Modeling Updates of Scholarly Webpages Using Archived Data

Yasith Jayawardana
Computer Science Department
Old Dominion University
Norfolk, VA, USA
yasith@cs.odu.edu

Alexander C. Nwala
Center for Complex Networks and Systems Research
Luddy School of Informatics, Computing, and Engineering
Indiana University, Bloomington, IN, USA
anwala@iu.edu

Gavindya Jayawardena
Computer Science Department
Old Dominion University
Norfolk, VA, USA
gavindya@cs.odu.edu

Jian Wu
Computer Science Department
Old Dominion University
Norfolk, VA, USA
jwu@cs.odu.edu

Sampath Jayarathna, Michael L. Nelson
Computer Science Department
Old Dominion University
Norfolk, VA, USA
{sampath,mln}@cs.odu.edu

C. Lee Giles
Information Sciences & Technology
Pennsylvania State University
University Park, PA, USA
giles@ist.psu.edu

2020 IEEE International Conference on Big Data (Big Data) | 978-1-7281-6251-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/BigData50022.2020.9377796

***Abstract*—The vastness of the web imposes a prohibitive cost on building large-scale search engines with limited resources. Crawl frontiers thus need to be optimized to improve the coverage and freshness of crawled content. In this paper, we propose an approach for modeling the dynamics of change in the web using archived copies of webpages. To evaluate its utility, we conduct a preliminary study on the scholarly web using 19,977 seed URLs of authors' homepages obtained from their Google Scholar profiles. We first obtain archived copies of these webpages from the Internet Archive (IA), and estimate when their actual updates occurred. Next, we apply maximum likelihood to estimate their mean update frequency (λ) values. Our evaluation shows that λ values derived from a short history of archived data provide a good estimate for the true update frequency in the short-term, and that our method provides better estimations of updates at a fraction of resources compared to the baseline models. Based on this, we demonstrate the utility of archived data to optimize the crawling strategy of web crawlers, and uncover important challenges that inspire future research directions.**

***Index Terms*—Crawl Scheduling, Web Crawling, Search Engines**

I. Introduction

The sheer size of the Web makes it impossible for small crawling infrastructures to crawl the entire Web to build a general search engine comparable to Google or Bing. Instead, it is more feasible to build specialized search engines, which employ *focused web crawlers* [1], [2] to actively harvest webpages or documents of particular topics or types. Google Scholar, for instance, is a specialized search engine that is especially useful for scientists, technicians, students, and other researchers to find scholarly papers.

The basic algorithm for a focused web crawler is straightforward. The crawl frontier is first initialized with seed URLs that are relevant to the search engine's focus. Next, the crawler visits webpages referenced by seed URLs, extracts hyperlinks in them, selects hyperlinks that satisfy preset rules (to ensure that only related webpages are visited), adds them to the crawl frontier, and repeats this process until the crawl frontier exhausts [3]. Although this works for relatively short seed lists, it does not scale for large seed lists. For instance, the crawler may not finish visiting all webpages before they change. Given such circumstances, re-visiting web pages that have not changed since their last crawl is a waste of time and bandwidth. It is therefore important to select and prioritize a subset of seeds for each crawl, based on their likeliness to change in the future.

Without sufficient crawl history, it is difficult to accurately predict when a webpage will change. Web archives, such as the well-known Internet Archive's (IA) Wayback Machine [4] and others, preserve webpages as they existed at particular points in time for later replay. The IA has been collecting and saving public webpages since its inception in 1996, and contains archived copies of over 424 billion webpages [5], [6]. The resulting record of such archived copies is known as a *TimeMap* [7] and allows us to examine each saved copy to determine if a change occurred (not every saved version will represent a change in the webpage). TimeMaps provide a critical source of information for studying changes in the web. For example, if a researcher created his website in 2004, via a TimeMap we could retrieve copies of the website observed by the IA between 2004 and 2020, and examine these copies for changes.

In this paper, we propose an approach to model the dynamics of change in the web using archived copies of webpages. Though such dynamics have been studied in previous papers, e.g., [8]–[10], online activities have evolved since then, and to the best of our knowledge, the use of archived data to model these dynamics has not been explored. While many web archives exist, we use the IA to obtain archived copies of webpages due to its high archival rate, and efficiency of mass queries. Given a URL, we first obtain its TimeMap from the IA's Wayback Machine, and identify mementos that represent updates. Next, we use this information to estimate their mean update frequency (λ). We then use λ to calculate the probability (p) of seeing an update d days after it was last updated. Before each crawl, we repeat this process for each seed URL and use a threshold (θ) on p to select a subset of seed URLs that are most likely to have changed since their next crawl.

Our preliminary analysis demonstrates how this approach can be integrated into a focused web crawler, and its impact on the efficiency of crawl scheduling. Here, we select the scholarly web as our domain of study, and analyze our approach at both homepage-level (single webpage) and at website-level (multiple webpages). The former, investigates changes occurring on an author's homepage, while the latter, investigates changes occurring collectively on the homepage and any webpage behind it, e.g., *publications*, *projects*, and *teaching* webpages. Our contributions are as follows:

1) We studied the dynamics of the scholarly web using archived data from the IA for a sample of 19,977 authors' websites.
2) We verified that the updates to authors' websites and homepages follow a near-Poisson distribution, with spikes that may represent non-stochastic activities.
3) We developed *History-Aware Crawl Scheduler* (HACS), which uses archived data to find and schedule a subset of seed URLs that are most likely to have changed before the next crawl.
4) We compared HACS against baseline models for a simulated web crawling task, and demonstrated that it provides better estimations.

A. Crawling the Web

Although the basic focused web crawling algorithm [3] is simple, challenges in the web, such as scale, content selection trade-offs (e.g., coverage vs freshness), social obligations, and adversaries, makes it infeasible to crawl the web in that manner. Crawl frontiers should thus be optimized to improve the robustness of web crawlers. One approach is to reorder the crawl frontier to maximize some goal (e.g., bandwidth, freshness, importance, relevance) [11], [12]. *Fish-Search* [13], for instance, reorders the crawl frontier based on content relevance, and is one of the earliest of such methods. Given a seed URL and a driving query, it builds a priority queue that prioritizes webpages (and their respective out-links) that match the driving query. *Shark-Search* [14] is an improved version of Fish-Search which uses cosine similarity (number between 0 and 1) to calculate the relevance of a webpage to the driving query, instead of binary similarity (either 0 or 1) used in Fish-Search. Such algorithms do not require the crawl history to calculate relevance, and can be applied at both the initial crawl and any subsequent crawls.

In incremental crawling, webpages need to be re-visited once they change, to retain the freshness of their crawled copies. Several methods have been proposed [15], [16]. Olston et. al. [17], for instance, studied the webpage revisitation policy that a crawler should employ to achieve good freshness. They considered information longevity, i.e., the lifetime of content fragments that appear and disappear from webpages over time, to avoid crawling ephemeral content such as advertisements, which have limited contribution to the main topic of a webpage. Such methods require sufficient crawl history to identify ephemeral content, and until sufficient crawl history is generated, the algorithm may yield sub-optimal results.

Algorithms proposed by Cho et al. [18], reorders the crawl frontier based on the importance of webpages. Here, the *query similarity* metric used in Fish-Search and Shark-Search was extended with additional metrics such as, *back-link count*, *forward-link count*, *PageRank*, and *location* (e.g., URL depth, top-level domain). Alam et al. [19] proposed a similar approach, where the importance of a webpage was estimated using *PageRank*, *partial link structure*, *inter-host links*, *webpage titles*, and *topic relevance* measures. Although such methods take advantage of the crawl history, the importance of a webpage may not reflect how often it changes. Thus, such methods favour the freshness of certain content over the others.

