ROACH: Online Apprentice Critic Focused Crawling via CSS Cues and Reinforcement

Asitang Mishra[1], Chris A. Mattmann[1,2], Paul M. Ramirez[1], Wayne M. Burke[1]
chris.a.mattmann@jpl.nasa.gov

[1]Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109 USA

[2]Computer Science Department
University of Southern California
Los Angeles, CA 90089 USA

ABSTRACT

The Internet today is replete with forum sites that - in the context of the deep/dark web - are used to sell illicit goods, and to deal in illegal activities including human trafficking (HT). Working on the DARPA MEMEX project, our team collaborated with law enforcement to perform bulk analysis and crawl these sites and in doing so found a number of challenges. Forum sites contained pages with many links and little content and the ads - the rich source of content - were hidden behind link-dense hubs. To identify important links that led to ads, crawlers had to take advantage of CSS visual cues. As forum sites changed often, training a model offline would not be sufficient. We address these issues by creating ROACH, or Reinforcement-based, Online, Apprentice-Critic based focused crawling approacH. ROACH provides an online, adaptive crawling mechanism that employs static subject matter expert knowledge, with online learning based on a simplified version of reinforcement learning with back propagation. We use the widely popular apprentice-critic framework for performing this capability. ROACH is independent of any crawler implementation. The approach is scalable and accurate and overall provides better link relevancy scores than two baseline approaches including Apache Nutch. We evaluate ROACH on a dataset collected in the human trafficking domain from the DARPA MEMEX effort.

CCS CONCEPTS

• **Information systems** → **Information retrieval**; • **Applied computing** → *Computer forensics*;

KEYWORDS

Web Crawling, Focused Crawling, Information Retrieval

ACM Reference Format:
Asitang Mishra[1], Chris A. Mattmann[1,2], Paul M. Ramirez[1], Wayne M. Burke[1]. 2018. ROACH: Online Apprentice Critic Focused Crawling via CSS Cues and Reinforcement. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining (KDDD 2018).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

KDDD 2018, August 2018, London, United Kingdom

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The abundance of data on the so-called "deep web" has been estimated to include more than 97% of the Internet as we know it [9]. Hidden in the deep web are a variety of legitimate business products: for example, the deep web includes your personal purchases of products on Amazon, your home banking and mortgage information; and other e-commerce. Also in the deep web are a variety of what can best be described as illicit goods and activities: drugs, weapons, forgeries; even humans as slaves or what has been more commonly known as "human trafficking" (or HT, as we use throughout the paper) [10].

The data on the deep web is hidden behind a series of checkpoints that typically stop crawlers and big companies e.g., Google and Bing from accessing it: for example, the content is behind a login/password, or behind Javascript/AJAX page rendering calls, or finally the content itself is multimedia and not easily analyzed or summarized. We experienced this as we worked on the DARPA MEMEX initiative [10] and built the first full catalog of human trafficking on the deep web: including over 60 million ads (today 80 million) and 40 million images [10].

While developing approaches to perform bulk analysis on this data with DARPA and our law enforcement partners, our team noticed that many of these products are sold on what are commonly called "forum sites". Forum sites are akin to older bulletin board systems in which an Internet user authors a post (in this case an "ad") on then other Internet users browse the ad and interact with it. The user determines if she is interested in the product being sold; may post a comment; or potentially write down or call the phone number associated with the ad (if listed). Users may be required to register with these sites, and may have some identifying "handle", e.g., a username, a location, and other properties.

Given that the deep web is so large; and that the MEMEX program only had access to a limited set of computational resources, developing a *focused* crawling approach - in which we direct the crawlers only at pages that are "relevant" to the topic at hand - was a key interest of many of the teams on the program, including our own. While working on our own focused crawler for the human trafficking domain, developing methodologies to handle forum sites became quite a challenge. This is part due to the organization of forum sites. Forums are typically "hubs" of links [19], that contain little content on the "hub" pages, but then detailed information on the "authority" pages or "content" pages they point to, in this case, the HT ads. This problem is often described as the web crawling "myopia problem" which refers to crawlers discounting hub pages that do not have relevant content.

Complicating the situation was the problem of detecting when a forum hub link actually pointed to an ad, versus, e.g., pointed to navigation for another area of the site, or another product that did not have to do with human trafficking. Manual examination of the forum pages by our team and Subject Matter Experts (SMEs) in the domain revealed that styling and visual properties of links and their surrounding tags were the key "cue" highlighting the difference between an actual ad in the HT domain instead of navigation or other links on the site.

Our team surveyed the available focused crawlers and selected the Apache Nutch system because of our familiarity with the project and because Nutch was representative of most of the other crawlers in the domain. Our prior work focused on adding focused crawling support for both multi-class page labeling (using Naive Bayes); and/or page text feature modeling (using Cosine Similarity). Neither of these focused crawling capabilities or others for that matter e.g., from Scrapy [29]; ACHE [25]; Storm Crawler [17] were sufficient because the focused crawling models they use are trained *offline* and they do not adjust to the board's constantly changing visual stylistic structure for its hub links based on forum posts.

To address this myriad of issues, we developed an approach for focused crawling that we dub "ROACH" for *R*einforcement based *O*nline *A*pprentice *C*ritic focused crawling approac*H*. ROACH makes several contributions that overcome the issues encountered while trying to perform focused crawling over forum sites on the modern Internet, including: (1) delivering an online/adaptive crawler that can learn in a scalable and accurate manner even on forum sites; (2) considering modern CSS visual and stylistic properties; (3) providing a mechanism for the crawler to learn, both statically, seeded with SME domain knowledge; and in an online adaptive fashion using a simplified version of reinforcement learning and back propagation; and (4) making the approach independent of crawler implementation as it can be applied to all of the major crawling systems and does not rely on any particular library.

