DSDD: Domain-Specific Dataset Discovery on the Web

Haoxiang Zhang
New York University
haoxiang.zhang@nyu.edu

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

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

ABSTRACT

With the push for transparency and open data, many datasets and data repositories are becoming available on the Web. This opens new opportunities for data-driven exploration, from empowering analysts to answer new questions and obtain insights to improving predictive models through data augmentation. But as datasets are spread over a plethora of Web sites, finding data that are relevant for a given task is difficult. In this paper, we take a first step towards the construction of domain-specific data lakes. We propose an end-to-end dataset discovery system, targeted at domain experts, which given a small set of keywords, automatically finds potentially relevant datasets on the Web. The system makes use of search engines to hop across Web sites, uses online learning to incrementally build a model to recognize sites that contain datasets, utilizes a set of discovery actions to broaden the search, and applies a multi-armed bandit based algorithm to balance the trade-offs of different discovery actions. We report the results of an extensive experimental evaluation over multiple domains, and demonstrate that our strategy is effective and outperforms state-of-the-art content discovery methods.

CCS CONCEPTS

• **Information systems** → **Web searching and information discovery**.

KEYWORDS

Focused Crawling, Domain-Specific Dataset Discovery, Meta Search, Online Learning, Multi-Armed Bandit

ACM Reference Format:
Haoxiang Zhang, Aécio Santos, and Juliana Freire. 2021. DSDD: Domain-Specific Dataset Discovery on the Web. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482427

1 INTRODUCTION

The increasing availability of datasets on the Web opens new opportunities for data-driven exploration, from deriving insights from data to improving predictive models through data augmentation. Consider the following example.

Explaining Food Insecurity in Ethiopia. Despite the strong economic gains over the past two decades, food insecurity persists in Ethiopia. According to the United Nations World Food Programme, an estimated 8 million people require food assistance and about 3.9 million women and children are nutritionally vulnerable in Ethiopia [45]. A group of data scientists aims to construct predictive models to explore potential causes and triggers for food insecurity. The Ethiopian government shared with them data about population, health, and income. But on the Web, there are many additional datasets, including, for example, indicator data provided by the World Bank and the Food and Agriculture Organization, which cover different aspects of life in Ethiopia – from weather and droughts to conflicts. By collecting and analyzing relevant datasets, domain experts can gain additional insights and discover features to construct predictive models that are robust and accurate. These modes, in turn, can help inform interventions to avoid food insecurity and mitigate its effects.

However, finding relevant datasets on the Web is challenging. There is a large number of datasets spread across many different data repositories and Web sites. As a point of reference, if we consider urban data, there are over 200 repositories powered by Socrata [56] containing many thousands of datasets (e.g., New York City Open Data [16], Chicago Data Portal [39]). Recognizing this challenge, Google took a first step towards improving findability with Google Dataset Search (GDS), a search engine for datasets [10]. However, to have datasets indexed by GDS, data publishers must include metadata using standard formats such as `schema.org` or W3C's DCAT [10]. Unfortunately, relying on metadata markup leads not only to a limited coverage but also to a veracity problem: up to 61% of web sites that provide `schema.org/Dataset` markup on their web pages do not actually describe datasets [4]. Therefore, an open question remains for domain experts: how to efficiently discover *all* data relevant to a given domain of study or research question, including datasets not indexed by Google Dataset Search.

One possible approach is to use a focused crawler [8, 12, 15, 18, 20, 35, 54, 57]. Instead of attempting to cover all Web pages, a focused crawler searches for a target concept (or topic) while maximizing the number of on-topic pages it retrieves and minimizing the number of irrelevant pages visited. Focused crawlers thus bring many benefits: they lead to substantial savings in hardware and network resources compared to searching the whole Web; they make it cheaper to maintain the crawl up-to-date; crawls can go deeper and obtain a better coverage for a given topic; and the derived index, being focused, is more likely to return a higher fraction of actually relevant pages, reducing the information overload for the end user. Focused crawlers have been proven effective to acquire topic-specific Web pages as well as to discover specific objects inside Web pages, such as events [24, 28] and product information [22, 49]. They leverage the page contents to determine relevance to a topic and to harvest links that are used to expand the search. But crawling for datasets on the Web creates new challenges. Datasets are often published in repositories and do not link to other datasets or repositories. Therefore, we need new crawling strategies to find regions of the Web that contain domain-specific datasets.