Focused web crawlers should ideally discover all webpages relevant to its focus. However, the coverage that it could achieve depends on the seed URLs used. Wu et al. [20], for instance, proposed the use of a whitelist and a blacklist for seed URL selection. The whitelist contains high-quality seed URLs selected from parent URLs in the crawl history, while the blacklist contains seed URLs that should be avoided. The idea was to concentrate the workforce to exploit URLs with potentially abundant resources. In addition, Zheng et al. [21] proposed a graph-based framework to select seed URLs that maximize the value (or score) of the portion of the web graph "covered" by them. They model this selection as a *Maximum K-Coverage Problem*. Since this is a NP-hard [22] problem, the authors have proposed several greedy and iterative approaches to approximate the optimal solution. Although this works well for a general web crawler, studies show that the scholarly web has a disconnected structure [23]. Hence, the process of selecting seed URLs for such use cases may benefit from the crawl records of a general web crawler.

CiteSeerX [24] is a digital library search engine that has more than 10 million scholarly documents indexed and is growing [25]. Its crawler, identified as *citeseerxbot*, is an incremental web crawler that actively crawls the scholarly web and harvests scholarly papers in PDF format [25]. Compared to general web crawlers, crawlers built for the scholarly web has different goals in terms of optimizing the freshness of their content. The crawl scheduling model used by *citeseerxbot*, which we refer to as the *Last-Obs* model, prioritizes seed URLs based on the *time elapsed since a webpage was last visited*. In this work, we use the *Last-Obs* model as a baseline to compare with our method.

B. Modeling Updates to a Webpage

Updates to a webpage can be modeled as a Poisson process [9], [26], [27]. The model is based on the following theorem.

Theorem 1: If T is the time of occurrence of the next event in a Poisson process with rate λ (number of events per unit time period), the probability density for T is

$$f_T(t) = \lambda e^{-\lambda t}, \quad t > 0, \quad \lambda > 0. \tag{1}$$

Here, we assume that each update event is independent. While this assumption is not always true (i.e. certain updates are correlated), as shown later, it is a reasonable estimation. By integrating $f_T(t)$, we obtain the probability that a certain webpage changes in interval $[t_0, t]$:

$$P(\Delta t) = \int_{t_0}^{t} f_T(t)\, dt = 1 - e^{-\lambda \Delta t} \tag{2}$$

Note that the value of λ may vary for different webpages. For the same webpage, λ may also change over time but for a short period of time, λ is approximately constant. Therefore, by estimating λ, we calculate how likely a webpage will be updated since its last update at time t_c. Intuitively, λ can be estimated using,

$$\hat{\lambda} = X/T \tag{3}$$

in which X is the number of updates detected during n accesses, and T is the total time elapsed during n accesses. As proven in [9], this estimator is biased and it is more biased when there are more updates than accesses in the interval T. For convenience [26] defines an intermediate statistical variable $r = \lambda/f$, the ratio of the update frequency to the access frequency. An improved estimator was proposed below:

$$\hat{r} = -\log\left(\frac{\bar{X}+0.5}{n+0.5}\right), \quad \bar{X} = n - X. \tag{4}$$

This estimator is much less biased than X/T and i It is also consistent, meaning that as $n \to \infty$, the expectation of $\hat{r}$ is r.

Unfortunately, since archival rates of the IA depend on its crawl scheduling algorithm and the nature of the webpages themselves, its crawl records have irregular intervals. As a result, archived copies may not reflect every update that occurred on the live web, and not all consecutive archived copies may reflect an update. Since both Eq. (3) and Eq. (4) assume regular access, they cannot be used directly. To address this limitation, we use a *maximum likelihood estimator* to calculate which λ is most likely to produce an observed set of events.

$$\sum_{i=1}^{m} \frac{t_{c_i}}{\exp(\lambda t_{c_i}) - 1} = \sum_{j=1}^{n-m} t_{u_j}, \tag{5}$$

Here, t_{c_i} is the i-th time interval where an update was detected, t_{u_j} is the j-th time interval where an update was *not* detected, and m is the total number of updates detected from n accesses (see Figure 1). λ is calculated by solving Eq. (5). Since this equation is nonlinear, we solve it numerically using Brent's method [28]. There is a special case when $m = n$ (i.e. updates detected at all accesses) where solving Eq. (5) yields $\lambda = \infty$. In this case, Eq.(5)'s solution is infinity and Eq.(4) is used.

To the best of our knowledge, there has not been an open source crawl scheduler for the scholarly web that takes advantage of the update model above. With IA providing an excellent, open-accessible resource to model the updates of scholarly webpages, this model can be applied on focused crawl schedulers to save substantial time on crawling and re-visitation.

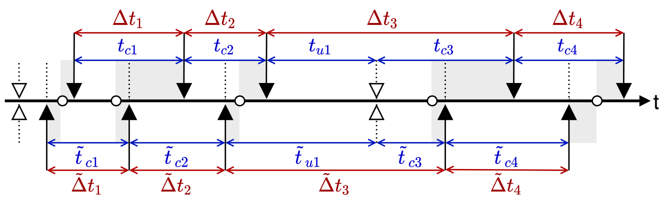


Fig. 1. An illustration of accesses (▽,△), accesses with updates (▼), true update occurrences (○) and the interpolated update occurrences (▲) over time. Gray shades represent the deviation of the observed and interpolated update occurrences from the true update occurrences.

II. Methodology

A. Data Acquisition

The seed list used in this work was derived from a dataset containing Google Scholar profile records of *396,423* researchers. This dataset was collected around 2015 by scraping profile webpages in Google Scholar with a long crawl-delay. The steps for data acquisition and preparation are illustrated in Figure 2.

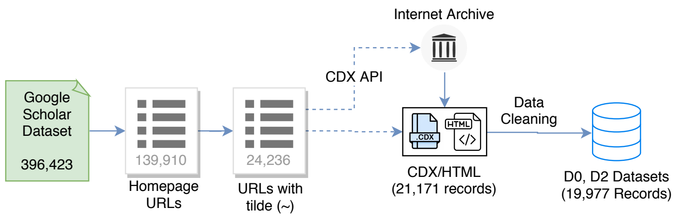


Fig. 2. Steps followed to acquire and prepare data from IA (depths 0–2).

Step 1: From the Google Scholar profile records, we discovered *139,910* profiles that provided homepage URLs. These URLs referenced either individual author homepages, or organizational websites. Since our study focused on modeling the dynamics of the websites of individual authors, we removed organizational websites. This was nontrivial using a simple rule-based filter as there were personal homepages that look similar to organizational homepages. Therefore, we restricted our scope to homepage URLs hosted *within a user directory* of an institution, i.e., URLs with a tilde (∼) in them (e.g., foo.edu/∼bar/). In this manner, we obtained *24,236* homepage URLs.

Step 2: Next, we performed a wildcard query on the IA Wayback CDX Server API [29] to obtain TimeMaps for each author website under their homepage URL. Out of *24,236* websites, we obtained TimeMaps for *21,171* author websites (87.35% archival rate). The remaining websites were either not archived, or the CDX Server API returned an error code during access. The resulting TimeMaps provided information such as the crawl timestamps and URI-Ms of archived copies of each webpage. From these webpages, we selected webpages at depth ≤2 (Depth 0 is the homepage). For instance, for a homepage foo.edu/~bar, a link to foo.edu/~bar/baz is of depth 1 and is selected. However a link to foo.edu/~bar/baz/qux/quux is of depth 3 and is not selected.