A roadmap of the paper is as follows. Section 2 will motivate ROACH by considering related work in the domain. Section 3 describes how ROACH crawls including its approach for online learning and training. Section 4 describes ROACH's derived visual features. Section 5 highlights the experiments conducted with ROACH and the results of the work and Section 6 concludes the paper, pointing to areas of future work.

2 RELATED WORK

Some of the earliest works in the area of focused crawling were the *FishSearch* system by De Bra et al. [7] and *SharkSearch* by Hersovici et al. [12]. These systems used simple keyword search or matching based on the bag of words approach. Menczer et al. [23] provide an evaluation framework for topical crawling algorithms. They experimented with the trade-off between exploration vs. exploitation and concluded that a hybrid approach works best. They also improved and generalized the previous topic crawling algorithms. Chakrabarti et al. coined the term focused crawling in their paper [6]. In their work, they engage the user to provide initial examples of good urls, and according to their pre-fitted classification tree, trained using a canonical taxonomy, help user decide the best classes or categories for the urls. They use this information to train a classifier which is used during the crawl process.

Since these earlier efforts, there have been many projects to improve the idea of focused crawling. Lu et al. [22] used a multi-stage process for focusing their crawler, where each outgoing link from a page (or "outlink") undergoes multiple filtering stages based on varying level of contexts: from more general to more specific. They use a classifier based on an improved tfidf weighting system to classify the whole web page. They then further divide the web page into smaller blocks based on its layout, and further classify each block. The outlinks from the non-relevant blocks are subjected to classification based on anchor-text and link-context to be able to finally make it to a priority queue. Hao et al. [11] experimented with Latent Symantec Indexing (LSI) as an alternate to tfidf in vectorizing the text used in classification and similarity finding tasks usually during a focused crawl. They proved that the combination of LSI and tfidf achieves better relevancy results. Pant et al. [24] use web page text and two kinds of link context features: one with a fixed window size and another using the hierarchical tag structure around the link. They use combinations of each link context feature with the web page text feature and a balancing hyper-parameter to combine these features in different weight proportions.

Unlike the above mentioned general purpose focused crawlers, there are some crawlers that have targeted certain part of the Internet. For example, the work by Illou et al. [14] creates a crawler specifically to crawl the dark web, similar to our own efforts on MEMEX and with ROACH. The project uses text features from the parent and target web pages and it uses textual context from around the outlink to predict its relevancy. Illou et al.'s system employs a rule based approach to define the combinations of different classification techniques for different situations, for example when crawling Tor or the surface internet, similar to our own work with ROACH. Unlike our ROACH project though, Illou et al.'s crawling work does not learn or adapt - it simply follows the rules as to which classifier to use: instead ROACH learns through its Apprentice-Critic framework (described in the next section). Jiang et al. [15] created a crawler specifically for collecting data from web forums, similar to the goals of ROACH. Jiang et al.'s crawler learns the patterns in the URL structures across forums since they are structured and these patterns are common across many if not most of the forums. Our own work on ROACH is less focused on URL patterns (though it does consider textual features from around the URL) and it employs visual/stylistic pattern learning.

Several works have entertained the idea of the graph context or delayed reward for focused web crawling. Much of this work stems from the domain of reinforcement learning (RL) literature and informs the ROACH approach to apprentice-critic learning (described in the next paragraph briefly and more so in the next section). Some initial works on this concept came from Diligenti et al.[8] and Rennie et al. [26]. Dilligenti et al. creates and trains 'context graphs' to assess how far a particular page is from a relevant page. They start from a list of seed urls, which are supposed to be examples of good targets. They build a backward link graph for each seed using a search engine. They then use the information about the distance of each backward page from the target pages to train a predictor. Some other notable works have considered the idea of backlinks, although not solely for the purposes of crawling,

```
{
  "tag": "div",
  "text": "",
  "css": "<css_properties>",
  "attr": {
    "class": "abc"
  },
  "children": [
    {
      "tag": "div",
      "text": "best offer",
      "css": "<css_properties>",
      "attr": {
        "href": "www.bestcofeee.com"
      },
      "children": []
    }
  ]
}
```


Figure 1: *Shows a small sample of the recursive json format saved by our custom parser*

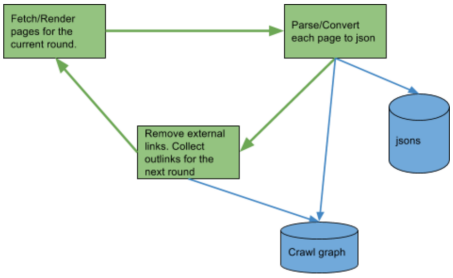

Figure 2: *Crawling (Data collection) Framework*

are [4] and [2]. Rennie et al. [26] map context features from around the outlink to a future reward value (or Q value), rather than the immediate reward for the page. By doing this, it simplifies the problem of using RL for crawling to a classification problem. Other crawling works that leverage RL include L. Jiang et al. [16]. The authors formalize Q-leaning for crawling. Their system searches the web based on queries and assumes the web as a database. The authors of "Yet Another Focused Crawler" [28] use the whole web page instead of just the outlink context to predict the future reward. The FICA approach by Bidoki et al. [3], although not a topically focused crawler, provides a general purpose algorithm based on RL to rank the pages better to avoid crawling the web exhaustively. Although not RL based, the work of Hongyu Liu at al. [21] models focused crawling as a sequential task and uses maximum entropy markov models (MEMM) to infer the "state/distance" of a new link relative to a target/good page.

Chakrabarti et al. [5] model crawling in a online learning framework called the apprentice-critic framework. The approach enables a crawler to adaptively and in an online fashion. The crawler uses two classifiers. One is an online classifier, "apprentice", which also learns with more crawl data. The other classifier, a "critic", is a static relevancy model that provides training instances to the apprentice. The apprentice-critic approach has been used by several other researchers besides our own work, for example in Barbosa at al. [1]. They use the actor critic framework and url context features to make predictions. Barbosa et al.'s work learns to predict the expected distance from a good page and also to balance between exploration and exploitation steps it tries to fetch links with both highest expected reward and delayed reward. Apprentice-critic is the focused crawling technique we employ in ROACH.

