Learning to Discover Domain-Specific Web Content

Kien Pham
New York University
kien.pham@nyu.edu

Aécio Santos
New York University
aecio.santos@nyu.edu

Juliana Freire
New York University
juliana.freire@nyu.edu

ABSTRACT

The ability to discover all content relevant to an information domain has many applications, from helping in the understanding of humanitarian crises to countering human and arms trafficking. In such applications, time is of essence: it is crucial to both maximize coverage and identify new content as soon as it becomes available, so that appropriate actions can be taken. In this paper, we propose new methods for efficient domain-specific re-crawling that maximize the yield for new content. By learning patterns of pages that have a high yield, our methods select a small set of pages that can be re-crawled frequently, increasing the coverage and freshness while conserving resources. Unlike previous approaches to this problem, our methods combine different factors to optimize the re-crawling strategy, do not require full snapshots for the learning step, and dynamically adapt the strategy as the crawl progresses. In an empirical evaluation, we have simulated the framework over 600 partial crawl snapshots in three different domains. The results show that our approach can achieve 150% higher coverage compared to existing, state-of-the-art techniques. In addition, it is also able to capture 80% of new relevant content within less than 4 hours of publication.

ACM Reference Format:
Kien Pham, Aécio Santos, and Juliana Freire. 2018. Learning to Discover Domain-Specific Web Content. In *WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining, February 5–9, 2018, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3159652.3159724

1 INTRODUCTION

The wide availability of data on the Web has enabled many new applications in different types of organizations, from government agencies and NGOs, to research universities and tech start-ups. Consider for example, the problem of human trafficking. It is estimated that 63% of sex trafficking victims are advertised online [6]. Analysts at NGOs and law enforcement agencies continuously monitor escort ads posted online to detect traffickers and more quickly rescue the victims. At the Bureau of Alcohol, Tobacco, Firearms and Explosives (ATF), agents surveil online markets where guns and drugs are sold to generate investigation leads as well as forensic evidence to prosecute criminals. Similarly, organizations that fight against illegal trading of wildlife track online marketplaces such as ebay.com to identify illegal products and obtain evidence to prosecute traffickers [20]. As a point of reference, in 2012 one operation by the US Fish and Wildlife Service (FWS) resulted in 154 busts of illegal wildlife goods on the Web [9]. Humanitarian aid organizations need to assess the nature and magnitude of major humanitarian crises around the world in order to prioritize the responses to them. In the early stages of an emergency, they crucially rely on analysis of secondary data (i.e., primary data that is processed by local and international public institutions, non-governmental organizations and news media) collected from the Web [5].

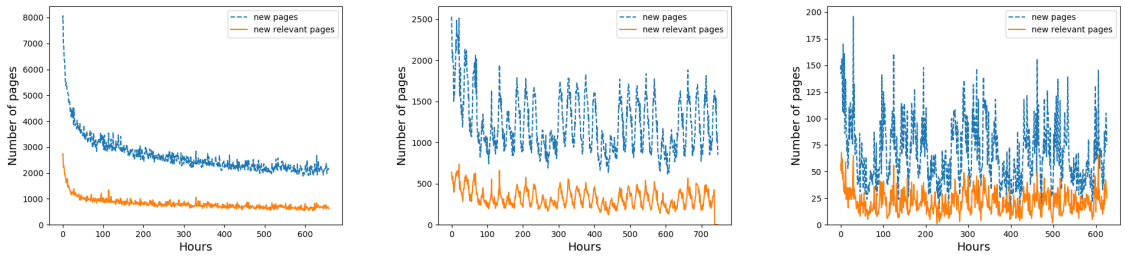
In these applications, the goal is to find *all* pages relevant to a given domain continuously and in a *timely* fashion. But doing so is difficult. Search engines, such as Google and Bing, are the main tools for users looking for information on the Web. They make use of massive computing power to both crawl the Web and create the indexes, which currently cover hundreds of billions of documents. These systems, however, have limitations when faced with domain-specific information needs. Because they aim to maximize coverage and breadth, queries often return a very large number of results, including many that are of little relevance. For example, a law enforcement agent searching for "escort ads" will find links to escort sites but also many links to news sites, which do not contain ads. While it is possible to issue more specific queries such as "site:newyork.backpage.com/WomenSeekMen", which retrieve pages in a dating section of Backpage, there is no guarantee that all pages will be retrieved. Due to resource limitations, search engines often do not download all the pages in each site they encounter. Pruning techniques are used and pages that are important for a domain may be overlooked. In addition, to cover a given domain, a potentially large number of queries need to be issued. In the case of Backpage, to retrieve all escort ads, at least one query per city is needed; and there are many more sites that contain ads in addition to Backpage. Because of the rate limits search engines impose, users may not be able to retrieve even the ads that are in their indexes.

The ideal would be to construct an index that provides comprehensive coverage of a given domain and that is kept up to date: as new pages become available, they are automatically added to the index. Focused crawlers can be used to discover pages that are relevant to a domain [7]. These systems collect pages that satisfy a set of properties and carefully prioritize the crawl frontier to maximize the number of relevant pages discovered while minimizing the number of off-topic pages visited. But continuously crawling to discover new content is difficult and expensive. Consider again the human trafficking domain. Thorn[1] estimates that over 100,000 new escort ads are posted online everyday in a plethora of Web sites – Backpage alone has millions of ads. For users and organizations in the long tail, with limited computational resources, constantly and exhaustively re-crawling all known Web sites is not practical.

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA

ACM ISBN 978-1-4503-5581-0/18/02…$15.00
https://doi.org/10.1145/3159652.3159724

[1] Thorn is an international anti-human trafficking organization that works to address the sexual exploitation of children.



