for retrieval efficiency in practice .*

I. Problem & Motivation

The exponential growth of the World Wide Web has given rise to an explosion of publicly accessible data in the form of unstructured natural language text and semi-structured hyper-text. Often we are interested in semantically similar content that is available in many different web domains because the aggregate of the content can contain richer information than each individual datum. The connections between different data sources are commonly non-existent and a user is forced to search for individual data items separately. However, finding and retrieving this data manually is often unrealistic at scale.

Topical crawlers are the specialization of traditional search engines or crawlers to finding relevant pages across domains: a form of "smart" search. A topical crawler makes a basic assumption of *topical locality* [3] within the web that asserts that relevant pages are typically close together between sites, i.e. there is a short link distance between them which enables traversal across sites to find relevant content. Here we address a different reformulation of the topical crawling problem where the topical locality assumption does not necessarily hold:

> *How can we find semantically similar information from separate known sources that do not reference each other?*

In this work we are motivated by the example application of retrieving course information and descriptions from multiple university websites. This specific task is difficult for multiple reasons:

- Retrieving content on one site doesn't directly lead to content on another site, i.e. no inter-site topical locality.
- University websites are highly varying in structure from one institution to the next, so reaching the relevant information with a general, but efficient technique is difficult.
- The highly variate structure makes identifying a relevant seed page difficult. Without a relevant seed page, we are forced to start at the root domain page, e.g. `www.buffalo.edu`.
- Course information in university websites is sparse. Thus we must significantly restrict and navigate the search space in an efficient manner.

In many ways, this problem is akin to the automation of mimicking a user's browsing decision-making process for finding relevant content on many sites by starting at the sites' root pages: at each page, we want

[*]Partially supported by NSF Grant DUE-CCLI-0920335

to identify and follow the link which will lead to our target page in the least number of steps.

II. Background & Related Work

The rapid growth of the World Wide Web presents many challenges to finding specific topical information for traditional web crawlers. Often a crawler is targeted towards a particular topic and is unable to find topical information in an efficient manner. A *focused* crawler attempts to make decisions on which pages to crawl based on a classifier trained to estimate the relevance of a page [10]. Typically, a focused crawler represents a page using the "Vector Space Model" in which a page is represented by a vector of features (often words) so that a traditional classifier may be applied.

There are multiple challenges that arise for focused crawlers. The main issue in the context of our problem is *tunneling*, in which a relevant page may only be reachable by traversing irrelevant pages. [9] used reinforcement learning to assign credit to irrelevant pages on the relevant path. [5] address the issue by using a maximum depth counter that resets when the crawler reaches a relevant page. [7] trained a set of Naive-Bayes classifiers that attempt to predict the link-distance the current page is from a target page; This is called "Context-Focused" crawling. [2] evaluates both page content and link structure separately to predict the best links to follow. Both [6] and [12] used Named-Entity Recognizers to extract named-entities as features in addition to words.

III. Uniqueness of Approach

In this work we build on previous works for developing efficient focused crawlers and apply the techniques to a crawler that can traverse highly variate, disparate sources where the topical locality assumption from one site to the next does not necessarily hold. Our framework is designed to develop an efficient system that utilizes two machine learning regressors to make relevance predictions for pages and their links with limited training data. This is presented in four parts: the page feature representation (input features), the page relevance representation (target variable), the training phase, and the deployment phase.

Page Representation

Web page source code provides semi-structured information in the form of HTML. We convert this original data into a feature vector using the vector space model. We use HTML tree parsing to split a page P into a content vector $\vec{p}$ and a set of url vectors $U = \{\vec{u}_1, ..., \vec{u}_k\}$, where k is the number of `<a>` tags on the page. We then quantify these two representational elements separately using information retrieval techniques.

Representing the Page Content

To represent the page content, we take the following steps:

1. Convert all the text inside the `<body>`, `<title>`, and `<a>` tags into vectors $\vec{b}, \vec{t}, \vec{a}$ of terms by splitting on whitespace.
2. Remove special characters and English stop-words using NLTK [8].
3. Stem the vectors using WordNetLemmatizer [8].
4. Expand the vectors by adding in bigrams (contiguous word pairs) of the terms.
5. Take the TF-IDF [11] of the vectors to get the relative frequencies of the terms within the document and the vocabulary.
6. For the `<body>` vector only: use Latent Semantic Analysis [4] to embed the vector in a lower dimensional semantic space. This greatly reduces the size of the input vector.
7. Concatenate these three transformed vectors as the page feature vector $\vec{p} =< \vec{b}', \vec{t}', \vec{a}' >$ [1].