3 APPROACH

3.1 Crawling Process

In ROACH, we first fetch the pages in rounds or in a breadth first manner. Each round corresponds to a particular articulation of the priority order in which links (either initial seeds, or those extracted from pages) are visited. For each link, the corresponding page content is fetched and then extracted and parsed as described later in this section. The parsed page is represented as a recursive JSON structure, an example of which is provided in Fig. 1. A JSON is created for every page and saved for later processing. The JSON is a simplified version of the html content of the page that goes through a cleaning process described in the next paragraph. As part of the fetching process, ROACH headlessly renders every page using PhantomJs [13] and then uses the computed values to discern Cascading Style Sheet (CSS) properties - these are the properties that capture stylistic and visual elements of links in the modern Internet and dealing with these properties captures the intuitive cues that Subject Matter Experts (SME) use when browsing the web.

As part of the ROACH cleansing process, we remove the script, style, link and meta tags for the rendered page. All the tag attributes are ignored except `href` and `link` attributes associated with `a` tags and attributes associated with `link` tags. Additionally, ROACH converts all relative links to absolute links. Finally, text is extracted from each tag on the rendered page. Since the extracted text contains an abundance of boilerplate in our experiments, to expel it, ROACH removes any sentence that is less than five words. ROACH then vectorizes the document using 1,2 and 3-grams. We create a vector for each page and save it as our `Critic` model described in the next subsection.

ROACH restricts its focus to only external outlinks – those not belonging to the same host as the page itself. If we considered all links (external and internal) the fetch-list becomes too large too quickly, violating our already stated goal of ensuring that ROACH can be online and adaptive. Only considering external links also aids our process by removing biases and loops in the crawl graph. All the outlinks that we process are given ids and a reference to their parent ids is maintained in order trace their lineage. This data structure is referred to as the ROACH crawl graph. Since each round grows in size quickly, we crawl each round in parts incrementally. If there is an error and the crawler aborts, the next crawl will be checkpointed and will start from the last step of that round, and not from the very beginning of that round. The overall ROACH process is highlighted in Fig. 2.

3.2 Learning Process

ROACH employs the critic-apprentice model first suggested by Chakrabarti et al. [5]. In this model there are two parts. One is a static evaluator, called as the `Critic`. The other part is the online learning system, called the `Apprentice`. The `Critic` represents a subject matter expert (SME) who has a good sense of what she is looking for. The `Critic` is equivalent to a topic or language model. It is usually trained offline and not during the crawl and is therefore a static model. The `Apprentice` can use the knowledge of the `Critic` to learn the best approach for finding more relevant pages. It can be thought of as the necessary clues that the SME learns while browsing that achieve the most relevant pages in the fastest manner.

ROACH implements the `Critic` with a Cosine Similarity based approach. Cosine Similarity is used here instead of classification (e.g., Naive Bayes) to assess the relevancy of pages because it is difficult to create an initial dataset with both relevant and irrelevant examples. This is especially apparent in domains where exploratory analysis is required, such as the DARPA MEMEX domain of human trafficking that motivated this work. In particular, it is difficult to label any one page as "not relevant", especially when exploratory analysis is required. So, by using Cosine Similarity, an SME can simply define pages that are (more) relevant.

The `Apprentice` is implemented in ROACH using regression and tree-based algorithms. The `Apprentice` learns to map a list of features based on extracted page text and its CSS visual properties to the importance of a page. This model then can be used to predict the importance of a link before fetching it. Regression and random forest classification are both used to train the ROACH `Apprentice` model using data from previous rounds and to predict scores for the later rounds. The results of this process are evaluated based on how well the ROACH `Apprentice` ranks the outlinks. The `Critic` model provides a relevancy score to a page once it is fetched, whereas the `Apprentice` model provides a score to an unfetched link.

A correct ranking of the outlinks is very important for any focused crawler as it helps to conserve resources by only considering highly scored links, and by removing poorly scored links each round. Considering that in a round-based crawling model, the number of pages per round increases exponentially, having an appropriate consideration of only the most important outlinks saves the crawler time and memory resources. Along these lines, to evaluate our overall focused crawling approach in ROACH, we used a weighted version [20] of Kendall's Tau statistic [27]. Kendall's Tau statistic is used to measure the similarity between two rankings. In ROACH we are strongly interested in pages in which rankings that are not strongly correlated become strongly correlated. To embody this idea we penalize these situations/inversions based on the absolute distance between the true relevancy of the pages. This makes the ranking similarity evaluation more robust and also removes noise.

4 FEATURES

In this section, we describe ROACH's unique features extracted that provide appropriate cues used in focused crawling to emulate a Subject Matter Expert (SME). First we describe page score (*PS*), recorded for both the page and its parent - this is a *Lineage*-based Feature. Then we describe two *Context*-based features: the first is the extracted CSS properties capturing visual and stylistic elements of the link (*VC*); and second is the text of the page (*TC*).