(a) Human Trafficking domain (b) Politics domain (c) Humanitarian Crisis domain

Figure 1: Number of new pages published hourly

Recognizing this problem, approaches have been proposed to make this task more efficient by reducing the number of re-crawled pages while still discovering a large percentage of new pages. Dasgupta et al. [13] estimate that 90% of all new content can be discovered by re-crawling a carefully chosen set of existing pages. They modeled the Web as graph and computed the vertex cover in order to identify a set of pages that can lead to the discovery of links to new pages over time. The selected pages are then periodically re-crawled. Unfortunately, applying this approach to discover new content in a *timely manner* creates new challenges. Re-crawling these selected pages frequently (e.g., hourly or daily) is not practical due to bandwidth limits, standard politeness constraints crawlers must adhere to, and the cost of computing resources. Also, lesser-known web crawlers (i.e., which are not associated to major search engines) are sometimes blocked when detected by sites [23], which do not have incentives to spend resources serving pages to crawlers that do not generate user traffic. Another challenge comes from the fact that pages from different domains can have very different publishing rates [25] and even within a fixed domain, these rates may change over time. This is illustrated in Figure 1 which shows how the number of new links discovered varies depending on different factors such as hour of the day, day of the week, and domain.

Timely Discovery of Domain-Specific Content. In this paper we focus on the problem of *timely discovery of new content in a domain-specific setting*. More formally, we define the content discovery problem as follows: given a set of seed pages S and we want to select the top-k pages S_k^t for every timestamp t, where $S_k^t \subset S$ and $|S_k^t| \ll |S|$, such that re-crawling every S_k^t at a timestamp t maximizes the number of new (relevant) links discovered. We assume that new content can be discovered by re-crawling previously crawled pages, but instead of crawling them periodically using a fixed schedule, we propose new algorithms that learn and leverage page change patterns to dynamically *derive efficient re-crawl schedules that optimize the new content discovery rate over time*. Unlike previous approaches which assumed the crawler has full knowledge of how pages change over time [13, 15], we dynamically learn them as the crawl proceeds and more knowlege about pages is acquired.

We propose a two-stage framework to generate dynamic re-crawl schedules. In the first phase, we predict the number of new outlinks $\hat{o}$ that the page will yield at time $t+1$ and use the prediction to select a set of candidate pages. To do so, we identify a set of useful features that are good predictors for pages that lead to a high yield (i.e., are likely to contain links to new pages) and use machine-learning based algorithms that combine these features to estimate $\hat{o}$. Because different pages may share links, selecting pages that have high yield but whose link sets overlap would negatively impact the overall performance. Thus, during the second phase, we rank the candidate pages taking into account not only the estimated number of new links, but also the estimated overlap among the sets of outlinks in the associated pages. Given a pair of pages, to efficiently estimate the overlap between their links, we propose a new algorithm that takes into account a realistic crawler scenario where only incomplete information is available, i.e., information from the top-k pages crawled in previous re-crawling cycles.

While the greedy, learning-based approach is effective, it may suffer from bias: by selecting only pages that are expected to have high-yield, it may miss new pages could lead to higher yields. A common approach to this problem is to introduce exploration: instead of just exploiting known seeds, we can explore to find (and learn the patterns from) new seeds. The challenge is how balance exploration and exploitation. We propose a method that uses the multi-armed bandits strategy [2] to automatically select an exploration threshold and to dynamically adapt the threshold as the crawl progresses.

To assess the effectiveness and efficiency of the proposed approach, we performed a detailed experimental evaluation to compare it against state-of-the-art discovery techniques using real data from different domains. The results show that the features we selected and the strategy used to combine them lead to effective predictors of page yield. By learning these features and taking the overlap into account, our algorithm outperforms all other approaches: it achieves higher coverage using less resources. In addition, by balancing exploration and exploitation, higher coverage is attained.

Contributions. Our main contributions are as follows:

- We approach content discovery as a ranking problem, where the crawling policy needs to select the top-k best seed pages to be re-crawled at every timestamp t to maximize the total number of new pages discovered. To the best of our knowledge, ours is the first attempt to consider this problem in a domain-specific setting.
- We propose a method to generate dynamic re-crawling schedules that predicts the estimated number of new outlinks $\hat{o}$ the page will yield at time $t+1$ and that also takes the estimated overlap of outlinks into account. We also show that by using

the multi-armed bandits algorithm to automatically balance the tradeoff between exploitation and exploration, our method attains improved coverage.
- We perform extensive experiments to evaluate the proposed approach and compare it against the state-of-the-art methods. The results show improvements up to 150% higher coverage compared to state-of-the-art techniques.

2 RELATED WORK

There is a significant body of work related to re-crawling web pages. Problems addressed include the characterization of the temporal dynamics and evolution of the Web [1, 3, 13, 19], maintenance of overall freshness of crawled content [11, 21, 25–27], and discovery of new links (content) [13–16]. We focus on the latter, and in particular, on *timely* content discovery in specific domains, a problem that, to the best of our knowledge, has not been tackled before. Domain-specific content discovery has been considered in the context of focused crawlers [7], where the goal is to find pages whose content is relevant to a given domain.

Temporal Dynamics and Evolution. The temporal dynamics and evolution of Web content has been studied in [1, 3, 13, 19]. Ntoulas et al. [19] observed that web pages are created and retired at a fast pace. Olston and Pandey [21] studied the longevity of information found in web pages and proposed re-crawl scheduling policies that allow crawlers to target persistent content, instead of ephemeral content that will be quickly overwritten by subsequent changes. Adar et al. [1] studied how Web content changes both with respect to time intervals (hourly and sub-hourly crawls) and the page structure (content, DOM tree and terms). Bar-Yossef [3] studied the decay of the Web and showed that not only do some web pages exhibit a rapid death but also large sub-graphs of the Web decay significantly. For applications, such as the ones discussed in Section 1, that aim to achieve large coverage of a domain over time, this underscores the importance of discovering pages fast so that they can be properly archived before they disappear.

Re-Crawling for Content Freshness. Coffman et al. [10] were one of the first to study the problem of re-crawling, and postulated that web page change events follow a Poisson process: changes occur randomly and independently. Considering the Poisson model, Cho and Garcia-Molina [8] proposed efficient change frequency estimators for various scenarios, including the web crawling scenario in which information about all change events is not available. Other works observed that features extracted from the content of web pages, web link structure, and web search logs can be used to effectively predict change patterns [4, 24, 28]. Barbosa et al. [4] were the first to exploit the use of static features extracted from the content of a page to predict its change behavior. Based on this idea, Tan and Mitra [28] proposed the use of new dynamic features, and other features extracted from the web link structure and web search logs to group pages with similar change behavior. Radinsky and Bennett [24] went a step further and proposed a change prediction framework that uses not only features from the content, but also the degree and relationship among the observed changes to a page, the relatedness to other pages, and the similarity of changes. Santos et al. [26, 27] proposed the use of ranking-based re-crawling strategies for scheduling web page updates. They described a genetic programming framework to generate ranking functions to select the top-k best pages to be re-crawled. This line of work differs from our work in the sense that their goal is to maintain overall freshness, whereas in here we aim to discover new content. We follow a ranking-based approach as in [26, 27], but with different machine learning methods and objectives.