Another challenge comes from the complexity involved in configuring focused crawlers. To specify a topic, an expert must provide both a set of seed URLs that serve as the starting points for the crawl and a *classifier* to automatically recognize relevant content. While finding seeds can be time consuming, the task of creating an effective classifier is difficult and often out of reach for experts without training in computing.

DSDD: A Usable Dataset Discovery Framework. In this paper, we propose DSDD, an end-to-end system for domain-specific dataset discovery on the Web. Because datasets are distributed over disconnected regions of the Web that cannot be reached by crawling alone, we need mechanisms to *jump* to pages that serve as entry points to dataset repositories. To address this challenge, our system tries to mimic the manual and iterative process users follow to discover relevant datasets: issue queries (e.g., *"Ethiopia dataset"*) to a search engine, visit the returned URLs, determine their relevance, and refine the search (e.g., search for links *related* to one of the relevant results). Our system automates this process and utilizes a rich set of discovery actions to *jump-crawl* the Web, including keyword and related search, supported by search engines, and forward and backward crawling [13]. Since these actions have an associated cost, including rate limits posed search engines, it is important to maximize their benefit, i.e., the number of datasets they discover. However, selecting the appropriate discovery actions is difficult because the effectiveness of an action depends on multiple factors, such as the domain of interest, seed pages, and keyword queries. We propose an algorithm based on multi-armed bandits to dynamically schedule the actions with the goal of attaining a balance between the exploitation of productive regions and the exploration of new regions that can increase the coverage and diversity of the discovered datasets.

DSDD was inspired by DISCO [43], an approach to bootstrap domain-specific search. Given a small set of seed web sites, DISCO automatically expands the seed set. Similar to DISCO, our approach employs discovery strategies including keyword and related search, backward and forward crawling, and adapts a multi-arm bandit based algorithm to balance the exploitation and exploration of different strategies. But different from DISCO, DSDD uses a classifier to narrow down the set of candidates, automatically verifies these candidates, and uses the verified results to update the classifier as well as to extend the seeds set. This leads to significant improvements as we discuss in Section 5.

DSDD attains *usability* by 1) requiring users to provide just a small set of descriptive keywords for the domain of interest; and 2) applying online learning to incrementally and automatically improve the classifier that determines which pages are potential dataset repositories. This classifier is bootstrapped with manually collected samples of open data portals, which provide an initial set of features for pages that represent a dataset repository. At each discovery iteration, the classifier identifies potential dataset repositories, which are then verified by DSDD, i.e., the system checks whether the identified repository actually leads to dataset pages. The results of the verification are then used to automatically update and improve the classifier. Unlike Web pages and objects inlined in pages, datasets can be large. Therefore, it may not be feasible to download and parse their contents at crawling time to determine relevance. Even if the data are downloaded, determining their relevance and quality is challenging [41, 44, 58, 60]. Therefore, our goal is to maximize the number of dataset pages (and repositories) we discover for a given domain, so that a rich data collection can be assembled. After the datasets are collected, profiled and indexed by a dataset search engine such as Auctus [11] or Find Open Data [62], users will be able to pose a rich set of discovery queries [61] to identify which datasets are relevant for different tasks (e.g., improve a machine learning model, augment an existing dataset). Information about datasets found to be relevant could then be used to refine the discovery process of DSDD.

Contributions. To the best of our knowledge, this paper proposes the first system for automatic domain-specific discovery of datasets on the Web that does not rely on web page annotations. Our main contributions can be summarized as follows:

- We propose an end-to-end, usable framework to automatically and efficiently discover domain-specific datasets (Section 4).
- We adapt a multi-armed bandit-based algorithm to balance the exploitation and the exploration of discovery actions, and combine it with online classification to attain both usability and efficiency. We show that this combination increases the harvest rate for the discovery process (Section 4).
- We perform an extensive experimental evaluation using multiple domains, and present results which show that not only does our approach lead to harvest rates that are nearly 100% or higher compared to state-of-the-art domain-specific content discovery systems, but it also finds dataset pages that are not indexed by Google Dataset Search (Section 5).

2 RELATED WORK

Our work builds on large body of research from information retrieval on web crawling and document ranking, and from machine learning on supervised classification and reinforcement learning algorithms. In what follows, we review some of the relevant related work in these areas.