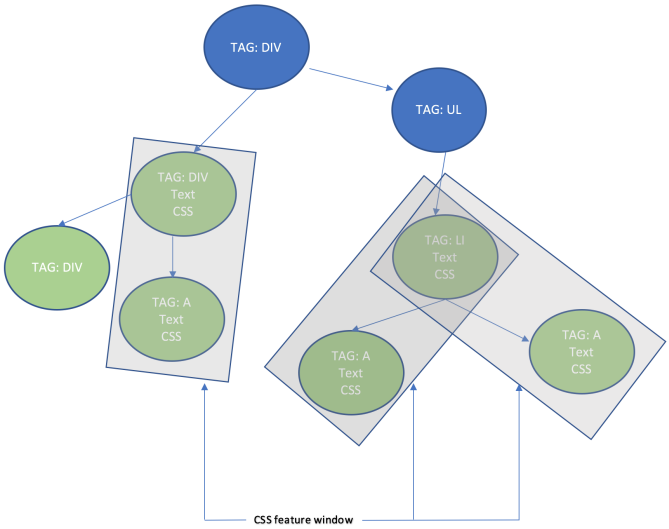

Figure 3: *Shows the context features with respect to the DOM tree of an HTML page.*

4.1 Lineage Based Feature

4.1.1 Page Score. As already mentioned in the prior section as part of the `Critic` as part of ROACH we train a Cosine Similarity based model to give an immediate relevancy score or reward to each page in the context of a domain/topic - for example in our test data for the human trafficking (HT) domain. This relevancy score is both a feature and also is used for ground truth. The score is ranged between 0 and 1 and is given to a parsed page. Every page has two relevancy scores associated with it. One is that of its parent page, which is used as a feature, and the second is the relevancy score of itself (when fetched). The `Apprentice` tries to learn the true relevancy score, beyond the initial value used for ground truth.

We start with a list of relevant pages curated by subject matter experts (SMEs) in the domain - for our work - in the HT domain. We then parse each of pages in the list and extract text from as detailed in process outlined in Section 3.1. To score a new page we use the same process and create a vector for the new page and find Cosine Similarity with each of the page vectors in our `Critic` model. ROACH then computes an arithmetic mean of all the cosine scores. The mean is then used as the relevancy score for the page.

To further improve our model, we remove the pages that demonstrate lesser relevance compared to other pages in the model. To do so we allow the user to provide a relevancy threshold which can be used to tune the algorithm's scalability and to ensure that it can be run in an online mode.

4.2 Context Based Features

We also consider features from a particular outlink's surrounding *context*. ROACH identifes features "closest" to the outlinks through inspection of the Document Object Model (DOM) structure of the

Categorical-valued	Numerical-valued	Multi-valued
float	height	font-family
display	width	
clear	font-size	
text-align	text-indent	
text-decoration	letter-spacing	
font-style	word-spacing	
border(bottom,left,right,top)-style	line-height	
visibility	border-(bottom,left,right,top)-width	
vertical-align	left	
	top	
	right	
	bottom	
	z-index	
	background-color	
	color	
	border-(bottom,left,right,top)-color	
	margin-(bottom,left,right,top)	
	padding-(bottom,left,right,top)	

Figure 4: *CSS features used: divided categorically according to their type of value*

page. As an example, Fig. 3 represents a small part of a page's DOM tree. The green nodes represent the leaf nodes and the node just before the leaf node in the tree. For each outlink we select its immediate tag and its parent's tag. The boxes represent the contexts for the tree outlinks (represented by the a tags). We use two types of context features. The CSS properties and text in the context of the outlink, both of which we will describe below.

4.2.1 CSS features. CSS features represent the modern codification of visual and stylistic cues and are often overlooked in other approaches besides ROACH. ROACH extracts CSS and then cleans, normalizes, and vectorizes the CSS (referred to as *VC*) as we will describe in this section. We used 28 types of CSS attributes as shown in Fig. 4. We divide the features into three parts based on the type of value of the attribute: (1) *Numerical-valued features* – These features are the CSS properties that have a numerical value. We directly use the numerical value to create our vector. Some values have more than one part e.g., `background-color=rgba(255,0,0,0)`. We divide such properties into e.g., `background-color1:0`, `background-color2:0`, `background-color3:0` and `background-color4:0`; (2) *Categorical-valued features* – these CSS attributes have a categorical value, and not a number. For example, the CSS property `float` can have values e.g., `top`, `right`, `left` and so on. We use one-hot encoding to vectorize such properties/features; and (3) *Multivalued features* – The only CSS multi-valued property we are currently evaluating is `font-family`. Each html tag may contain one or more fonts associated with it. We use a multi-label binarizer to encode this feature. Finally, we concatenate all the features together to one large feature array for each data sample. In total we have 61 features after flattening the color based features as shown.

ROACH also considers the CSS properties of the tag closest to the outlink (a tag) but also the parent tag of it. So for each outlink we have 61+61=122 CSS features.

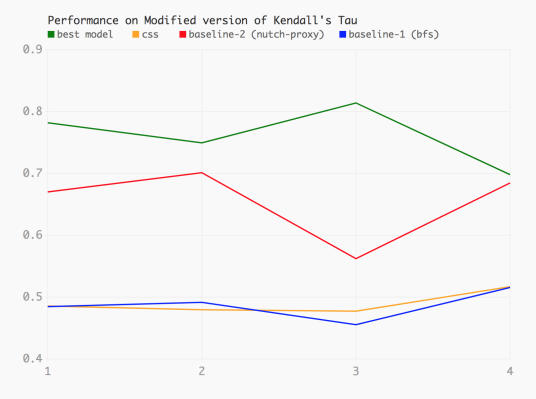

Figure 5: *x-axis: The round used for evaluation*
y-axis: The Modified Kendall's Tau score

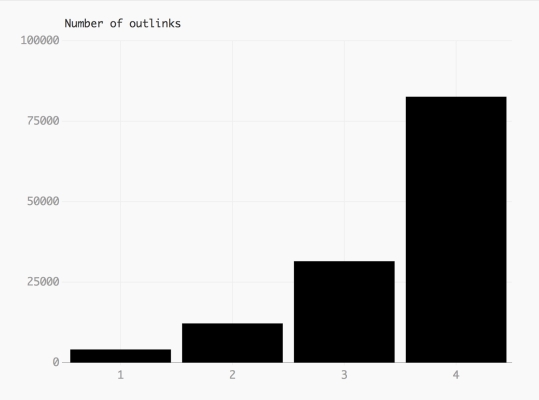

Figure 6: *x-axis: Round*
y-axis: Number of outlinks found

4.2.2 Textual features. ROACH extracts text features from the context of each outlink - the text is also vectorized, similarly to the CSS feature context. The vectorized text is referred to as "Text Content" or *TC*. Text is extracted from the leaf and parent a tag, similar to the CSS feature step. During experimentation, we noticed that the text was very noisy. In the domain of human trafficking, the text contained phone numbers, names of people and had many biases including location names. We performed named entity recognition (NER) on the text and removed numbers, names, dates and locations and common English stop words. We further removed names/usernames by removing any term that had a number or punctuation in it.