Re-crawling for Content Discovery. Dasgupta et al. [13] studied the extent to which new pages can be efficiently discovered. They observed that by re-crawling a subset of existing pages, it is possible to discover a large portion of the newly generated content. Another important finding in their work was that a large number of page pairs share outlinks to new content: if only pages within a web site are considered, more than 20% of pages have Jaccard coefficient very close to 0 (i.e., they do not share any links) and more than 40% have a Jaccard coefficient very close to 1 (i.e., they have the same links). They proposed algorithms for selecting a small subset of pages that can be re-crawled in order to efficiently discover new content. The algorithms assume that full snapshots of the content is available. However, for scenarios where resources are limited, constructing full snapshots may not be possible. In contrast, our approach does not assume complete information is available and dynamically learns patterns as information becomes available.

Kumar et al. [15] studied the problem of discovery for pages with a high number of incoming links. They presented different policies to select pages for re-crawling based on multiple page scoring strategies such as the *yield* (number of links not present in the previous crawl of that page divided by the time since the prior crawl) and the *link score* (the number of inlinks of the discovered pages). Other policies select pages with probability proportional to the number of total inlinks, new outlinks, or both. In summary, all proposed policies try to maximize discovery of high in-degree pages and only leverage the link structure of the Web, whereas our approach aims to maximize full content coverage and does so by using additional features to learning efficient re-crawling policies.

Gupta et al. [14] proposed techniques to schedule the re-crawling of a given set of seed pages to discover news headlines. Given the crawling frequency per day as a resource constraint, they proposed a technique that monitors each news source for a particular time period to collect the news update patterns, and then analyzes them using mixed integer programming to discover the optimal crawling schedule that maximizes accuracy. In another variation of the problem, they optimized the crawling frequency and the corresponding crawling schedule for a desired accuracy level. While Gupta et al. aim to find an optimal fixed schedule for each page, our approach is adaptive. We automatically learn a crawling policy that is applied at each timestamp to select the pages that should be crawled in order to maximize coverage. We do not assume that pages are monitored during a particular time, instead, the patterns are learned dynamically as more information becomes available. Furthermore, since their seed pages were selected manually, they ignore the fact that different pages may share outlinks to new content. This overlap can negatively affect the overall performance of the crawling policy. As we discuss in Section 4, our approach takes overlap into account.

Lefortier et al. [16] studied the problem of crawling popular pages (in terms of user interest) in a timely fashion. They showed that page popularity changes over time and many pages follow a pattern where the popularity decays quickly. They propose strategies for

holistic crawling which aim to attain a balance between crawling newly discovered links or re-crawling previously crawled pages (content sources). One of their best-performing strategies crawls content sources with priority that is proportional to the new link discovery rate, the popularity of the discovered pages, and the time since last crawl; to interleave downloads of content sources and newly discovered links, it simply crawls new links immediately after their discovery. In this work, we focus on the problem of scheduling when content sources should be re-crawled, but our approach can be easily extended to crawl all new links right after discovery. In addition, while Lefortier et al. [16] optimize discovery of popular pages based on search engine click logs, our main motivation is to obtain full coverage of a given domain.

3 PROBLEM FORMULATION

Let S be the set of n seed pages chosen for re-crawling and $T = t_0, t_1, ..., t_\infty$ be the discrete representation of timestamps for the re-crawling cycles. Let $O^t(s)$ denote a set of new pages discovered from $s \in S$ at the time $t \in T$. The set of all pages discovered from the seed pages is $O = \bigcup_{t \in T} O^t$. Let $r : O \rightarrow 0, 1$ be a function indicating the relevance of a page in O with respect to the domain D:

$$r(o) = \begin{cases} 1 \text{ if } o \in D \\ 0 \text{ if } o \notin D \end{cases}$$

At each timestamp t, let U_k^t be the set of distinct pages discovered from k seed pages $S_k^t \subset S$ for the first time:

$$U_k^t = \bigcup_{s \in S_k^t} O^t(s)$$

$R^t(S_k^t)$ denotes the number of distinct pages discovered from S_k^t that are relevant to domain D at time t:

$$R^t(S_k^t) = \sum_{p \in U^t(S_k^t)} r(p)$$

At each timestamp t, and given the information obtained from prior crawls $O^{t'}$, where $t' < t$, we would like to select S_k^t such that re-crawling S_k^t maximizes the cummulative $R^t(S_k^t)$ over time:

$$C_k = \sum_{t \in T} R^t(S_k^t)$$

We use *Coverage* to measure the efficiency of a selection strategy:

$$Coverage = \frac{C_k}{C_n}$$

where C_n is total number of relevant pages that can be discovered if all seed pages are recrawled at each timestamp. Note that *Coverage* captures the percentage of relevant pages discovered over time, and if all discovered pages are considered relevant (i.e., $O \subseteq D$ and $\forall o : r(o) = 1$), the problem is reduced to the general non-focused crawling setting, which we also consider in our evaluation.

4 ONLINE RE-CRAWLING FRAMEWORK

Unlike previously-proposed scheduling algorithms that require complete past crawls, we propose a method that learns from temporal features derived from multiple incomplete historical crawls. Below, we give an overview of the framework and then present our approach to prediction as well as how we balance exploration and exploitation during the re-crawling process to increase coverage.

4.1 Approach Overview

As outlined in Algorithm 1, our approach works in an iterative fashion. First, it builds a model (line 5) that, given a page P, predicts the number of new relevant links P will yield. Then, using this model (*Predictor*), it selects a small subset S_k^t of the seed pages S (line 6) to be crawled (in line 7) that maximizes the *Coverage* over time, i.e., S_k^t will yield a large number of new outlinks $O^t(S_k^t)$ if crawled at time t. Finally, based on the retrieved data, historical information is updated (lines 8-9) which will be used in subsequent iterations.