2.1 Web and Focused Crawling

A general-purpose web crawler automatically collects information from (all) web pages it reaches to be indexed by search engines. A focused crawler, in contrast, aims to effectively build high-quality collections of content relevant to specific topics [15]. It achieves this objective by focusing the crawl on links that are most relevant to the target topic and avoiding irrelevant ones. For example, Chakrabarti et al. [14] leverages two classifiers to accelerate crawling: a baseline classifier trained on a topic taxonomy, and an online classifier that continually learns from crawled data to improve the classification precision. Focused crawling methods with online learning strategies have been proposed to locate hidden-web entry points [8] and find structured data [36]. A key feature that distinguishes crawling to discover datasets from other topics considered in the focused crawling literature is that datasets on the Web are not highly connected. As we show in our experiments (Section 5), traditional focused crawlers are not effective in this scenario and attain low harvest rates. Our approach addresses this limitation by leveraging search engines to discover crawling seeds and an online classifier that improves future decisions by continuously learning from verified sites.

2.2 Web Page Classification

A web page classifier is essential for focused crawlers to evaluate the relevance of a web page and narrow down the crawling boundary [48]. Different types of features have been used for web page classification. URL-based features are used to classify web pages in a fast way as they do not need to access the contents of the web page [1, 27, 50]. Incorporating HyperText Markup Language (HTML) tag or HTML structure features demonstrates the improved classification accuracy [25, 53, 59]. In addition to features on the web page to be classified, features can be utilized from its neighbors, such as parent, sibling and child pages [17, 46, 47, 55]. In this work, we focus on the features on the web page to be classified. Because of the sparse distribution of relevant datasets on the Web, it is inefficient, in terms of accuracy and time, to utilize features of neighboring pages.

2.3 Exploration and Exploitation

The relation between the exploration of new opportunities and the exploitation of existing certainties was first considered in organizational learning problems [34]. In web crawling and web content discovery, exploration often refers to the search for new relevant pages from resources that are several steps away from the relevant subset, and exploitation refers to the search of relevant pages from resources known to be relevant [40].

The idea of exploitation and exploration has also been used in the context of re-crawling for domain-specific content discovery, where web pages from different domains can have very different change rates over time [52]. Pham et. al. [42] proposed a method to balance between the exploitation of re-crawling policies that yield high discovery rates with the exploration of policies that may yield low discovery rates in isolation but ultimately improve the overall discovery rates over time when combined with policies that are known to be good.

In our approach, we make use of several actions to find new Web sites to jump to: forward links, backward links, related pages and keyword queries. These techniques have been shown to be effective for content discovery in different domains [13, 19, 20, 23, 37, 42, 43, 49]. Since the effectiveness of these strategies varies for different domains and sites, we propose a multi-armed bandit-based algorithm to balance the trade-offs of the exploitation of high-rewarding discovery actions, and the exploration of actions that have not demonstrated good performance in the past, but may bring future rewards.

3 PROBLEM DEFINITION

Before presenting our approach, we first introduce concepts needed to formally define the problem we address in this paper.

Definition 1. ***Dataset Page (DP)***. *A dataset page pd is a web page that contains at least one link to a dataset.*

We consider dataset files that are typically used to store structured data including, for example, CSV, TSV, JSON, XLS, XML, shapefiles for spatial data, and file collections (e.g., ZIP and GZ).

Definition 2. ***Dataset Repository Entry Page (DREP)***. *A web page pr is a dataset repository entry page, if pr links directly or indirectly (within k steps) to a non-empty set of dataset pages $S = \{pd_1, pd_2, ..., pd_n | \ n \geq 1\}$.*

Note that this definition captures pages that serve as entry points to data portals (e.g., NASA Open Data Portal [38] and World Bank Open Data [6]), as well as other web pages from which datasets can be discovered. For example, a researcher's homepage that contains links to dataset pages. Our system uses $k = 3$ when checking if a web page is a DREP. We choose this value based on our observation that, in most open data portals we crawled, dataset pages can be discovered within three steps away from the entry page. Given these definitions, we can define the problem addressed in this paper:

Definition 3. ***Domain-Specific Dataset Discovery***. *Given a domain of interest, we aim to efficiently discover dataset repository entry pages and dataset pages that are related to the domain.*

4 DATASET DISCOVERY FRAMEWORK

In this section, we first provide an overview of our solution, and then describe our system components in detail.