Step 3: Next, we generated the **D0 dataset** and **D2 dataset**, which we use in our analysis. First, we de-referenced the URI-Ms of each URL selected in Step 2, and saved their HTML for later use. When doing so, we dropped inconsistent records such as records with invalid checksum, invalid date, multiple depth 0 URLs, and duplicate captures from our data. The resulting data, which we refer to as the **D2 dataset**, contained HTML of **19,977** websites, totaling **581,603** individual webpages. The average number of webpages per website is 227.49. The minimum and maximum number of webpages per website are 1 and 35,056, respectively. We selected a subset of the **D2 dataset** consisting HTML of only the **19,977** homepages, which we refer to as the **D0 dataset**.

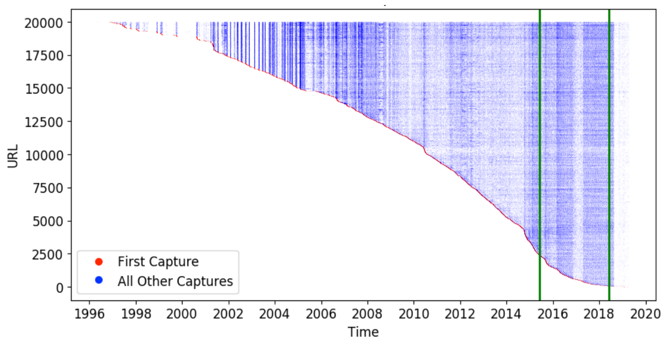


Fig. 3. Captures (blue dots) of homepage URLs over time, with URLs sorted by their earliest capture time (red dots). The captures between 2015-06-01 and 2018-06-01 (green vertical lines) were used for the evaluation.

Figure 3 shows the distribution of captures in the D0 dataset, sorted by their earliest capture time. Here, the median crawl interval of 80% of author homepages were between $20 - 127$ days. The distribution of capture density over time suggests that the capture densities of IA vary irregularly with time. For instance, captures during 2015–2018 show a higher density on average than the captures during 2010–2014. Since high-cadence captures help to obtain a better estimation for the update occurrences, we scoped our analysis to the period between June 1, 2015 and June 1, 2018, (shown by green vertical lines in Figure 3).

B. Estimating Mean Update Frequency

The exact interpretation of update may differ depending on the purpose of study. We examine a specific type of update – **the addition of new links**. The intuition here is to identify when authors add *new publications* into their webpages, as opposed to identifying when that webpage was updated in general. We claim that this interpretation of update is more suited to capture such behavior.

For each webpage in datasets **D0** and **D2**, we processed each capture m_i to extract links $l(m_i)$ from its HTML, where $l(m_i)$ is the set of links in the i^{th} capture. Next, we calculated $|l^*(m_i)|$, i.e., the number of links in a capture m_i that was never seen before m_i, for each capture in these datasets. Formally,

$$l^*(m_i) = l(m_i) - \cup_{k=1}^{i-1} l(m_k), \quad i \geq 2.$$

and $\cup_{k=1}^{i-1} l(m_k)$ is the union of links from captures m_1 to m_{i-1}. Finally, we calculated the observed-update intervals $t_{c_i} \in T_c$ and observed non-update intervals $t_{u_j} \in T_u$ based on captures that show link additions, i.e., $l^*(m_i) > 0$ and ones that do not, i.e., $l^*(m_i) = 0$ (see Figure 1). We estimate λ in two ways.

1) Estimation Based on Observed Updates: For each webpage, we substituted t_{c_i} and t_{u_j} values into Eq. (5) or Eq.(4) and solved for λ using Brent's method to obtain its **estimated mean observed-update frequency** (λ). In this manner, we calculated λ for author websites at both homepage-level (using **D0** dataset) and webpage-level (using **D2** dataset).

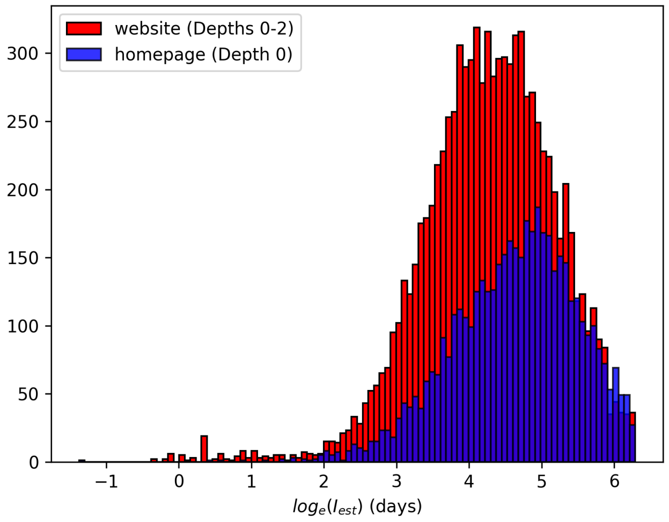
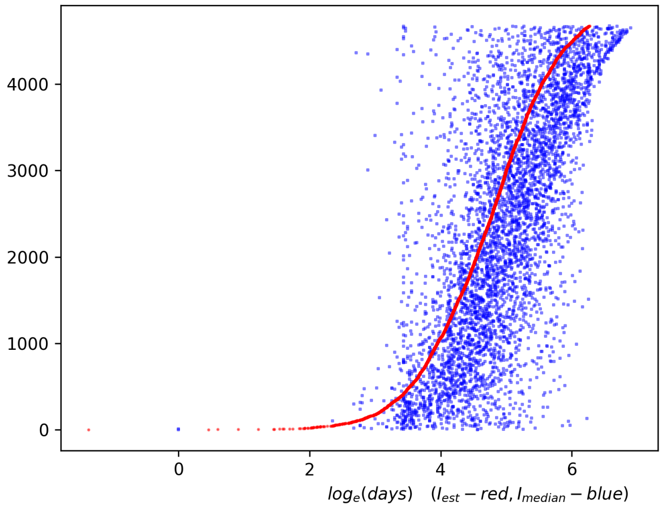


Fig. 4. Distribution of $1/\lambda$ of author websites at website-level (red) and homepage-level (blue). Here, λ was calculated using captures from 2015-06-01 to 2018-06-01.

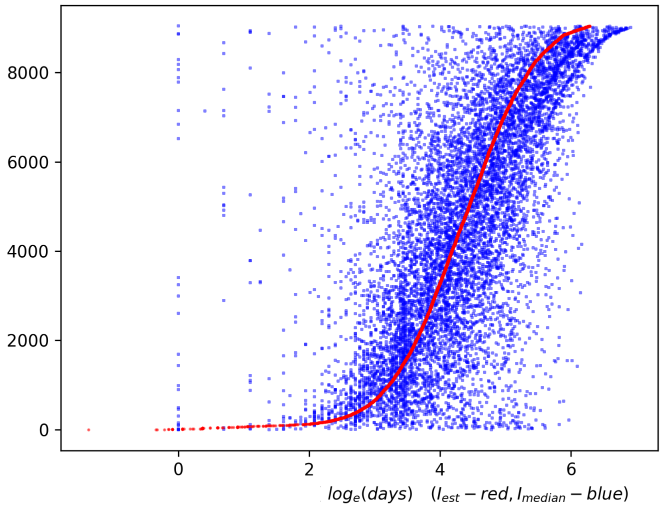
Figure 4 shows the distribution of $I_{\text{est}} = 1/\lambda$ at both website-level and homepage-level, obtained using captures from 2015-06-01 to 2018-06-01. Both distributions are approximately log-normal, with a median of 74 days at website-level, and of 110 days at homepage-level. This suggests that most authors add links to their homepage less often than they add links to their website (e.g., *publications*).