As discussed in Section 1, pages from different domains can have very different change rates [25]. Indeed, our experiments (see Section 5.2) confirm not only that, but also show that these rates change over time and depend on different factors. To tackle this problem, we apply learning algorithms to train a *Predictor* that uncovers such patterns from data obtained in previous re-crawling cycles (stored in *HIST*) and effectively predicts link yields. Thus, *HIST* needs to maintain the information $O^{t'}$ obtained from prior re-crawling cycles t', where $t' < t$, as well as the relevance of all discovered pages with respect to domain D (line 9). This information can then be used to generate features for the prediction (Section 4.2).

However, a significant fraction of page pairs have overlapping outlink sets [13]. Thus, if we greedly consider pages that lead to a high yield in isolation, we are likely to waste resources: if the pages selected have a large overlap, the overall yield will be low. To address this problem, we take overlap into account: if two seed pages are predicted to have a high yield but also have a high estimated overlap, only one should be selected for re-crawling. Since overlap can change over time, the overlap estimates must be updated regularly. In the previous approach [13], the overlap could be computed within a single snapshot from the most recent crawl since all seed pages are scheduled to be re-crawled at every timestamp. However, in our setting, only small subset of seed pages is re-crawled, therefore only $O(S_k)$ is available rather than $O(S)$. As a result, we need to estimate the overlap differently. To update *SIM* (line 8), we compute all pair-wise similarities between selected seed pages in S_k. Note that $|S_k| \ll |S|$ and we only need to compute the similarity between pages that are in the same web site, which significantly reduces the number of pairs to compute.

To measure the overlap between two pages p and q, we use the Jaccard similarity:

$$Jaccard(p, q) = \frac{|O(p) \cap O(q)|}{|O(p) \cup O(q)|}$$

Algorithm 2 details the page selection function. First, the predictor is applied to all pages to estimate the number of new outlinks that can be discovered from the seed pages (lines 2-5). Next, a greedy selection is performed over the candidate pages, taking into account the overlap estimate from the most recent crawls (lines 6-11).

4.2 Predicting New Outlinks

Given the information $O^0(S_k^0), O^1(S_k^1), ..., O^{t'}(S_k^{t'})$ collected during previous crawls, we need to predict the number of new pages discovered by re-crawling each seed page $s \in S$ at the timestamp t, where $t > t'$. We define the temporal prediction function as: $f : S \times T \rightarrow \mathbb{R}$ and the set of temporal training examples $X = \{x_0, x_1, ..., x_{t'}\}$, such that x_{t_i} contains set of training examples obtained by re-crawling

Algorithm 1 Online Re-Crawling

```
procedure Online_Recrawl(k)
    HIST = ∅
    SIM = ∅
    for t ∈ T do
        Predictor ← Train a model with HIST
        S_k^t = Select(k, t, Predictor, SIM)
        O^t(S_k^t) ← Crawl(S_k^t)        ▷ Crawl seed pages and new outlinks
        Update SIM with O^t(S_k^t)
        Update HIST with O^t(S_k^t)
    end for
end procedure
```

Algorithm 2 Select Top-k Seed Pages

```
function Select(k, t, Predictor, SIM)
    Candidates = { Seed Pages }
    for p ∈ Candidates do
        E_p = Predictor.predict(p, t)
    end for
    S_k = ∅
    for 1 to k do
        Select p in Candidates that has highest E_p
        Remove all q from Candidates, where SIM[p, q] > ξ
        S_k^t ← S_k^t ∪ p
    end for
    return S_k^t
end function
```

selected seed pages at timestamp t_i. Each seed page s that was re-crawled is represented as one training example: $\langle F(s,t), |O^t(s)| \rangle$, where $F(s,t)$ is the featurization function that transforms a page into a single vector and $|O^t(s)|$ is the number of new relevant pages yielded by s at timestamp t.

Features. We consider features that capture temporal content creation patterns. Note, however, that other features types (e.g., content and URL, etc) can be easily integrated into our framework. We leave this for future work. Each page re-crawled at timestamp t is converted into a single vector resulting from the concatenation of the fetures described below.

avg: *Average number of new pages discovered from s in the past crawls.* In order to limit the size of the data stored and to prioritize the recent page history, we compute this feature considering only a window of size w. For example, for $w = 10$, we compute avg using the number of discovered pages from the last 10 previous crawls.

std: *Standard deviation of the number of new pages discovered from s in the past crawls.* This feature captures the variance of the ***avg*** value. As in avg, we compute std within a window size w.

age: *Time since the seed page s was last re-crawled.* This feature aims to prioritize *older* seeds. The intuition is that if a seed page has not been re-crawled recently, it is more likely it contains previously unseen outlinks.

tod: *Time of the day of the timestamp t*. This feature takes a value from 0 to 23. We treat this value as the categorical feature and use one-hot encoding to represent it. The rationale for this feature is that some pages may have higher probability of creating new links during speficic times of the day (e.g., day or night).

dow: *Day of the week of the timestamp of the next crawling cycle.* This feature takes value from 0 to 6, corresponding to each day of the week (Monday is 0 and Sunday is 6). As in *tod*, we use one-hot encoding to represent this feature, and the rationale is that some pages may create more links during specific days of the week (e.g., during weekends). ***aa = avg∗age***: *Reinforcement of greedy strategy.* This feature gives higher priority to pages that generate more new links on average and have not been crawled for a long time. It also introduces non-linearity that would not be possible for simple linear algorithms to learn otherwise.

Learning Algorithms. Multiple learning methods can be used to map the features into predictions. Since we are trying to predict a number (the number of new outlinks), regression algorithms are a natural choice for this task. In practice, the actual predicted number is used only for ranking the pages. Therefore, learning to rank (LTR) [17] is another class of algorithms suitable for this task. In Section 5, we present an experimental evaluation of these two classes of algorithms for our problem.