4.1 Solution Overview

We propose DSDD, a new end-to-end dataset discovery system. Figure 1 shows the architecture of the system. A user starts by providing a set of keywords that are descriptive of the domain of interest. The system bootstraps the process by applying a set of discovery actions, including keyword queries and related queries using a search engine, and forward and backward crawling. Given that the effectiveness of these actions varies for different domains and sites, we developed an algorithm based on the multi-armed bandits approach to balance the exploitation of high-reward discovery actions and the exploration of actions that may lead to delayed benefits (**Discover**, Sections 4.2 and 4.3, Algorithm 1 lines 5-9).

As new pages are discovered, to ascertain whether a page is a DP (Definition 1), DSDD searches for links in the page that point to a individual dataset (e.g., CSV, TSV, JSON, XLS, XML, shapefiles for spatial data) or a collection of datasets (e.g., ZIP and GZ). Checking whether a page pr is a DREP (Definition 2) can be more expensive, as

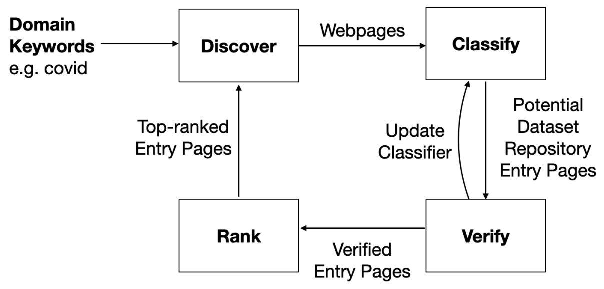


Figure 1: The architecture of our dataset discovery system.

Algorithm 1 Dataset Discovery Framework

```
1:  procedure DATASET_DISCOVERY(domain)
2:      data_repos = ∅
3:      analyzed_pages = 0
4:      while analyzed_pages< $k$ do
5:          if first iteration then
6:              webpages = $keyword\_search$ (domain)
7:          else
8:              act = $select\_discovery\_action$()
9:              webpages = $discover$(act, top_repos)
10:         potential_repos= $classify$ (webpages)
11:         verified_repos, n_pages = $verify$ (potential_repos)
12:         analyzed_pages = analyzed_pages + n_pages
13:         $update\_classifier$(verified_repos)
14:         data_repos = data_repos ∪ verified_repos
15:         top_repos = $rank$(data_repos)
16:     dataset_pages = $crawl$ (data_repos)
17:     return dataset_pages
```

this requires a recursive crawl to search for DPs that can be reached from pr. Performing this check for a large number of candidate pages can be prohibitively expensive, thus greatly reducing the discovery harvest rate. Instead, we use a classifier to predict whether a page is a repository entry page (**Classify**, Sections 4.4, Algorithm 1 line 10). This classifier is pre-trained with positive and negative examples collected for general open-data portals.

Pages that are classified as DREPs are then verified (**Verify**, Section 4.5, Algorithm 1 lines 11-13) to check whether they actually lead to DPs. Using the outcome of the verification, we can apply online learning and automatically improve the classifier at each discovery iteration.

To bootstrap a new discovery iteration, we need a set of seed pages. While all verified DREPs could be used as seeds, in practice, using such a large set can be expensive. In addition, since search engine APIs have a rate limit on requests, ideally, we should focus on the *best* pages. As we discuss in Section 4.6, we introduce a ranking function and select the top-k seeds (**Rank**, Algorithm 1 lines 14-15).

4.2 Discovery Actions

The goal of the **Discovery** component is to find new dataset repository entry pages (DREPs). We obtain links from entry pages collected from dataset repositories, such as sibling pages, child pages, related pages, and pages returned by keyword queries issued to search engines using keywords extracted from the pages. In what follows, we provide detailed descriptions of the discovery actions used to obtain new pages.

4.2.1 Following Backward Links. For a given web page P, a backward link is a link in another web page that points to P. The process to find relevant pages from backward links is similar to that of finding relevant papers through citations. Papers that are related are usually cited together. Similarly, related web pages are frequently referenced together [30]. As a result, searching backward links for dataset repositories is likely lead to hubs with links to other repositories. We leverage search engine APIs to obtain backward links from the selected web pages. From the retrieved backward links, we collect their outlinks to discover additional dataset repository pages.