2) Estimation Based on Interpolated Updates: The method described in Section II-B1 calculates the maximum likelihood of observing the updates given by intervals t_{c_i} and t_{u_j}. Intuitively, an update could have occurred at any time between $t(m_{x-1})$ and $t(m_x)$, where $t(m_x)$ is the time of an updated capture, and $t(m_{x-1})$ is the time when the capture before it was taken. Here, we use an improved method where we first interpolate when a URL was updated. We define interpolated-update time (▲) as $(t(m_{x-1}) + t(m_x))/2$, i.e., the midpoint between $t(m_x)$ and $t(m_{x-1})$. Next, we obtain the update intervals $\tilde{t}_{c_i}$ and $\tilde{t}_{u_j}$ from these interpolated updates, and use them to calculate the **estimated mean interpolated-update frequency** ($\tilde{\lambda}$).

C. Distribution of Updates



(a) Homepage-level



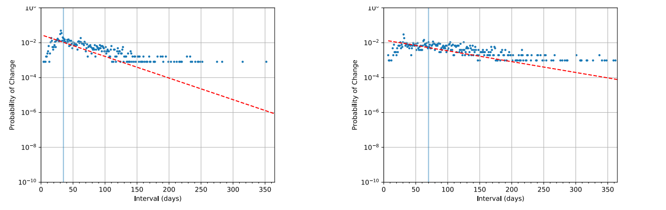
(b) Website-level

Fig. 5. The distributions of $1/\tilde{\lambda}$ (red) and the median *interpolated-update interval* ($\tilde{\Delta t}$) (blue) of author websites at (a) homepage-level and (b) website-level. The y-axis represents individual author websites, in the increasing order of $1/\tilde{\lambda}$.

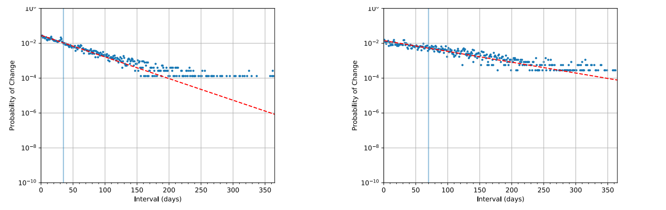
Figure 5 shows the distribution of $1/\tilde{\lambda}$ (red) and the median *interpolated-update interval* ($\tilde{\Delta t}$) (blue) of author websites at both homepage-level and website-level. It suggests that the distribution of $1/\tilde{\lambda}$ is consistent with the distribution of median $\tilde{\Delta t}$ at both homepage-level and website-level.

D. Poisson Distribution

Next, we observe whether updates to author websites follow a Poisson distribution, at both homepage-level and website-level. Here, we group author websites by their calculated $1/\tilde{\lambda}$ values into bins having a width of 1 day. Within each bin, we calculate the probability (y-axis) of finding an author website having an interpolated-update interval ($\tilde{\Delta t}$) of d days (x-axis).



(a) Homepage-level, $1/\tilde{\lambda} = 35$ days (b) Homepage-level, $1/\tilde{\lambda} = 70$ days



(c) Website-level, $1/\tilde{\lambda} = 35$ days (d) Website-level, $1/\tilde{\lambda} = 70$ days

Fig. 6. Probability (y-axis) of finding author websites with an interpolated-update interval ($\tilde{\Delta t}$) of d days (x-axis) at both homepage-level and website-level, among author websites having $1/\tilde{\lambda}$ of 35 days and 70 days, respectively. The vertical blue line shows where $d = 1/\tilde{\lambda}$.

Figure 6 shows the probability distributions for homepage-level (using **D0** dataset) and website-level (using **D2** dataset), at $1/\tilde{\lambda} = 35$ days and $1/\tilde{\lambda} = 70$ days, respectively. The majority of data points follow a power-law distribution in the logarithmic scale, indicating that they fit into a Poisson distribution. We also observe that at homepage-level, the data points follow a power-law distribution with a positive index when d is (approximately) lower than $1/\tilde{\lambda}$. We observe sporadic spikes on top of the power law. This indicates that: (1) For a given $\tilde{\lambda}$, consecutive changes within short intervals occur less frequently than predicted by a Poisson distribution, (2) The updates of scholarly webpages are not absolutely random but exhibit a certain level of weak correlation. Investigating the reasons behind these correlations is beyond the scope of this paper, but presumably, they may reflect collaboration or community-level activities.

E. Prediction Model

We formally define our prediction model using two functions, f and g. The function $f : m \rightarrow (\lambda, \tau)$ takes the captures m (i.e. crawl snapshots from the IA) of a website as input, and outputs its estimated mean update frequency λ (See Eq. (5)) and last known update time τ. The function $g : (\lambda, \tau, e) \rightarrow p$ takes a website's estimated mean update frequency (λ), its last known update time (τ), and a time interval (e) as input, and outputs the probability (p) that the website changes after the time interval e since its last known update time τ.

III. Evaluation

Here, we study how archived copies of webpages, and the quasi-Poisson distribution of webpage updates can be

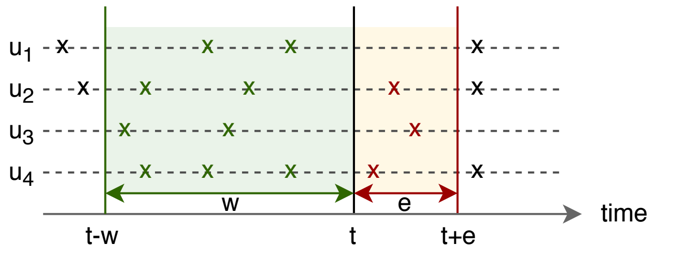


Fig. 7. An illustration of history size (w), reference point (t), evaluation interval (e), and updates ($\times$). For each URL u_i, λ was estimated using updates between $[t-w,t]$ (green), and the probability of change (p) at $t+e$ was calculated. In Evaluation 1, the correctness of p (red) was checked using the actual updates between $[t,t+e]$. In Evaluation 2, URLs were ordered by p, and compared against the ordering of those that changed first after t.

leveraged to build a focused crawl scheduler for the scholarly web.

Figure 7 illustrates our crawl scheduling model, HACS. For a selected date t between 2015-06-01 and 2018-06-01, we first obtain, from the **D2** and **D0**, archived captures of seed URLs within w weeks prior to t (i.e., in the interval $[t-w,t]$). Based on these captures, we calculate the **estimated mean interpolated-update frequency** ($\tilde{\lambda}$) of each seed URL. Next, we use the $\tilde{\lambda}$ values thus obtained, to calculate the probability (p) that each seed URL would exhibit a change e days from t (i.e., by day $t+e$). Following this, we sort the seed URLs in the decreasing order of p, and apply a threshold parameter (θ) to select a subset of seed URLs to be crawled on that date.

A. Simulated Crawl Scheduling Task

Here, we set $e=1$ week, and advance t across different points in time from 2015-06-01 to 2018-06-01, to simulate a crawl scheduling task. At each t, we use standard IR metrics to evaluate whether the selected subset of seed URLs were the ones that actually changed within the interval $[t,t+e]$. We also experiment with different values of w (i.e., history size), to determine which w yields an optimal result.