4.3 Balancing Exploitation and Exploration

Since Algorithm 2 performs a greedy selection, it naturally favors seed pages that are known to be *good*. This can introduce bias in the estimator, which by learning just from this group may end up missing other, different seed pages that have high yield in the future, and consequently, will fail to capture their patterns. To address this problem, we propose a strategy to balance exploitation and exploration using multi-armed bandits (MAB) [2], which is known to be an effective means of solving such trade-off problems. The MAB algorithm operates as follows: at each timestamp t, we choose one of a set of arms available. Then, we observe the reward obtained by choosing that arm. The goal is to select arms such that the cumulative reward over time is maximized. One intuitive way to apply this method in our setting is to model two selection methods as arms, one which favors exploitation and another that favors exploration, and then select one arm at each re-crawling cycle. However, since exploration generally yields much lower coverage than exploitation, this causes the MAB algorithm quickly coverge and therefore rarely select the explorative method. As a result, we observed that the coverage obtained by this strategy is no different than using solely exploitation. Instead, we combine exploration and exploitation using a ratio r, i.e., we select r% of the seed pages using a greedy strategy (such as Algorithm 2), and $(1-r)$% using a strategy that favors exploration. Then, we model the arms as a finite list of ratios between 0.0 and 1.0 and apply the UCB1 algorithm [2] to choose the ratio to be used at each timestamp t. This strategy is detailed in Algorithm 3. Finally, after we crawl the selected pages we use the average number of discovered new outlinks over time by the choosen ratio as reward.

5 EXPERIMENTAL EVALUATION

We demonstrate the effectiveness of our re-crawling framework and prediction methods through extensive experiments using real-world datasets. To create a controlled environment in which we can accurately compare different strategies and avoid confounding factors that arise due to the dynamic nature of the Web, we

Algorithm 3 Balancing Exploitation and Exploration

```
function Bandit_Select(k, t, Predictor, SIM)
    ratio = UCB1.select_arm()
    k1 = ratio * k
    k2 = k − k1
    TopK1 = Select(k1, t, Predictor, SIM)
    TopK2 = Select k2 pages using exploration strategy
    TopK = TopK1 ∪ TopK2
    return TopK
end function
```

collected data from three different domains to serve as the ground truth. Using these data, we compared the coverage attained by our approach against the state of the art in re-crawling in both focused and non-focused settings. In the focused setting, we assume that the page classifier is available during the re-crawling process, while in the non-focused setting it is not. The non-focused setting reflects the scenario when the page classifier is not available and one wants to re-crawl all pages from a number of specific web sites that are considered fully relevant to the domain. We also verified the benefit of balancing exploration and exploitation, and the ability of our approach to discover new content in timely fashion.

5.1 Data Collection

To gather the ground truth data for carrying out the re-crawling experiments, we selected the following domains:

- Politics: represents news articles and blog posts that are related to politics.
- Human trafficking: represents adult ads from classified ads, escort, and related sites.
- Humanitarian crisis: contains reports about disasters that threaten human life from NGOs and news sites.

We selected these domains for two main reasons: they are very different in nature – this is confirmed in the different behaviors observed in our experimental analysis; and we had access to training examples, labeled by subject matter experts, which are critical to train a classifier required to test the relevance of the crawled pages.

For each domain, we selected the seed pages as follows. First, we fired a focused crawl using the ACHE open-source crawler[2] and retrieved between 200K to 500K pages. Note that each crawl makes use of a classifier pre-trained for the respective domain. We then sorted all pages based on the number of relevant outlinks they contain and selected pages with the highest number of outlinks, while at the same time, avoiding gathering too many pages from the same site. Besides covering a larger number of sites, we need to limit the number of pages per site so that it is possible to obtain full snapshots in hourly intervals. This also reduces the risk of the crawler being blocked by the target web sites. For our experiments, by setting the page limit to 10 in politics domain and 50 in human trafficking and humanitarian crisis domain, we can create full snapshots hourly.

The ground-truth data was obtained by crawling all seed pages and the new pages from their outlinks (i.e., ones that were not discovered in previous crawls) hourly during a 1-month period. From the crawled data, we extracted in-site outlinks from the HTML source of each seed page and ran the page classifiers to assess the relevance of pages linked by these outlinks. Table 1 summarizes the statistics of the collected data.

[2]http://github.com/ViDA-NYU/ache

Table 1: Statistics of the collected data

	Human Trafficking	Politics	Hum. Crisis
# seed pages	8543	17892	3480
# seed sites	2215	3524	96
average # pages per site	2.59	5.07	36.25
# discovered pages	2,012,764	682,845	43,012
# discovered relevant pages	1,004,075	243,482	18,532

5.2 Discoverability Patterns

We obtained 665, 632, and 741 snapshots for human trafficking, humanitarian crisis and politics domains respectively. The total size of these data is roughly 2.7TB uncompressed. To better understand the patterns associated with the discoverability of new pages, we examined the number of new pages discovered at each re-crawling cycle. Figure 1 shows the number of new pages and new relevant pages that are directly reachable from the seed pages in the three domains. The figures clearly show that percentage of relevant pages among new pages is small, which underscores the importance of optimizing the selection strategies with respect to relevant pages. In all domains, the following pattern is observed: the number of new links starts to drop in the beginning and then stabilizes after the first 50 hours of the crawls. We initially hypothesized that the drop was due to the crawler being blocked by sites. But after inspecting the logs, we found that the number of successful requests to seed pages remains stable even at the beginning of the process. The actual reason for the drop is that this set of discovered pages already existed before the start of our crawl, and were only gradualy discovered over time after they appear in the monitored seed pages. The steeper decline in the human trafficking domain reveals that pages from the past tend to appear more often. Note that the decline for new relevant pages is less steep, suggesting that irrelevant pages (e.g., boilerplate ads) are more likely to appear repeatedly. In the plots, we also observe repeating valleys and peaks. These correspond to daily and weekly patterns, which indicate the importance of capturing the temporal features (time of the day and day of the week) described in Section 4.2.