Thus we have a numerical feature vector that represents the content of a page. We note that **this representation is generic** with no domain-specific engineered features.

Representing the Link Content

To represent each url $\vec{u} \in U$ we extract and segment its `href` attribute to get a vector of terms $\vec{h}$. We also extract the anchor-text within the `<a>` tag as we did for the page to get the vector $\vec{a}$. We then form the final representation $\vec{u} =< \vec{h}', \vec{a}' >$ using the same sequence of steps as described above for the page representation (excluding step 6).

Defining a Relevance Metric

To reach relevant pages from the irrelevant root url of a site, we must identify and traverse many irrelevant pages that lie on a path to the relevant content. We also need to address this issue in a way that is robust to differing path lengths. Thus we define the target relevance

[1]The length of $\vec{p}$ is constant and ensured through the TF-IDF and LSA transformations.

R of a page P to be the normalized link distance from P to a target page T, with the link distance from the starting page S to T as the normalization factor. Formally,

$$R(P) = 1 - \frac{LinkDist(P,T)}{LinkDist(S,T)} \tag{1}$$

Intuitively this metric can be thought of as what fraction of the total path length is this page P from our target page T. In practice a page may lie on many relevant paths and in this case we average all of its relevance scores for the final score.

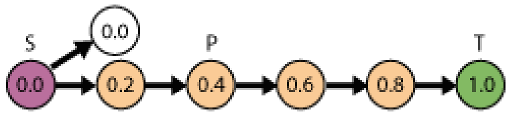

Figure 1: Here we graphically show the relevance scores of all of the pages on a relevant path. The source page in purple, the irrelevant pages in yellow, the target in green, and a completely irrelevant page in white.

This relevance score may now be used in combination with the corresponding page feature vector and machine learning methods to make predictions about the relevance of a page. We also define the relevance of each link $\vec{u}$ to be the relevance of the page the link leads to. Thus we may use the links to make predictions about which pages to traverse to next.

Training the System

Since the goal of this work is to automate a manual process, we approach the problem of learning accurate regressors with as little manually gathered training data as possible. We utilize a semi-supervised learning technique, self-training [13], to iteratively improve our regressors' performance as follows:

1. The user manually marks a few sample traversals from the start page to a target page as a sequence of urls.
2. A training crawler follows these paths and downloads each page on the path as labeled data. The crawler also downloads the neighboring pages as unlabeled data.
3. The features of the labeled pages and links are extracted along with their corresponding relevance metrics.
4. Two random forest regressors [1] [2] are trained on these data so we can make predictions of page and url relevance as defined in equation 1.
5. The regressors are used to make predictions on the unlabeled pages. The most confident predictions (the highest and lowest ranked 25 pages) are then given labels of 1.0 and 0.0, respectively. All unlabeled pages lying on paths to the new target pages are labeled with their calculated relevancies.
6. Iteratively loop steps 2-5 until regressors reach desired accuracy or the unlabeled dataset is used up.

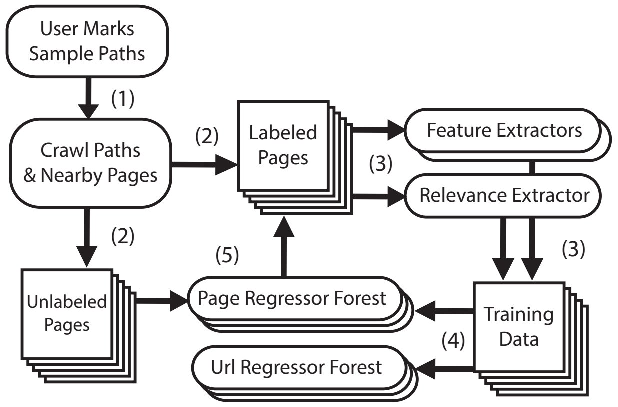

Figure 2: Flowchart of self-training procedure for the regressors. This iterative approach allows for training of accurate regressors with little manually-gathered training data.

Using this technique, we are able to automatically train accurate regressors with little manually labeled training data.

Deploying the System

After training the regressors, we can deploy them to retrieve pages of interest on other sites.