4.2.2 Following Forward Links. Given web page, forward links are the outlinks inside the page. For a web page P that has been verified as a dataset repository page, to focus the search on new sites, we consider only the forward links that have a domain name that is different from the domain of P.

4.2.3 Related Pages. The related pages of a given web page P are the pages returned by a search engine for the query $related : P$. In practice, we have observed that many of the related pages cannot be found by following backward and forward links. For example, given the URL *https://data.ny.gov*, one of the results from the query *related:https://data.ny.gov* is *https://data.-austintexas.gov* which cannot be discovered through either backward or forward search. Therefore, searching for related pages helps enrich our discovery range.

4.2.4 Keyword Search. The keyword search obtains relevant web pages by submitting keyword queries to search engine APIs. For each page retrieved, we extract textual contents from the title, links and the body. Then, we tokenize the extracted text and remove stop words from the stop word list curated from pre-collected sample web pages. We select the most frequent keywords to build keyword queries. Furthermore, we concatenate the domain given by the user and the data keywords to construct more precise keyword queries. For example, *cases* and *hospitalized* are two terms extracted from repository pages for the *COVID* dataset, but issuing queries with these keywords individually rarely returns dataset repositories, let alone COVID dataset repositories. After concatenating the selected terms with *COVID* domain and the term *dataset*, the resulting keyword queries *COVID cases dataset* and *COVID hospitalized dataset* definitely return more web pages that are related to COVID dataset repositories.

4.3 Bandit-Based Exploration and Exploitation

At each discovery iteration (Algorithm 1, line 8), DSDD must select a discovery action to find new candidate pages. An important challenge is how select the action that maximizes the harvest rate and coverage. If we stick to the same action when it shows good performance, we can lose the opportunity to diversify the candidates and, eventually, this may lead to a decrease the number of new dataset repository pages the system can discover. This is an instance of the dilemma between the exploitation of resources that are known to be promising and the exploration of uncertain resources.

We use an approach based on multi-armed bandits, specifically UCB1 algorithm [5], to address this challenge. Each discovery action is an individual bandit arm. The goal is to select the arm considering both its accumulated reward and future potential. The UCB1 algorithm uses an upper confidence bound to measure this future potential. The algorithm can be expressed as the following equation:

$$score_t(a) = \overline{r(a)} + \sqrt{\frac{2ln(n)}{n_t(a)}} \qquad (1)$$

where $score_t(a)$ is the score of the bandit arm (a) at time t, $\overline{r(a)}$ is the average accumulated reward of discovery action a, $n_t(a)$ is the number of web pages bandit arm a has discovered at time t, and n is the total number of collected web pages by all discovery strategies. Intuitive, an action with low accumulated reward will be explored at time t if it gains a large amount of future potential.

Next, we define the reward of a discovery action $r(a)$. If a web page p has been discovered, based on its features, the dataset repository page classifier will label it either as relevant or not. As a result, the reward of p is defined as:

$$R_p = \begin{cases} 1 & \text{if } p \text{ is relevant} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

If that a discovery action a returns m web pages, its reward is:

$$r(a) = \frac{\sum_{p=1}^{m} R_p}{m} \tag{3}$$

At each iteration, we calculate the score of each discovery action based on these equations and select the one with the highest score.

4.4 Classifying Dataset Repository Entry Pages

As new pages are discovered, we could inspect them to determine which are dataset repository entry pages. But doing this is expensive, as we need to crawl the links in the page and subsequent pages. Instead, we make use of a classifier to predict whether a page is a repository entry page.

We treat the web page classification problem as a binary document classification problem, and assign a positive or a negative label to a web document. Web pages that are dataset repository entry pages are labeled as positive, and all other pages are labeled as negative. The classifier is pre-trained with manually collected positive and negative samples of general open-data portals. These samples serve as a starting point to help the classifier obtain features that are representative of dataset repository entry pages. Then, for different domains being crawled, we apply online learning to incrementally improve this classifier by using discovered results to update it. This online learning approach helps the classifier gradually capture domain-specific features of dataset repository pages.