5.3 Comparing Selection Strategies

We compare our approach with several crawling strategies. The Round Robin (*RR*) strategy selects seed pages that have the highest ***age*** (time since last crawl). It is a uniform strategy that guarantees that all pages are re-crawled once every n/k re-crawling cycles, where n is the number of seed pages. We also compare to the upper-bound greedy algorithm (*GREEDY*) from [13], which greedily selects the best pages with foreknowlege of all outlinks $O^t(S)$ and taking the overlap into account. It is unrealistic since it assumes full knowledge of $O^t(S)$ before crawling S at time t, but it serves as an approximation of the ideal upper bound coverage. The strategies *OD-WIN*, *CLIQ-WIN** and *COV** are variants of the algorithms proposed in [13]. We use * in a algorithm name to denote that the

original algorithm needed to be slightly modified to be applicable in our problem setting. In what follows, we explain each of them.

OD-WIN: This algorithm selects the top-k seed pages with the highest average of new relevant pages discovered in the past crawls.

*COV**: The original algorithm *COV* is the realistic version of *GREEDY*, which utilizes $O^{t-1}(S)$ for selecting seed pages at timestamp t. However, in our setting we only crawl k pages at each timestamp t, then only $O^{t-1}(S_k^{t-1})$ is available. One solution for this is to merge the set of outlinks from all past crawls as follows: for each page $s \in S$, let $\hat{O}^{t-1}(s) = \bigcup_{t'<t} O^{t'}(s)$, where $t' < t$. Then we apply the *GREEDY* algorithm, but using only past information $\hat{O}^{t-1}(S)$.

*CLIQ-WIN**: In its original form, the algorithm groups pages based on the overlap identified in the most recent re-crawl, and then picks one representative page from each group, starting from the one with highest estimated number of new pages. The estimation is done by computing the average number of new relevant pages discovered in the past crawls. To make this work in our setting, where all past re-crawls are incomplete, we first compute the overlap as described in Section 4. Then, we use this information to form groups and pick only one page from each group.

Prediction-Based Methods. We evaluate our framework (Section 4) using four different configurations:

REG: In this strategy we apply the Algorithms 1 and 2 described in Section 4, and use linear regression to train the *Predictor*. Note that we do not use exploration in this method.

LTR: This strategy is similar to REG, but instead of linear regression, it applies learning-to-rank. More specificaly, we represent a training example as $\langle F(s,t), log(1+|O^t(s)|), qid \rangle$, where $log(1+|O^t(s)|)$ is the relevance of the document and qid is the query. We set the qid to be equal to the timestamp t when the training example was created.

BANDIT: This method uses the bandit algorithm described in Section 4.3 to choose the ratio between *REG* and *RR*.

REG-RR: To evaluate the benefit of using the bandits algorithm, we also experiment with a strategy that combines *REG* and *RR* using a fixed ratio. We fix the ratio between *REG* and *RR* to be 0.9, which we empirically found to be the best in humanitarian crisis domain. We pick *REG* over *LTR* to combine with *RR* and *BANDIT* due to its better speed and coverage performance as shown later.

Implementation Details. Any available machine learning toolkit to learn the predictor described in Section 4.2; for our experiments, we used scikit-learn [22]. To learn a regression-based predictor, we considered three options: Gradient Boosting Decision Trees (GBDT), Linear Regression (LR), and Polynomial Regression (PR). However, we observed no significant improvement by using PR and GBDT over LR, therefore we only show the results obtained using LR. To construct a learning-to-rank-based predictor, we use RankLib [12] and therefore have multiple algorithm options. We picked Coordinate Ascent [18] over others to show in the final results due to its superior performance. In all re-crawl simulations using *REG*, *REG-RR*, *BANDIT* and *LTR*, we use *OD-WIN* method during the first 10 hours to bootstrap the training data. We set window size w to 168 hours (7 days) so that the predictor only learns from most recent crawls without losing the daily and hourly patterns. The predictor is re-trained every 3 hours to reduce the computation time. We set ξ in the Algorithm 2 to 0.7. We set k to 50, 400, 500 in humanitarian crisis, human trafficking and politics domain respectively. We pick the k close to the minimal value that the *GREEDY* method can obtain 100% coverage. For the *BANDIT* method, we choose the set of possible ratio values as $\{0.6, 0.7, 0.8, 0.9, 1.0\}$. We intentionally omit values smaller than 0.6 to force the algorithm to always favor exploitation over the exploration. Ideally, the bandit algorithm should figure this out, however this is likely to hurt coverage since it may take several re-crawling cycles to get to the optimal value.

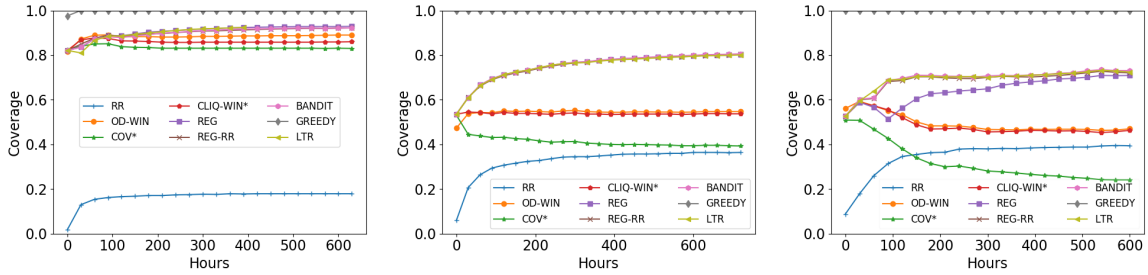
Table 2: Coverage of selection methods in a focused setting

	Human Trafficking	Politics	Humanitarian Crisis
RR	0.179	0.368	0.391
OD-WIN	0.890	0.546	0.473
*COV**	0.829	0.393	0.465
*CLIQ-WIN**	0.859	0.538	0.465
REG	**0.929**	0.805	0.714
REG-RR	0.922	0.807	0.723
BANDIT	0.921	**0.808**	**0.735**
LTR	0.924	0.804	0.724

Comparison in a Focused Setting. In this setting, we only consider a page to be relevant if it is classified as positive by the page classifier for the corresponding domain. Figure 2 shows how coverage changes over time for the different strategies, while Table 2 shows the coverage obtained at the last timestamp. We observe that our approach outperforms all the baselines. In Humanitarian Crisis and Politics, *BANDIT* attains more than 150% of the coverage obtained by all the baselines.