The following metrics are used for evaluating our model in comparison with several baseline models. First, we look at precision, recall, and F_1 to measure how accurately the scheduler selects URLs for a simulated crawl job (see Evaluation 1). Then, we use P@K to evaluate how accurate the scheduler ranks URLs in the order they change (see Evaluation 2).

B. Evaluation 1

Because most implementations of scholarly web crawlers are not published, we compare with two baseline models (1) random URLs (Random), and (2) Brute Force (select all URLs). We introduce a threshold parameter $\theta \in [0,1]$ to select webpages with a probability of change $p \geq \theta$ for crawling. Formally, we define the scheduling function as,

$$D_{w,t}(\theta) = \{u;\ g(\lambda,\tau,1) \geq \theta, (\lambda,\tau) = f(M_{w,t}(u)) \mid \forall u \in U\}$$

$$M_{w,t}(u) = \{m_x; x \in [t-w,t] \mid \forall m \in M_u\}$$

Here, U is the set of all seed URLs, and M_u is the set of captures of a seed URL u. The parameters w, t, and θ are the history size, reference point, and threshold, respectively. The functions f and g are as defined in Section II-E. For each (w,t,θ), the following actions are performed: In the HACS model, we use $D_{w,t}(\theta)$ to select URLs for crawling. In the Random model, we randomly pick $|D_{w,t}(\theta)|$ URLs from $D_{w,t}(0)$, i.e., all URLs having captures within the time window of $[t-w,t]$. In the Brute Force model, we mimic the behavior of a hypothetical crawler by picking all URLs from $D_{w,t}(0)$. The results from each model were compared to the URLs that *actually* changed within the interval $[t,t+e]$.

Following this, we counted the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) at each (w,t,θ). Next, we got rid of the reference point t by macro/micro-averaging over t, and calculated Precision (P), Recall (R), and F1 (F) for each w and θ, respectively. At each w, we then calculated the threshold $\theta=\hat{\theta}$ which maximizes $F1$ for both homepage-level and website-level. Table I shows the results from this evaluation.

We also show how P, R and $F1$ changes with $\theta \in [0,1]$ for both homepage-level and website-level updates. Figures 8,9, and 10 illustrate these results at $w=1$ and $w=2$.

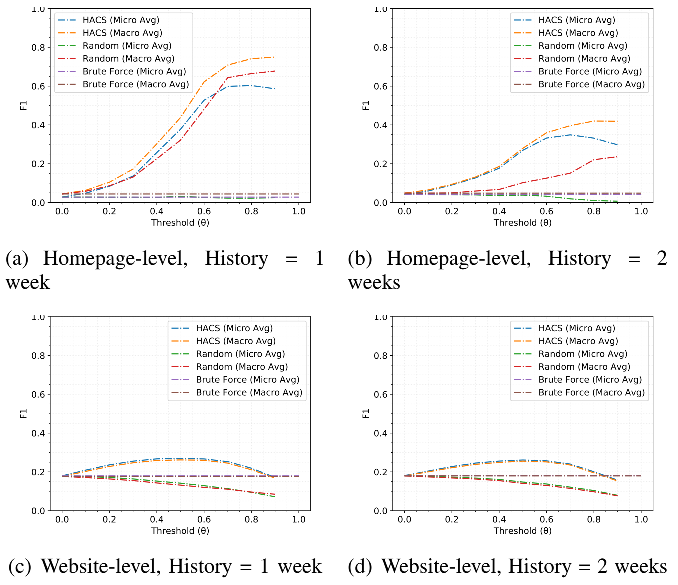


Fig. 8. F_1 vs Threshold (θ). The HACS model produced a higher F_1 than other baseline models. This lead is more visible at the homepage-level than the website-level. As θ increases, the F_1 of the HACS model increases up to $\theta=\hat{\theta}$, and then drops as θ further increases. This drop is more visible at the website-level than the homepage-level. The macro-average F_1 of Random model follows the HACS model with a similar trend at the Homepage-level, History = 1 week.

C. Evaluation 2

Here, the HACS model was compared against two baseline models: Last-Obs and Random. In the HACS model, URLs that have a higher probability of change on the crawl date $(t+e)$ are ranked higher. In the Last-Obs model, URL ranks are determined by the date they were last accessed. Here, URLs that have not been updated the longest (i.e. larger $(t-\tau)$) are ranked higher. In the Random model, URLs are ranked

TABLE I

COMPARISON OF HACS MODEL TO BASELINE MODELS USING PRECISION (P), RECALL (R) AND $F1$ VALUES AT $e = 1$ WEEK, AND AT THRESHOLD $\hat{\theta}$ WHERE $F1$ IS MAXIMUM. HERE, w IS THE HISTORY SIZE (IN WEEKS). MAXIMUM VALUES ARE IN BOLD, AND HIGHLIGHTED IN BLUE.

		Homepage-level							
		Micro Average				Macro Average			
	w	$\hat{\theta}$	HACS	Random	Brute	$\hat{\theta}$	HACS	Random	Brute
P	1	0.8	**0.759**	0.028	0.014	0.9	**0.919**	0.034	0.026
	2	0.7	**0.367**	0.020	0.021	0.8	**0.647**	0.018	0.026
	3	0.7	**0.267**	0.031	0.026	0.7	**0.305**	0.025	0.029
	4	0.6	**0.175**	0.046	0.037	0.7	**0.243**	0.038	0.039
	5	0.6	**0.178**	0.053	0.044	0.7	**0.220**	0.047	0.046
	6	0.6	**0.155**	0.047	0.044	0.7	**0.186**	0.045	0.045
	7	0.6	**0.134**	0.047	0.043	0.6	**0.136**	0.044	0.045
	8	0.6	**0.124**	0.046	0.043	0.6	**0.125**	0.044	0.045
	9	0.7	**0.134**	0.050	0.045	0.7	**0.139**	0.048	0.047
	10	0.7	**0.127**	0.047	0.045	0.7	**0.132**	0.046	0.047
	11	0.7	**0.121**	0.047	0.045	0.7	**0.125**	0.046	0.047
	12	0.7	**0.114**	0.050	0.045	0.7	**0.118**	0.050	0.046
R	1	0.8	0.500	0.019	**1.000**	0.9	0.556	0.026	**1.000**
	2	0.7	0.332	0.018	**1.000**	0.8	0.321	0.007	**1.000**
	3	0.7	0.291	0.033	**1.000**	0.7	0.346	0.025	**1.000**
	4	0.6	0.426	0.111	**1.000**	0.7	0.299	0.043	**1.000**
	5	0.6	0.445	0.133	**1.000**	0.7	0.322	0.070	**1.000**
	6	0.6	0.445	0.136	**1.000**	0.7	0.325	0.083	**1.000**
	7	0.6	0.448	0.156	**1.000**	0.6	0.459	0.147	**1.000**
	8	0.6	0.454	0.168	**1.000**	0.6	0.466	0.164	**1.000**
	9	0.7	0.335	0.125	**1.000**	0.7	0.342	0.122	**1.000**
	10	0.7	0.342	0.125	**1.000**	0.7	0.351	0.124	**1.000**
	11	0.7	0.348	0.134	**1.000**	0.7	0.356	0.134	**1.000**
	12	0.7	0.349	0.153	**1.000**	0.7	0.358	0.152	**1.000**
$F1$	1	0.8	**0.603**	0.022	0.028	0.9	**0.750**	0.678	0.044
	2	0.7	**0.349**	0.019	0.041	0.8	**0.420**	0.221	0.048
	3	0.7	**0.279**	0.032	0.051	0.7	**0.306**	0.087	0.055
	4	0.6	**0.248**	0.065	0.071	0.7	**0.255**	0.070	0.074
	5	0.6	**0.254**	0.076	0.084	0.7	**0.253**	0.071	0.087
	6	0.6	**0.230**	0.070	0.084	0.7	**0.228**	0.067	0.086
	7	0.6	**0.206**	0.072	0.083	0.6	**0.205**	0.073	0.086
	8	0.6	**0.194**	0.072	0.082	0.6	**0.194**	0.071	0.086
	9	0.7	**0.191**	0.071	0.086	0.7	**0.192**	0.072	0.090
	10	0.7	**0.186**	0.068	0.086	0.7	**0.187**	0.069	0.089
	11	0.7	**0.180**	0.069	0.086	0.7	**0.181**	0.067	0.089
	12	0.7	**0.172**	0.075	0.085	0.7	**0.173**	0.074	0.088