Beginning at the starting url, we traverse a site by featurizing the urls on a page and using the url regressor to make predictions of the relevance for the page the urls lead to. We then rank the urls by highest predicted relevance and insert them into a priority queue. To choose the next page to crawl, we simply pop from the queue. Thus we use a greedy approach in traversal by consistently following the link of highest predicted relevance in search of relevant pages. This has the advantage of being able to significantly restrict our search space in a site.

[2] We choose to use Random Forests as our regressors because they have been shown to be robust to sparse, highly-variable data. [1]

At each page, we also use the page regressor to make a prediction about the page content's relevance. If the predicted relevance is above a threshold (in practice we use .85), then we classify the page as relevant and add it to the set of gathered pages.

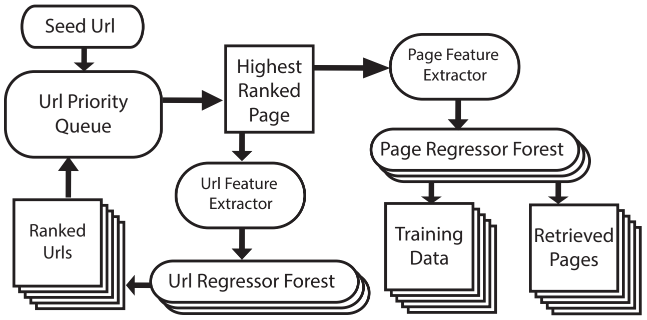

Figure 3: Flowchart of the deployment procedure. Starting at the root url, the system greedily follows the highest ranked url to traverse the sites efficiently. At each page, the relevance is predicted and the page is saved if this prediction is above a certain threshold.

To continually grow the training set throughout deployment, we utilize active-learning [14] to query the user for input on pages with high discrepancy between regressors.

IV. Results & Contributions

To evaluate the system, we present two metrics. We first evaluate the accuracy of the page and url regressors using mean absolute regression error. We measure this as a function of the size of the training data set as grown through self-training. We then evaluate the efficacy of the system in practice by measuring the harvest rate of retrieving relevant course description pages on university websites.

We tested on five university sites: `buffalo.edu`, `illinois.edu`, `bu.edu`, `washington.edu`, and `northwestern.edu` [3]. For each site, we marked ten sample traversal paths from the root url to course descriptions pages. In evaluation, we utilize hold-one-out methods by training the system on all sites except the site we test on.

Training Results

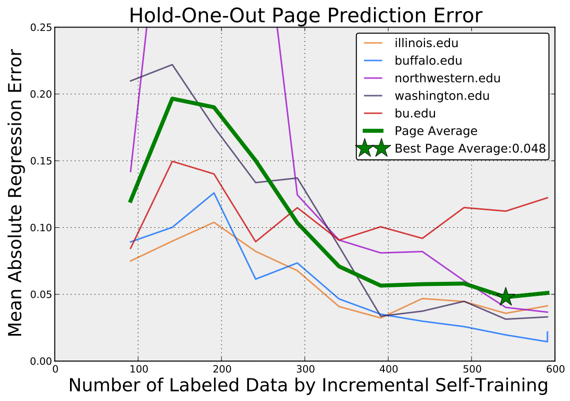

Figure 4: This plot shows mean absolute prediction error for the page regressor as we iteratively increase the training dataset using self training. Each university is represented in its school color and the average among all five in green. The star represents the lowest average of .048 at 542 pages. Here we see that initially the system is quite inaccurate, but is eventually able to make better predictions while training on other sites through self-training.

Using the 50 sample paths generates a little less than 100 initial training examples among the five sites. We then grow the training dataset approximately 50 pages per iteration using self-training. From the graph, we see that at about 550 training examples the average mean page prediction error among all five sites (represented by the thick green line) reaches .048. A similar plot of the mean url prediction error would show that the average among the sites reaches .052 for the same training set. We omit this plot due to space constraints.

Testing Results

We evaluate the efficacy of the trained system in retrieving relevant course descriptions pages using the standard metric in focused crawling, harvest rate. The harvest rate is the fraction of pages retrieved out of the pages visited so far, i.e. $HarvestRate = \frac{\#Retrieved}{\#Visited}$.

We again test on the previously mentioned five sites, using hold-one-out training methods. In the graph we see that we reach a harvest rate of 80% for 4 of 5 sites within 50 pages. This shows the system efficiently navigating to the content rich areas of the site. We also note that `illinois.edu` is less successful. This is because

[3] These five sites were chosen for evaluation because they each have a different organizational structure for accessing course descriptions, but still present the data in HTML instead of requiring database querying.