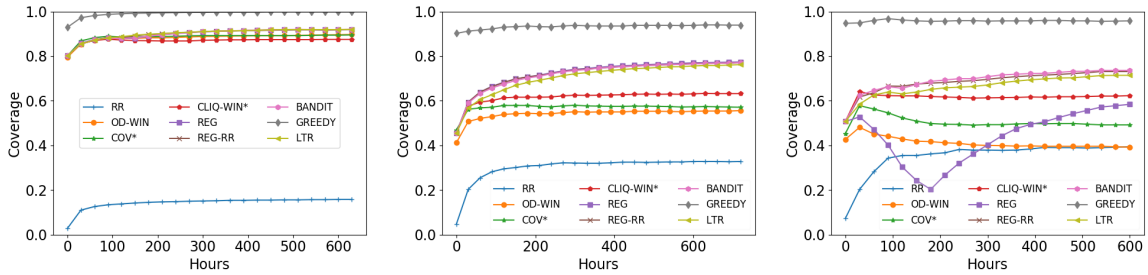
For Human Trafficking, the difference in coverage is smaller. As we observed in the Figure 1, this domain does not have the daily and weekly patterns present in the other domains, and the variation in the number of new pages hourly is also comparatively small. As a result, all baselines perform well in this *easy* domain. This suggests that our methods are especially effective for domains that present higher variability in the the generation of new pages. Note that for this domain, *OD-WIN*, which does not take overlap into account, is the best baseline. This can be explained by Table 1. Indeed, this domain has the smallest average number of pages per site. Therefore, it is likely that there is little overlap among the selected seed pages. Another interesting observation is that in Humanitarian Crisis domain, *COV** performs worst. One potential explanation for this is that, in this domain, the number of new relevant pages discovered per seed page in each hour is very small, which makes the estimation $\hat{O}^{t-1}(S)$ less accurate.

Comparing *REG*, *REG-RR* and *BANDIT*, their performance is comparable in Human Trafficking and Politics domain. However, in Humanitarian Crisis domain, the *BANDIT* method obtains best result and *REG* is the worst. As Figure 1 shows, this domain has the highest variability in the generation of new pages. This suggests that depending on the nature of the data to be collected, using exploration may or may not lead to improvements. However, *BANDIT* provides the best solution: by balancing exploration and exploitation in a dynamic fashion, it is able to adapt automaticall for different domains.



(a) Human Trafficking domain (b) Politics domain (c) Humanitarian Crisis domain

Figure 2: Comparison of coverage between baselines and prediction based methods in focused setting



(a) Human Trafficking domain (b) Politics domain (c) Humanitarian Crisis domain

Figure 3: Comparision of coverage between baselines and prediction based methods in non-focused setting

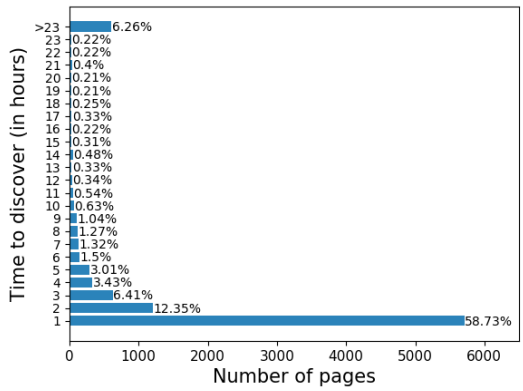


Figure 4: Age of pages discovered by the best method in focused setting from Humanitarian Crisis domain

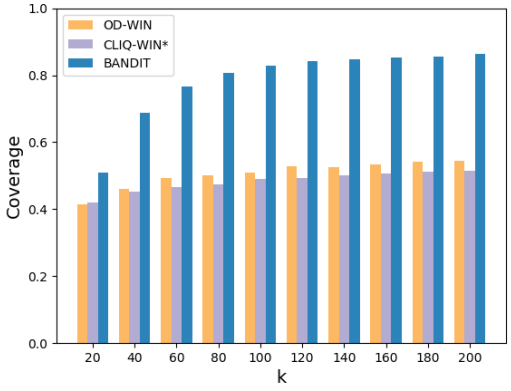


Figure 5: Coverages obtained by varing k in focused setting from Humanitarian Crisis domain

Comparison in a Non-Focused Setting. While the goal of our work is to support efficient discovery of pages in a domain, we are also interested in assessing the effectiveness of our approach for general discovery. To do so, we consider all discovered pages as relevant. Figure 3 shows how coverage varies over time for the different strategies. Although the magnitude of the difference between these methods seem to be smaller, our proposed methods still outperform all the baselines. Aside from that, we also observe that *REG* performs much worse than *BANDIT* and *REG-RR* in Humanitarian Crisis domain, which shows the effectiveness of combining exploration and exploitation. Another observation is that *CLIQ-WIN** obtains better coverage than *OD-WIN*, especially in Humanitarian Crisis domain. The reason is that the number of new pages discovered per site each hour in the non-focused setting is much higher than that in the focused one. This leads to larger overlaps between seed pages, which is considered in *CLIQ-WIN**. Note that in our data collection process, we limit the number of seed pages per site to be quite small, which likely reduces the overlaps. However, when the limit is higher, methods that eliminate the overlaps would be more effective.

5.4 Age of Discovered Pages

To demonstrate the efficiency of our method to discover relevant pages in a timely manner, we analyze the age (i.e., time interval

since publication until discovery) of relevant pages when they were discovered using our best method–*BANDIT*. Figure 4 shows that in the Humanitarian Crisis domain, 58.73% of relevant pages were discovered within 1 hour of publication; and over 80% within 4 hours. Note that in this setting, we set k to 50, which is equivalent as re-crawling the entire seed pages every 2.8 days. Similar results were obtained for the other domains, which we omit due to space limitation.

5.5 Impact of k on Coverage

In the previous experiments, we fixed k in all settings for the sake of comparison. We now explore how varying values of k impacts the coverage for different selection methods. For this experiment with focused setting in Humanitarian Crisis domain, we use our best method - *BANDIT*, and the two best baselines - *OD-WIN* and *CLIQ-WIN*. As Figure 5 shows, *BANDIT* outperforms the other methods even when we vary k. We note that in a production environment, k can be set based on resource availability (i.e., maximum number of pages to be crawled in each re-crawling cycle) or politeness constraints (i.e., maximum number of pages to be crawled from single web site in each re-crawling cycle).