For each round we prepare the above three features for each outlink. Parent score corresponds to the score for the parent page of the outlink (also referred to as PS_parent). We also score the page pointed by the outlink (PS_outlink) and use it as the ground truth. We remind the reader that PS_outlink becomes the PS_parent for the next round of data.

	Ridge	RandomForestRegressor
tfidf, css, parentscore	0.678523	0.621515
tfidf, parentscore	0.704451	0.639136
css, parentscore	0.617517	0.579628
css, tfidf	0.600537	0.58493
tfidf	0.781953	0.785731
css	0.485619	0.496471
parentscore (baseline-2/nutch-proxy)	0.670092	0.618994
BFS (baseline-1)	0.484697	0.484697

Table 1: Train round-0 , Test round-1

	Ridge	RandomForestRegressor
tfidf, css, parentscore	0.692217	0.639252
tfidf, parentscore	0.701446	0.615459
css, parentscore	0.684329	0.611126
css, tfidf	0.599863	0.602131
tfidf	0.749443	0.75054
css	0.479662	0.534636
parentscore (baseline-2/nutch-proxy)	0.701232	0.605897
BFS (baseline-1)	0.49161	0.49161

Table 2: Train round-1 , Test round-2

	Ridge	RandomForestRegressor
tfidf, css, parentscore	0.660804	0.629791
tfidf, parentscore	0.675493	0.602931
css, parentscore	0.546019	0.522309
css, tfidf	0.705901	0.708658
tfidf	0.813964	0.811132
css	0.477267	0.485781
parentscore (baseline-2/nutch-proxy)	0.562239	0.618519
BFS (baseline-1)	0.455447	0.455447

Table 3: Train round-2 , Test round-3

	Ridge	RandomForestRegressor
tfidf, css, parentscore	0.695267	0.623203
tfidf, parentscore	0.698008	0.644386
css, parentscore	0.678142	0.613548
css, tfidf	0.588431	0.578653
tfidf	0.677893	0.668048
css	0.516906	0.507159
parentscore (baseline-2/nutch-proxy)	0.684677	0.611027
BFS (baseline-1)	0.51565	0.51565

Table 4: Train round-3 , Test round-4

5 EXPERIMENTS AND RESULTS

To evaluate ROACH, we experimented using Regression and Tree based algorithms to understand the best ways to model our features. We also tried different ways to combine our various feature categories. Below we describe our two baselines, Baseline 1 or **B1**, and Baseline 2 (**B2**), respectively.

- B1 We use Breadth First Search or BFS as our first baseline. This baseline gives ranks to the outlinks according to the the sequence in which they were extracted by a system that does not care about rearranging the outlink queue according to some topical preference.
- B2 We use the idea behind Apache Nutch [18], which is one of the most popular open source web crawlers, as the baseline for our experiments. Our own prior work added a Naive Bayes classifier to Nutch for focused crawling. In each round the classifier discards outlinks that are not relevant based on a binary classification, which encodes SME knowledge of a page as 'relevant' or 'irrelevant' and which uses it to evaluate the outlinks of the parent page. We don't use Nutch as is, but have emulated its behavior by using just the pagescore (PS) feature to predict the relevancy/score of an outlink. We use this as our baseline, which we dub `Nutch-Proxy`.

There are some other key technical differenced between Nutch-Proxy and Nutch. The former provides predictions (between 0 and 1) of pages based on an average tfidf score, which helps us rank the pages and compare it against our results, whereas the latter discards the pages altogether if it is found irrelevant by a binary Naive Bayes classifier. This makes it difficult to rank the pages or even control the amount of outlinks for the next round. These are some of the reasons we are using a Nutch-Proxy which is technically different but embodies the same idea of judging the outlinks based on just the score for the parent page.

As many sites in the HT domain may present a captcha, or an explicit opt-in overlay button to log in to the page, these sites skew our overall raw *PS* scores resulting in a score of 0 for the pages where we were not able to fetch the link content. To deal with this we discard the data points that have a 0 *PS* score in our experiments when we are doing predictions for any outlinks.

For our experiments we prepared four datasets named Dataset-[1,2,3,4] with ground truth. We also have a Dataset-[0] but will only use it for our first experiment because it is rather small in size, only 1137 pages. To further understand it, consider that Dataset-[n] contains outlinks from the round-[n-1] of our crawl which is also the links crawled in round-[n]. It also contains the features for those outlinks i.e. context features from around the outlink and the relevancy score for its parent page. The ground truth for this round is the relevancy score for the fetched and parsed version of the outlinks themselves. One dataset can be used to train or test any model for predicting the relevancy of the page based just on the features of the outlink. For the rest of this paper we will use the word round in place of dataset as explained.