[4] A pseudo-relevant path has high initial predicted values, but never leads to a relevant page: a shortcoming of the greedy strategy.

the system initially followed a pseudo-relevant path [4] before discovering the correct region of the site. However even in this case, we find that the system is able to find relevant content in less than 200 visits.

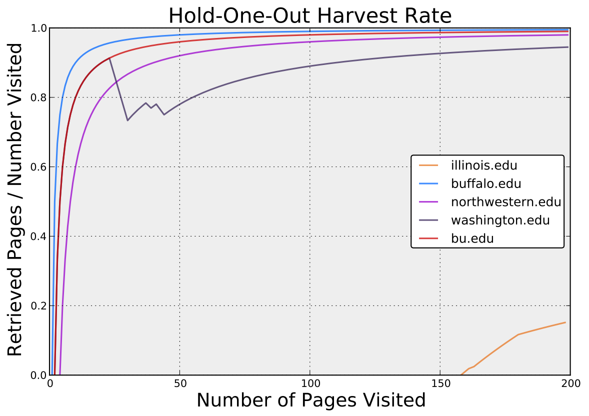

Figure 5: This plot shows the harvest rate of the system in deployment using hold-one-out training. We see that the system is able to achieve a harvest rate of 80% within the 50 pages crawled for 4 of 5 sites. We note that `illinois.edu` does not have the same success, as the system initially followed a pseudo-relevant branch. However, after exhausting this branch, the system does navigate to the course descriptions.

Conclusions and Impact

In this paper we have presented a novel focused crawling architecture tailored to retrieving semantically similar content from many highly variable disparate sites with no initial assumptions about site organization or topical locality. Using self-training, we require little manually-gathered data, yet are able to train accurate regressors to predict page and url relevance. The novel reformulation of page relevance as normalized link distance is key to addressing the issue of high organizational variation with a general scheme.

We evaluated the training stage of our system using mean absolute prediction error and showed that it significantly improved its regressors using self-training. We then evaluated the system in practice by measuring the harvest rate of the trained system. The system reached a harvest rate of 80% within 50 visits for 4 of 5 cases.

We note that this system has been designed as a general information retrieval framework and can be used in any scenario where the user wants to automate the process of collecting data from many disparate sources, but only has a list of the source domain names. In the future, this system could be significantly improved by utilizing crowd-sourcing services to generate considerably larger and more informative training datasets to produce extremely accurate regressors. Coupled with domain-specific information extraction tools, it could be used to automatically generate large databases of cross-site information.

References

[1] Breiman, L. Random forests. *Mach. Learn. 45*, 1 (Oct. 2001), 5–32.

[2] Chen, X., and Zhang, X. Hawk: A focused crawler with content and link analysis. In *e-Business Engineering, 2008. ICEBE'08. IEEE International Conference on* (2008), IEEE, pp. 677–680.

[3] Davison, B. D. Topical locality in the web. In *In Proceedings of the 23rd Annual International Conference on Research and Development in Information Retrieval (SIGIR 2000* (2000), ACM Press, pp. 272–279.

[4] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE 41*, 6 (1990), 391–407.

[5] Devi, P., and Thakur, R. Comprehensive review of web focused crawling.

[6] Di Pietro, G., Aliprandi, C., De Luca, A. E., Raffaelli, M., and Soru, T. Semantic crawling: An approach based on named entity recognition. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on* (2014), IEEE, pp. 695–699.

[7] Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., Gori, M., et al. Focused crawling using context graphs. In *VLDB* (2000), pp. 527–534.

[8] Loper, E., and Bird, S. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1* (Stroudsburg, PA, USA, 2002), ETMTNLP '02, Association for Computational Linguistics, pp. 63–70.

[9] McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. Automating the construction of internet portals with machine learning. *Information Retrieval 3*, 2 (2000), 127–163.

[10] Nasraoui, O. Web data mining: Exploring hyperlinks, contents, and usage data. *ACM SIGKDD Explorations Newsletter 10*, 2 (2008), 23–25.

[11] Rajaraman, A., and Ullman, J. D. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011.

[12] Samarawickrama, S., and Jayaratne, L. Focused web crawling using named entity recognition for narrow domains. *IJRET | DEC* (2012).

[13] Scudder, III, H. Probability of error of some adaptive pattern-recognition machines. *IEEE Trans. Inf. Theor. 11*, 3 (Sept. 2006), 363–371.

[14] Settles, B. Active learning literature survey. Tech. rep., 2010.