6 CONCLUSION AND FUTURE WORK

In this paper, we addressed the problem of efficiently discovering Web content for a given a domain of interest. We proposed a new framework whose goal is to attain high coverage for the domain while at the same time discovering new, relevant content in a timely fashion. Our framework is adaptive and updates the re-crawling schedules dynamically. At each re-crawling step, it selects the top-k seed pages that maximize the yield of new content using information collected in previous steps. Unlike previous approaches which assumed the crawler has full knowledge of how pages change over time, the framework incrementally learns the change patterns as the crawl proceeds and more knowledge about pages is acquired. In addition, to increase coverage, it also takes overlap into account and also balances exploration and exploitation. We performed a detailed experimental evaluation which shows that our approach is effective and outperforms previously-proposed, state-of-the-art methods. In future work, we plan to extend this work in several directions. First, a larger scale experiment, with a larger number of seeds and a longer data collection period, would allow a better understanding of the behavior and effectiveness of the multi-armed bandits approach over the prediction-based methods in the long run. Another direction that we would like to investigate is the effectiveness of content-based features, which were successfully employed to predict page changes in previous work [24], and may be correlated with the publication rates of new outlinks as well.

Acknowledgments. This work was supported by the DARPA MEMEX and D3M programs. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

REFERENCES

[1] Eytan Adar, Jaime Teevan, Susan T. Dumais, and Jonathan L. Elsas. The web changes everything: Understanding the dynamics of web content. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining*, pages 282–291, 2009.

[2] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, May 2002.

[3] Ziv Bar-Yossef, Andrei Z. Broder, Ravi Kumar, and Andrew Tomkins. Sic transit gloria telae: Towards an understanding of the web's decay. In *Proceedings of the 13th International Conference on World Wide Web*, pages 328–337, 2004.

[4] Luciano Barbosa, Ana Carolina Salgado, Francisco de Carvalho, Jacques Robin, and Juliana Freire. Looking at both the present and the past to efficiently update replicas of web content. In *Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management*, pages 75–80, 2005.

[5] Jeff Borland, Peter Dawkins, David Johnson, and Ross Williams. United nations disaster assessment and coordination, 2013.

[6] Vanessa Bouché. A report on the use of technology to recruit, groom and sell domestic minor sex trafficking victims. https://www.wearethorn.org/child-trafficking-statistics, January 2015.

[7] Soumen Chakrabarti. Focused web crawling. In *Encyclopedia of Database Systems*, pages 1147–1155. Springer, 2009.

[8] Junghoo Cho and Hector Garcia-Molina. Estimating frequency of change. *ACM Transactions on Internet Technology*, 3:256–290, 2003.

[9] Sandra Cleva and Gavin Shire. Global wildlife trafficking busts expose illegal internet sales in u.s., southeast asia. https://www.fws.gov/home/newsroom/operationwildwebNR07112013.html, July 2013.

[10] E. G. Coffman, Zhen Liu, and Richard R. Weber. Optimal robot scheduling for web search engines. *Journal of Scheduling*, 1(1):15–29, 1998.

[11] Ford D, Grimes C, and Tassone E. Keeping a search engine index fresh: Risk and optimality in estimating refresh rates for web pages. Technical report, 2008.

[12] Van Dang. The Lemur Project-Wiki-RankLib. https://sourceforge.net/p/lemur/wiki/RankLib/. [Online].

[13] Anirban Dasgupta, Arpita Ghosh, Ravi Kumar, Christopher Olston, Sandeep Pandey, and Andrew Tomkins. The discoverability of the web. In *Proceedings of the 16th International Conference on World Wide Web*, pages 421–430, 2007.

[14] Kanik Gupta, Vishal Mittal, Bazir Bishnoi, Siddharth Maheshwari, and Dhaval Patel. Act: Accuracy-aware crawling techniques for cloud-crawler. *World Wide Web*, 19(1):69–88, Jan 2016.

[15] Ravi Kumar, Kevin Lang, Cameron Marlow, and Andrew Tomkins. Efficient discovery of authoritative resources. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 1495–1497, 2008.

[16] Damien Lefortier, Liudmila Ostroumova, Egor Samosvat, and Pavel Serdyukov. Timely crawling of high-quality ephemeral new content. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 745–750, 2013.

[17] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, March 2009.

[18] Donald Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Infomation Retrieval*, 10(3):257–274, June 2007.

[19] Alexandros Ntoulas, Junghoo Cho, and Christopher Olston. What's new on the web?: The evolution of the web from a search engine perspective. In *Proceedings of the 13th International Conference on World Wide Web*, pages 1–12, 2004.

[20] Mary Catherine O'Connor. Inside the complicated world of online wildlife trafficking. https://www.theguardian.com/vital-signs/2015/aug/03/cecil-lion-ivory-online-wildlife-trafficking-ebay, August 2015.

[21] Christopher Olston and Sandeep Pandey. Recrawl scheduling based on information longevity. In *Proceedings of the 17th International Conference on World Wide Web*, pages 437–446, 2008.

[22] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, November 2011.

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

[24] Kira Radinsky and Paul N. Bennett. Predicting content change on the web. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, pages 415–424, 2013.

[25] Aécio Santos, Bruno Pasini, and Juliana Freire. A first study on temporal dynamics of topics on the web. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 849–854, 2016.

[26] Aécio S. R. Santos, Cristiano R. Carvalho, Jussara M. Almeida, Edleno S. Moura, Altigran S. Silva, and Nivio Ziviani. A genetic programming framework to schedule webpage updates. *Infomation Retrieval*, 18(1):73–94, February 2015.

[27] Aécio S. R. Santos, Nivio Ziviani, Jussara Almeida, Cristiano R. Carvalho, Edleno Silva Moura, and Altigran Soares Silva. Learning to schedule webpage updates using genetic programming. In *Proceedings of the 20th International Symposium on String Processing and Information Retrieval - Volume 8214*, pages 271–278, 2013.

[28] Qingzhao Tan and Prasenjit Mitra. Clustering-based incremental web crawling. *ACM Transactions on Information Systems*, 28:17:1–17:27, 2010.