5.1 Predicting outlink rankings

In this experiment, we use the feature group (PS, TC, VC) (also shown as $(Parentscore, tfidfandCSS)$ in the figures) and ground truth from round n to train our ROACH ranking. The ranking is

$$p_{n+1} = P(feat_{n+1}, M(feat_n, gt_n)) \quad (1)$$

$$s_{n+1} = S(p_{n+1}, gt_{n+1}) \quad (2)$$

Where P is the prediction function, M is the modeling function, S is the Modified Kendall's Tau scoring function, gt is the ground truth, p is the prediction and s is the Modified Kendall's Tau score.

a Regression based model as shown by the mathematical formulation in Eq. 1, which also shows that then we use the features for round n+1 to predict the relevancy scores using the trained model. Eq. 2 shows the evaluation of the predictions based on the ground truth, using the Modified Kendall's Tau Score. We used two different regression techniques, Ridge Regression (linear regression) and Random Forest Regression. We evaluated multiple feature combinations. To combine different features we concatenated them. We then trained on this concatenated set of features.

Experiments were conducted over four sets of rounds Train[0] Test[1], Train[1] Test[2], Train[2] Test[3], Train[3] Test[4]. Table 1, 2, 3 and 4 shows the results or each feature combination and Regression Model pair in a heatmap (that is, larger values, or more relevant links are selected when the color green is darker; otherwise lighter means less relevant). The results show that the in the first three rounds tfidf is a very dominating feature. In the last round however, which has the most number of data samples (82,609 pages), we note that the combination of tfidf and parentscore produced the best results, as highlighted in in the first row of Table 3. Although, the combination of all three features followed closely, it did not perform the best which suggests that better modeling approaches are needed to model all the three features together successfully. The CSS feature proved to be a better indicator of good pages compared to our non-focused baseline (BFS) in the last two rounds. Table 3 and Table 4 highlight the focusing capabilities of the CSS properties or visual indicators on a page. We also noted that the linear regressors produced better and more consistent results than the Random Forest regressors. Fig. 5 shows the plot of the Modified Kendall's Tau for the predictions on different rounds. We compare the results between the best performing models in each round, only CSS based model and the two baselines. It shows that our best models consistently outperformed both baselines. Fig. 6 shows the number of outlinks we found for each round.

5.2 Stacking

We hypothesized that concatenating the three features may not be the best technique because it increases the dimensionality of the data, which makes it harder for any model to learn and may also lead to the model becoming more biased towards certain feature groups. This is not ideal as the three different features are inherently different. All the three different features represent different cues from a page. Moreover, the Parentscore (*PS*) and tfidf (*TC*) have floating point values whereas the CSS feature (*VC*) has integer values. To address this concern with the simple model, we tried to combine the models using stacking. We used the first dataset to train multiple models and tested them on the second dataset. Eq. 3, 4, 5 and 6 show this process. a and b represent two different models.

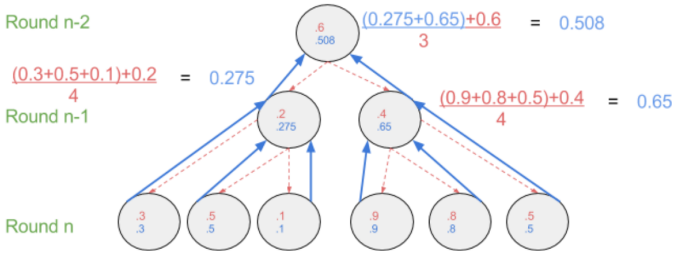

Figure 7: *Shows a representation of a small portion of the crawl graph and how scores are backpropagated*

For example, model-a could be based on CSS+tfidf and Model-b on Parentscore, as also shown in Fig. 8. Then based on the performance of the different models measured by the Mod. Kendall's Tau, we attributed them respective importances (In general this could be thought as using regression on the models themselves). We further normalized the importances to find the relative importances, as shown in Eq. 7, 8, 9, 10. We then used the second dataset to train multiple new models and predict the scores for the last dataset as in Eq. 11, 12. We then modified the predictions on the third dataset taking a weighted (based on the importance of the model) average of all the models, as in Eq. 13.

$$p_{n+1,a} = P(feat_{n+1}, M_a(feat_n, gt_n)) \quad (3)$$

$$p_{n+1,b} = P(feat_{n+1}, M_b(feat_n, gt_n)) \quad (4)$$

$$s_{n+1,a} = S(p_{n+1,a}, gt_{n+1}) \quad (5)$$

$$s_{n+1,b} = S(p_{n+1,b}, gt_{n+1}) \quad (6)$$

$$i_a = s_{n+1,a} \quad (7)$$

$$i_a = s_{n+1,b} \quad (8)$$

$$ri_a = s_{n+1,a}/(s_{n+1,a} + s_{n+1,b}) \quad (9)$$

$$ri_b = s_{n+1,b}/(s_{n+1,a} + s_{n+1,b}) \quad (10)$$

$$p_{n+2,a} = P(feat_{n+2}, M_a(feat_{n+1}, gt_{n+1})) \quad (11)$$

$$p_{n+2,b} = P(feat_{n+2}, M_b(feat_{n+1}, gt_{n+1})) \quad (12)$$

$$p_{n+2} = (p_{n+2,a} * ri_a) + (p_{n+2,b} * ri_b) \quad (13)$$

$$s_{n+2} = S(p_{n+2}, gt_{n+2}) \quad (14)$$

For this experiment we used a set of two and three models as shown in Table 5 and 6. It did perform well in combining the three features in the first set of data we used, row-1 Table 5, and provided a marked improvement over the score for the three feature combination in row 1 of Table 3 (0.660804 compared to 0.752245). Although, the second set of data we used did not provide any gain over the normal modeling, the results still remained close to that of the normal modeling approaches. Overall, the stacking approach produced resilient baseline scores against the remainder of the experiments. In an online situation when we have to make a decision about which ranking to choose for the next round, this can serve as a safe option. It shows that it is useful to use the Mod. Kendall's Tau as a feedback to learn model importances.