		Website-level							
		Micro Average				Macro Average			
	w	$\hat{\theta}$	HACS	Random	Brute	$\hat{\theta}$	HACS	Random	Brute
P	1	0.5	**0.195**	0.103	0.099	0.5	**0.191**	0.096	0.098
	2	0.5	**0.185**	0.104	0.099	0.5	**0.181**	0.099	0.099
	3	0.5	**0.164**	0.099	0.096	0.5	**0.162**	0.095	0.096
	4	0.5	**0.158**	0.097	0.096	0.5	**0.157**	0.094	0.096
	5	0.5	**0.150**	0.098	0.096	0.5	**0.150**	0.097	0.096
	6	0.5	**0.139**	0.094	0.092	0.5	**0.139**	0.093	0.093
	7	0.5	**0.130**	0.091	0.089	0.5	**0.131**	0.091	0.090
	8	0.5	**0.123**	0.089	0.087	0.5	**0.124**	0.089	0.088
	9	0.5	**0.118**	0.087	0.085	0.5	**0.120**	0.088	0.087
	10	0.5	**0.113**	0.084	0.084	0.5	**0.115**	0.085	0.085
	11	0.5	**0.108**	0.083	0.082	0.5	**0.110**	0.084	0.084
	12	0.5	**0.104**	0.081	0.080	0.5	**0.106**	0.082	0.082
R	1	0.5	0.435	0.230	**1.000**	0.5	0.444	0.228	**1.000**
	2	0.5	0.447	0.253	**1.000**	0.5	0.457	0.257	**1.000**
	3	0.5	0.460	0.277	**1.000**	0.5	0.471	0.280	**1.000**
	4	0.5	0.475	0.292	**1.000**	0.5	0.486	0.297	**1.000**
	5	0.5	0.482	0.315	**1.000**	0.5	0.493	0.320	**1.000**
	6	0.5	0.488	0.329	**1.000**	0.5	0.498	0.334	**1.000**
	7	0.5	0.492	0.345	**1.000**	0.5	0.502	0.349	**1.000**
	8	0.5	0.494	0.356	**1.000**	0.5	0.504	0.361	**1.000**
	9	0.5	0.497	0.367	**1.000**	0.5	0.507	0.373	**1.000**
	10	0.5	0.501	0.374	**1.000**	0.5	0.510	0.378	**1.000**
	11	0.5	0.505	0.385	**1.000**	0.5	0.512	0.389	**1.000**
	12	0.5	0.507	0.393	**1.000**	0.5	0.514	0.397	**1.000**
$F1$	1	0.5	**0.269**	0.142	0.180	0.5	**0.262**	0.132	0.177
	2	0.5	**0.261**	0.148	0.181	0.5	**0.256**	0.141	0.179
	3	0.5	**0.242**	0.145	0.175	0.5	**0.238**	0.140	0.174
	4	0.5	**0.237**	0.146	0.175	0.5	**0.234**	0.141	0.175
	5	0.5	**0.229**	0.150	0.175	0.5	**0.227**	0.146	0.175
	6	0.5	**0.216**	0.146	0.168	0.5	**0.215**	0.143	0.169
	7	0.5	**0.206**	0.144	0.163	0.5	**0.206**	0.143	0.164
	8	0.5	**0.197**	0.142	0.159	0.5	**0.197**	0.141	0.161
	9	0.5	**0.191**	0.141	0.157	0.5	**0.192**	0.140	0.159
	10	0.5	**0.184**	0.137	0.154	0.5	**0.186**	0.137	0.156
	11	0.5	**0.178**	0.136	0.151	0.5	**0.180**	0.136	0.154
	12	0.5	**0.173**	0.134	0.149	0.5	**0.175**	0.134	0.151

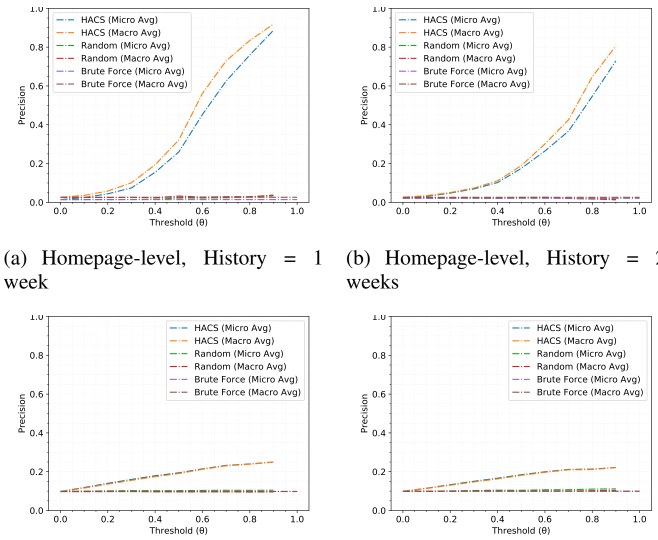
randomly. By comparing the URL rankings from each model to the *expected* URL ranking (where URLs that were updated closer to t were ranked higher), we calculate a weighted $P@K$ over all K. Here, the weights were obtained via a logarithmic decay function to increase the contribution from lower K values. This weighted $P@K$ provides a quantitative measure of whether URLs that were actually updated first were ranked higher. Next, we get rid of the reference point t by calculating the mean weighted $P@K$ over all t, at each history size w. In this manner, we obtain the mean weighted $P@K$ of each model when different history sizes (w) are used. Figure 11 shows the results from this evaluation.

IV. RESULTS

The results in Table I indicate that the P and F_1 values of HACS model are higher than the Random and Brute Force models for all values of w (history size in weeks). This lead is higher when w is lower. However, this difference becomes less significant as w increases. The Brute Force method had a consistent R of 1.00, since it crawls all URLs at all times. However, this model is impractical due to resource constraints. The HACS model produced a higher R than the Random model at all w. Also, $\hat{\theta} \in [0.7, 0.9]$ for homepage-level and $\hat{\theta} \in [0.5, 0.5]$ for website-level indicates the optimal ranges for θ.

From Figure 8, as θ increases, the F_1 score of HACS model increases until $\theta = \hat{\theta}$, and then drops as θ increases further. At $\hat{\theta}$, the HACS model yields the highest micro-average $F1$ score at both the homepage-level and the website-level. This trend is more prominent at the homepage-level than the website-level. In terms of macro-average $F1$, the Random model closely follows the HACS model at homepage-level when $w = 1$. However, the HACS model yields better $F1$ scores in all other cases. The Brute Force model gives constant $F1$ scores at both homepage-level and website-level, as it selects all seed URLs regardless of θ.