	Ridge	RandomForestRegressor
(tfidf, parentscore) (css)	0.652982	0.576178
(css, parentscore) (tfidf)	0.752245	0.746041
(css, tfidf) (parentscore)	0.732263	0.715133
(tfidf) (css)	0.724991	0.733631
(parentscore) (css)	0.543385	0.573244
(tfidf) (parentscore)	0.75834	0.757827
(tfidf) (parentscore) (css)	0.748109	0.744589

Table 5: Learning from : (Train round-1 , Test round-2) ; Manipulating : (Train round-2 , Test round-3)

	Ridge	RandomForestRegressor
(tfidf, parentscore) (css)	0.691044	0.634844
(css, parentscore) (tfidf)	0.687132	0.645605
(css, tfidf) (parentscore)	0.692451	0.638878
(tfidf) (css)	0.587084	0.572806
(parentscore) (css)	0.677425	0.60219
(tfidf) (parentscore)	0.691042	0.643882
(tfidf) (parentscore) (css)	0.687148	0.637933

Table 6: Learning from : (Train round-2 , Test round-3) ; Manipulating : (Train round-3 , Test round-4)

5.3 Reinforcement

As we stated earlier, a known shortcoming of focused crawlers is the "myopia problem". We try to address this issue by correcting the relevancy scores / ground truth for the outlinks. We reinforce the scores by backpropagating the scores using the crawl graph. We start from the most recent round and move backwards. The score for each link in a round is replaced by the average score of its child links in the next round.

The Eq. 15, 16, 17 represent the three different kinds of data enriching approaches we are contrasting in this section. Eq. 15 represents our usual approach with data from a single dataset/round. Eq. 16 represents the approach where we use data from multiple rounds in a model. Eq. 17 is equivalent to Eq. 16 except we recursively reinforce the data of the previous rounds with backpropagation as explained by the Figure 7. It shows a small crawl graph with three levels/rounds. We show how the scores are backpropagated in this graph.

The nodes represent a page. The dashed red lines represent the direction of crawl, whereas the blue arrows represent the direction of backpropagation in the graph. The numbers in red are the relevancy scores for a page. The scores in blue are the reinforced scores for the pages.

$$m_n = M(feat_{n-1}, gt_{n-1}) \tag{15}$$

$$m_{n,data} = M(C(feat_{n-1}, .., feat_{n-k}), C(gt_{n-1}, .., gt_{n-k})) \tag{16}$$

$$m_{n,reinforced,data} = M(C(feat_{n-1}, .., feat_{n-k}), C(gt_{n-1}, .., gt_{n-k-1,..,n-1}, gt_{n-k,..,n-1})) \tag{17}$$

$$p_n = P(feat_n, m_n) \tag{18}$$

$$p_{n,data} = P(feat_n, m_{n,data}) \tag{19}$$

$$p_{n,reinforced,data} = P(feat_n, m_{n,reinforced,data}) \tag{20}$$

Where C is the concatenation function, k is the depth of reinforcement.

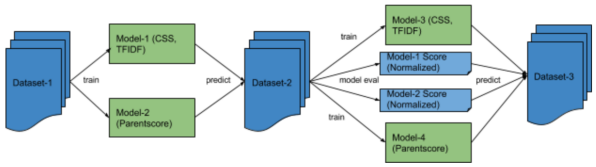

Figure 8: *Shows a representation of the process of stacking for two models.*

	Ridge	RandomForestRegressor
tfidf, css, parentscore	0.695267	0.623203
tfidf, parentscore	0.698008	0.644386
css, parentscore	0.678142	0.613548

Table 7: Train round-3 , Test round-4

	Ridge	RandomForestRegressor
tfidf, css, parentscore	0.695256	0.623165
tfidf, parentscore	0.698019	0.644316
css, parentscore	0.678149	0.613507

Table 8: Train round-(1,2,3) , Test round-4

	Ridge	RandomForestRegressor
tfidf, css, parentscore	0.69527	0.623132
tfidf, parentscore	0.698032	0.644271
css, parentscore	0.678153	0.613472

Table 9: Train round-(1,2,3) with reinforcement , Test round-4

To understand the usefulness of reinforcement, we "combined" the data from the rounds 1, 2 and 3 in with and without reinforcement. Whereas in both the cases we took all the data from these three rounds and used it to predict round-[4], in the latter we also reinforced the scores of round-[2] with round-[3] and then round-[1] with round-[2]. We also contrasted the results with the usual approach we used on all the previous experiments in this paper, training with round-[3] for a subset of our models. Table 7 (for reference to our usual approach), Table 8 and Table 9 present the results of this experiment. The results, shown in the second rows of Tables 7 and Table 8 respectively demonstrate that combining all the previous data to predict link accuracy improves the result. Also, the reinforced dataset improves the best score even a bit further as indicated in the second rows of Tables 8 and Table 9, respectively.

6 CONCLUSION

We presented ROACH, an online adaptive focused crawling approach that combines static subject matter expert (SME) knowledge with online learning and back propagation of scores based on CSS visual cues and other page and link features. Our approach improved on the baseline link accuracy when compared with Apache Nutch, and also a traditional BFS crawling approach, and found more relevant content, faster. It provided a case for using the Modified Kendall's Tau for evaluation of crawl results. ROACH provided some valuable insights into the usefulness of the previously neglected CSS features for focused crawling and a case for continuing its exploration. ROACH contributed a methodology for capturing SME learning using a simplified form of reinforcement learning with back propagation. The results of our work demonstrate that ROACH is a viable focused crawling approach and we intend to evaluate it in other web domains both on the surface and deep web. In future, we would like to explore better ways of modeling CSS features and using them to improve the more dominant textual features. We would also like to do more testing over other topic domains and a larger dataset to see how the results contrast. We are also interested in trying more Ensemble and Meta-learning approaches. We would also like to implement this crawler into an open source product to be able to compare it more fairly with other crawlers and techniques in a more online setting.

ACKNOWLEDGEMENTS

This effort was supported in part by JPL, managed by the California Institute of Technology on behalf of NASA, and additionally in part by the DARPA Memex/XDATA/D3M programs and NSF award numbers ICER-1639753, PLR-1348450 and PLR-144562 funded a portion of the work.

REFERENCES

[1] Luciano Barbosa and Juliana Freire. 2007. An adaptive crawler for locating hidden-web entry points. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 441–450.

[2] Krishna Bharat, Andrei Broder, Monika Henzinger, Puneet Kumar, and Suresh Venkatasubramanian. 1998. The connectivity server: Fast access to linkage information on the web. *Computer networks and ISDN Systems* 30, 1-7 (1998), 469–477.

[3] Ali Mohammad Zareh Bidoki, Nasser Yazdani, and Pedram Ghodsnia. 2009. FICA: A novel intelligent crawling algorithm based on reinforcement learning. *Web Intelligence and Agent Systems: An International Journal* 7, 4 (2009), 363–373.

[4] Soumen Chakrabarti, David A Gibson, and Kevin S McCurley. 1999. Surfing the Web backwards. *Computer Networks* 31, 11-16 (1999), 1679–1693.

[5] Soumen Chakrabarti, Kunal Punera, and Mallela Subramanyam. 2002. Accelerated focused crawling through online relevance feedback. In *Proceedings of the 11th international conference on World Wide Web*. ACM, 148–159.

[6] Soumen Chakrabarti, Martin Van den Berg, and Byron Dom. 1999. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer networks* 31, 11-16 (1999), 1623–1640.

[7] PME De Bra and RDJ Post. 1994. Searching for arbitrary information in the www: The fish-search for mosaic. In *WWW Conference*.

[8] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C Lee Giles, Marco Gori, et al. 2000. Focused Crawling Using Context Graphs.. In *VLDB*. 527–534.

[9] M Goodman. 2015. Most of the web is invisible to Google. Kere?s what it contains. *Popular Science, Retrieved November* 18 (2015), 2016.

[10] Larry Greenemeier. 2015. Human traffickers caught on hidden internet. *Scientific American* 8 (2015).

[11] Hong-Wei Hao, Cui-Xia Mu, Xu-Cheng Yin, Shen Li, and Zhi-Bin Wang. 2011. An improved topic relevance algorithm for focused crawling. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*. IEEE, 850–855.

[12] Michael Hersovici, Michal Jacovi, Yoelle S Maarek, Dan Pelleg, Menachem Shtalhaim, and Sigalit Ur. 1998. The shark-search algorithm. An application: tailored Web site mapping. *Computer Networks and ISDN Systems* 30, 1-7 (1998), 317–326.

[13] Ariya Hidayat et al. 2013. Phantomjs. *Computer software. PhantomJS. Vers* 1, 7 (2013).

[14] Christos Iliou, George Kalpakis, Theodora Tsikrika, Stefanos Vrochidis, and Ioannis Kompatsiaris. 2017. Hybrid focused crawling on the Surface and the Dark Web. *EURASIP Journal on Information Security* 2017, 1 (2017), 11.

[15] Jingtian Jiang, Xinying Song, Nenghai Yu, and Chin-Yew Lin. 2013. Focus: learning to crawl web forums. *IEEE Transactions on Knowledge and Data Engineering* 25, 6 (2013), 1293–1306.

[16] Lu Jiang, Zhaohui Wu, Qian Feng, Jun Liu, and Qinghua Zheng. 2010. Efficient deep web crawling using reinforcement learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 428–439.

[17] Shankar Karuppayah, Mathias Fischer, Christian Rossow, and Max Mühlhäuser. 2014. On advanced monitoring in resilient and unstructured P2P botnets. In *Communications (ICC), 2014 IEEE International Conference on*. IEEE, 871–877.

[18] Rohit Khare, Doug Cutting, Kragen Sitaker, and Adam Rifkin. 2004. Nutch: A flexible and scalable open-source web search engine. *Oregon State University* 1 (2004), 32–32.

[19] Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)* 46, 5 (1999), 604–632.

[20] Ravi Kumar and Sergei Vassilvitskii. 2010. Generalized distances between rankings. In *Proceedings of the 19th international conference on World wide web*. ACM, 571–580.

[21] Hongyu Liu, Evangelos Milios, and Larry Korba. 2008. Exploiting Multiple Features with MEMMs for focused web crawling. In *International Conference on Application of Natural Language to Information Systems*. Springer, 99–110.

[22] Houqing Lu, Donghui Zhan, Lei Zhou, and Dengchao He. 2016. An improved focused crawler: using web page classification and link priority evaluation. *Mathematical Problems in Engineering* 2016 (2016).

[23] Filippo Menczer, Gautam Pant, Padmini Srinivasan, and Miguel E Ruiz. 2001. Evaluating topic-driven web crawlers. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 241–249.

[24] Gautam Pant and Padmini Srinivasan. 2006. Link contexts in classifier-guided topical crawlers. *IEEE Transactions on knowledge and data engineering* 18, 1 (2006), 107–122.

[25] Kien Pham, Aécio Santos, and Juliana Freire. 2016. Understanding website behavior based on user agent. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 1053–1056.

[26] Jason Rennie, Andrew McCallum, et al. 1999. Using reinforcement learning to spider the web efficiently. In *ICML*, Vol. 99. 335–343.

[27] Charles Romesburg. 2004. *Cluster analysis for researchers*. Lulu. com.

[28] George Sakkis. 2003. YAFC: Yet Another Focused Crawler. *The First instructional Conference on Machine Learning (iCML 2003)* (2003).

[29] Jing Wang and Yuchun Guo. 2012. Scrapy-based crawling and user-behavior characteristics analysis on taobao. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2012 International Conference on*. IEEE, 44–52.