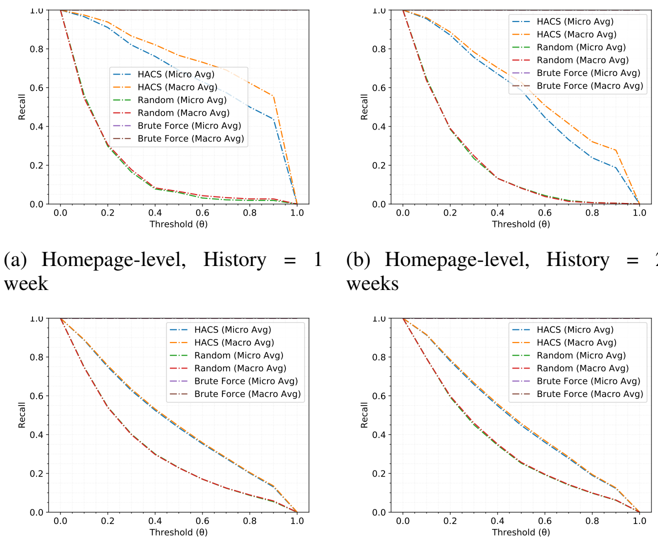
When comparing precision P, Figure 9 shows that both micro-average and macro-average P's of HACS model in-



(a) Homepage-level, History = 1 week (b) Homepage-level, History = 2 weeks

(c) Website-level, History = 1 week (d) Website-level, History = 2 weeks

Fig. 9. Precision (P) vs Threshold (θ). The HACS model produced a higher P than other baseline models, and increases with θ. This lead is more visible at homepage-level than website-level. Both Random and Brute Force models have a low P, regardless of θ.



(a) Homepage-level, History = 1 week (b) Homepage-level, History = 2 weeks

(c) Website-level, History = 1 week (d) Website-level, History = 2 weeks

Fig. 10. Recall (R) vs Threshold (θ). The HACS model produced a higher R at lower values of θ, and reaches 0 as θ increases. The HACS model has a much higher R than Random model at $0 < \theta < 1$. This lead is more visible at homepage-level than website-level. The Brute Force model has a constant R of 1.00.

creases as θ increases. This is expected as the URL selection becomes stricter as θ increases, which, in turn, generates less false positives. Similar to $F1$, the lead in P of the HACS model is more noticeable at homepage-level than website-level. Nevertheless, the HACS model yields higher P than other models in all cases. The Brute Force model has a constant P, as it selects all URLs regardless of θ. However, P of Brute Force model is lower than HACS model at both

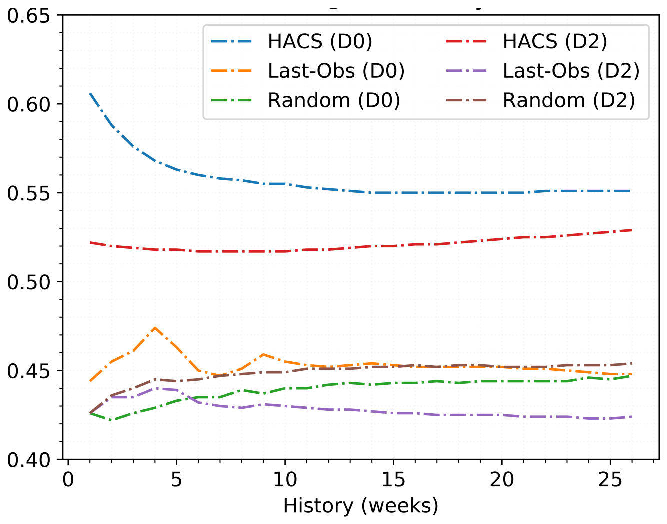


Fig. 11. Mean Weighted $P@K$ of rankings of HACS, Last-Obs, and Random models to the expected ranking, at different history (w) sizes. The HACS model outperforms the Last-Obs and Random models at both homepage-level and website-level.

homepage-level and website-level. Interestingly, the P of both Brute Force and Random models remain close to each other. At $\theta = 0.0$ (i.e. when no threshold is applied), all models give the same results, as they select all seed URLs.

When comparing results of R, Figure 10 shows that both micro-average R and macro-average R decreases as θ increases. This is expected as the URL selection becomes stricter as θ increases, which, in turn, generates less false negatives. The Brute Force model has a constant R of 1.00, as it selects all URLs regardless of θ. At $\theta = 0.0$ (i.e. when no threshold is applied), all models give $R = 1.00$ as they select all seed URLs. At $\theta = 1.0$, both HACS and Random models give $R = 0.00$, as they select no URLs here. For θ values other than these, the HACS model consistently yields better R than Random model at both homepage-level and website-level. However, this lead is less significant at website-level than at homepage-level, and diminishes as w increases.

When comparing the average P@K results, Figure 11 shows that the HACS model yields a better average P@K than the Last-Obs and Random models at both homepage-level and website-level, for all values of w. However, the HACS model yields a higher average P@K for lower values of w than for higher values of w. As w increases, the average P@K of all models become approximately constant. At homepage-level, the Last-Obs model yields a better average P@K than the Random model for lower values of w. At website-level, however, it yields a worse average P@K than the Random model for higher values of w.

V. Discussion

From Table I, the P, R, and $F1$ values obtained from the HACS model are greater than the baseline models at both the homepage-level and the website-level, when the optimal threshold $\hat{\theta}$ is selected. Figure 8 shows that regardless of the θ selected, the HACS model performs better than the baseline models. Also, the P of the HACS model increases as θ increases. This indicates that the HACS model predicted a higher probability (p) for the URLs that got updated first during $[t, t + e]$. This is also confirmed by the higher mean weighted $P@K$ values obtained by the HACS model (see Figure 11). Since R decreases with increasing θ while P increases with increasing θ, it is imperative that an optimal θ value should be selected. Results in Table I show that selecting $\theta = \hat{\theta}$ (which maximizes $F1$) provides a good compromise between precision and recall, yet perform better than the baseline models.

The P and R of the Brute Force model is constant irrespective of θ. Though this model yields the highest R (which is 1.00), it consumes a significant amount of resources to crawl everything. This approach does not scale well to a large number of seed URLs. It also yields a lower P and $F1$ than the HACS model across all w, at both homepage-level and website-level. These results suggest that the HACS model, which yields a much higher P and $F1$ at a marginal reduction in R, is more suited for a resource-constrained environment.

Recall that the archival of webpages is both irregular and sparse (See Figure 3). In our sample, authors updated their homepages every 141.5 days on average, and their websites every 75 days on average. Note that here, an update to a webpage means adding a new link into it. Authors may update their homepages or websites by updating content or adding external links. Content updates can be studied in a similar way by comparing the checksum of webpages. Since CDX files only contain mementos of webpages within the same domain, taking external links into consideration may require other data sources. The better performance of the HACS model in estimating the mean update frequency (λ) for homepages may be attributed to the fact that homepages undergo fewer changes than websites.

From Table I, the best micro-average $F1$ measure obtained at homepage-level and website-level were 0.603 and 0.269, respectively. Similarly, the best macro-average $F1$ measures obtained at homepage-level and website-level were 0.750 and 0.262, respectively. In both cases, these $F1$ measures originated from the HACS model when $w = 1$ and $\theta \in [0.5, 0.9]$.

Figure 8 demonstrates the efficiency of our model. As the threshold θ increases, the number of false positives is reduced, thereby increasing the precision. Here, we note that even a small increase in precision matters, because for a large number of seed URLs, even the slightest increase in precision attributes to a large decrease in false positives. If crawling is performed on a regular basis, the HACS model could be utilized to pick seed URLs that have most likely been updated. This, based on the above results, would improve collection freshness while using resources and bandwidth more effectively.

VI. Conclusion

We studied the problem of improving the efficiency of a focused crawl scheduler for the scholarly web. By analyzing the crawl history of seed URLs obtained from the IA, we fit their change information into a Poisson model and estimated the probability that a webpage would update (addition of new links) by the next crawl. Finally, our scheduler automatically generates a list of seed URLs most likely to have changed since the last crawl. Our analysis found that the estimated mean update frequency (or equivalently, update interval) follow a log-normal distribution. For the 19,977 authors we studied from Google Scholar, new links were added on an average interval of 141.5 days for a homepage, and 75 days for a website. We also observed that the median crawl interval of 80% of author homepages was between 20–127 days. Our evaluation results show that our scheduler achieved better results than the baseline models when θ is optimized. To encourage reproducible research, our research dataset consisting of HTML, CDX files, and evaluation results have been made publicly available[1].

In the future, we will investigate different types of updates, such as the addition of a scholarly publication in PDF format. Additionally, author websites could be crawled regularly to ensure that updates are not missed, and its effect on the estimation of mean update frequency could be evaluated. We will also generalize this work into more domains by exploring non-scholarly URLs.

VII. Acknowledgement

This work was supported in part by the National Science Foundation and the Dominion Graduate Scholarship from the College of Science at the Old Dominion University.

References

[1] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg, "Automatic resource compilation by analyzing hyperlink structure and associated text," *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 65–74, 1998.

[2] S. Chakrabarti, M. Van den Berg, and B. Dom, "Focused crawling: a new approach to topic-specific web resource discovery," *Computer networks*, vol. 31, no. 11, pp. 1623–1640, 1999.

[3] C. Olston and M. Najork, "Web crawling." *Foundations and Trends in Information Retrieval*, vol. 4, no. 3, pp. 175–246, 2010.

[4] B. Tofel, "Wayback'for accessing web archives," in *Proceedings of the 7th International Web Archiving Workshop*. IWAW'07, 2007, pp. 27–37.

[5] D. Gomes, J. a. Miranda, and M. Costa, "A survey on web archiving initiatives," in *Proceedings of Theory and Practice of Digital Libraries (TPDL)*, 2011, pp. 408–420.

[6] M. Farrell, E. McCain, M. Praetzellis, G. Thomas, and P. Walker, "Web archiving in the United States: a 2017 survey," https://ndsa.org/2018/12/12/announcing-publication-of-ndsa-s-2017-web-archiving-survey-report.html, 2018.

[7] H. Van de Sompel, M. L. Nelson, R. Sanderson, L. L. Balakireva, S. Ainsworth, and H. Shankar, "Memento: Time Travel for the Web," arXiv, Tech. Rep. arXiv:0911.1112, 2009.

[8] W. Koehler, "Web page change and persistence — a four-year longitudinal study," *Journal of the American Society for Information Science and Technology*, vol. 53, no. 2, pp. 162–171, 2002.

[9] J. Cho and H. Garcia-Molina, "Estimating frequency of change," *ACM Transactions on Internet Technology*, vol. 3, no. 3, pp. 256–290, Aug. 2003.

[10] K. Radinsky and P. Bennett, "Predicting content change on the web," in *WSDM '13: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, 2013.

[1]https://github.com/oduwsdl/scholarly-change-rate

[11] E. G. Coffman Jr., Z. Liu, and R. R. Weber, “Optimal robot scheduling for web search engines,” *Journal of Scheduling*, vol. 1, no. 1, pp. 15–29, 1998.

[12] C. Castillo, M. Marin, A. Rodriguez, and R. Baeza-Yates, “Scheduling algorithms for web crawling,” in *WebMedia and LA-Web, 2004. Proceedings*. Ribeirao Preto, Brazil: IEEE, Oct. 2004, pp. 10–17.

[13] P. De Bra, G.-J. Houben, Y. Kornatzky, and R. Post, “Information retrieval in distributed hypertexts,” in *Intelligent Multimedia Information Retrieval Systems and Management-Volume 1*. Centre de Hautes Etudes Internationales d’inform atique Docum entaire, 1994, pp. 481–491.

[14] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur, “The shark-search algorithm. an application: tailored web site mapping,” *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 317–326, 1998.

[15] F. Shipman and C. D. D. Monteiro, “Crawling and classification strategies for generating a multi-language corpus of sign language video,” in *19th ACM/IEEE Joint Conference on Digital Libraries, JCDL 2019*, 2019, pp. 97–106. [Online]. Available: https://doi.org/10.1109/JCDL.2019.00023

[16] G. Gossen, E. Demidova, and T. Risse, “icrawl: Improving the freshness of web collections by integrating social web and focused web crawling,” in *Proceedings of the 15th ACM/IEEE-CE Joint Conference on Digital Libraries*, 2015, pp. 75–84. [Online]. Available: https://doi.org/10.1145/2756406.2756925

[17] C. Olston and S. Pandey, “Recrawl scheduling based on information longevity,” in *Proceedings of the 17th International Conference on World Wide Web, WWW 2008*. ACM, 2008.

[18] J. Cho, H. Garcia-Molina, and L. Page, “Efficient crawling through url ordering,” *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 161–172, 1998.

[19] M. H. Alam, J. Ha, and S. Lee, “Novel approaches to crawling important pages early,” *Knowledge and Information Systems*, vol. 33, no. 3, pp. 707–734, 2012.

[20] J. Wu, P. Teregowda, J. P. F. Ramírez, P. Mitra, S. Zheng, and C. L. Giles, “The evolution of a crawling strategy for an academic document search engine: Whitelists and blacklists,” in *Proceedings of the 4th Annual ACM Web Science Conference*, ser. WebSci ’12, 2012, pp. 340–343.

[21] S. Zheng, P. Dmitriev, and C. L. Giles, “Graph-based seed selection for web-scale crawlers,” in *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009*. New York, NY, USA: ACM, 2009, pp. 1967–1970.

[22] D. S. Hochbaum and A. Pathria, “Analysis of the greedy approach in problems of maximum k-coverage,” *Naval Research Logistics (NRL)*, vol. 45, no. 6, pp. 615–627, 1998.

[23] M. Thelwall and D. Wilkinson, “Graph structure in three national academic webs: Power laws with anomalies,” *Journal of the American Society for Information Science and Technology*, vol. 54, no. 8, pp. 706–712, 2003.

[24] C. L. Giles, K. D. Bollacker, and S. Lawrence, “CiteSeer: An automatic citation indexing system,” in *Proceedings of the 3rd ACM International Conference on Digital Libraries, June 23-26, 1998, Pittsburgh, PA, USA*, 1998, pp. 89–98. [Online]. Available: http://doi.acm.org/10.1145/276675.276685

[25] J. Wu, K. Kim, and C. L. Giles, “Citeseerx: 20 years of service to scholarly big data,” in *Proceedings of the Conference on Artificial Intelligence for Data Discovery and Reuse*, ser. AIDR ’19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3359115.3359119

[26] J. Cho and H. Garcia-Molina, “The evolution of the web and implications for an incremental crawler,” in *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases*, 2000, pp. 200–209.

[27] ——, “Synchronizing a database to improve freshness,” in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000*, Dallas, Texas, USA., 2000, pp. 117–128.

[28] R. P. Brent, “An algorithm with guaranteed convergence for finding a zero of a function,” *The Computer Journal*, vol. 14, no. 4, pp. 422–425, 1971.

[29] Internet Archive, “Wayback CDX Server API,” https://github.com/internetarchive/wayback/tree/master/wayback-cdx-server, 2019.