Web Page Representation. Features in web page classification are usually divided into two parts: features located on the page to be classified, and features located on the neighbors of the page to be classified, such as its child pages or parent pages [48]. In this work, we only use features located on the page, as fetching features from neighboring pages is expensive, and as our experimental results show, this strategy is effective in practice. Specifically, we use the textual content in the page, which includes title text, link text, and body text. Our pre-processing pipeline consists of tokenization, uniformly transforming the text to lower case, and removal of non-alphabet terms and stop words. The pre-processed Web documents are transformed into numeric vectors using TFIDF, which is a numerical statistic that assigns weights to terms according to their importance. TFIDF aims to identify terms in a collection of documents that are useful to determine the category of a document [32]. Among the various existing term-frequency variants, we choose the augmented term frequency to mitigate the bias towards longer documents [33]:

$$tf(t,d) = 0.5 + 0.5 \cdot \frac{f(t,d)}{max_{t' \in d} f(t',d)} \tag{4}$$

where $f(t,d)$ is the count of the term t in the document d.

Classification Model. The TFIDF feature vectors created for the web pages are used as input for the classification model that decides which of these web pages are potential dataset repository entry page. We employ support vector machine (SVM) as the learning method, since it is effective for sparse features and performs competitively with state-of-art classifiers [33]. For the SVM model, we use a linear kernel and set the regularization parameter to 1.0.

4.5 Verifying Dataset Repository Entry Pages

To obtain more accurate results and improve the dataset repository entry page classifier with online learning, we need to verify which entry pages predicted by classifier link, directly or indirectly, to datasets. But crawling all subsequent pages from a given candidate entry page is both costly and can be infeasible in practice. Instead, we preform a partial crawl: starting from the given data repository entry point, we restrict the depth the crawl to three, i.e., the crawler searches pages within three hops of the starting page. We observed that, in most dataset repositories, dataset pages can be reached within three steps away from the entry page. During this in-site crawl, for each page, we use regular expressions to check for URLs that link to common dataset formats, that include but are not limited to individual data files (e.g., CSV, TSV, JSON, XLS, XML, shapefiles for spatial data) and collections of data files (e.g., ZIP and GZ).

Some dataset repositories contain large number of datasets, which can occupy most of the frontier resources of the crawler. This situation decreases the efficiency of the verification process, because under a fixed amount of resources, fewer other potential dataset repositories can be verified. Since our purpose in this step is to quickly check which repository entry pages contain datasets, we stop crawling the links from an entry page if a reasonable number of dataset pages have been identified. In practice, we observed that when few dataset pages were discovered from the entry page, the files were often not data files (e.g., a CSV file containing a description of the website being crawled). Therefore, DSDD has a threshold for the number of dataset files that are required to consider a given link as a DREP. For the domains we considered, we observed that a threshold of 10 was effective.

4.6 Selecting Seeds via Ranking

Since our end-to-end system is iterative, seed pages are needed for each round of discovery. Using all verified entry pages as seeds is expensive, and it may also exhaust the rate allowed by the search engine API. In addition, it can hurt performance as less relevant seeds can lead to fewer new DREPs.

To address this problem, we rank dataset repository entry pages by an automatically generated query, and select the top-ranked ones as seeds for next discovery iteration. This query is built on frequent keywords extracted from domain-specific pages. For example, in the *COVID* domain, the query can be *COVID hospitalized cases deaths data* where *hospitalized*, *cases*, and *deaths* are top terms extracted from the dataset repository pages we collected. We introduce the ranking function we use below.

Ranking is a fundamental problem in information retrieval: search systems rank a collection of documents, based on some criterion, with respect to a given query. In our problem setting, documents are the contents of dataset repository entry pages that have been verified (see Section 4.5), and the query is the one we built from these entry pages. We used the BM25 [51] method as the ranking function, which is a variation of the *vector-space model* based on the probabilistic framework that shows good performance in multiple document ranking tasks [9, 51].

5 EXPERIMENTAL EVALUATION

We evaluate the efficiency of our end-to-end system through a detailed experimental evaluation. We examine the *recall* of different discovery configurations and manually assess the retrieved results. For each experiment we carried out, we introduce the evaluation metrics and then present and discuss the results. We selected a diverse set of *social good* domains to evaluate our system:

- *Coronavirus (COVID)*: datasets relevant to study this infectious disease caused by severe acute respiratory syndrome coronavirus. These include, for example, daily new cases, quarantine policies, and traffic flow changes.
- *Ethiopia food security (Ethiopia)*: datasets helpful to analyze the reasons behind the food insecurity and design corresponding actions to mitigate its effects, such as Ethiopia weather and energy datasets.
- *Armed conflict (Conflict)*: datasets useful to analyze the factors contributing to violent conflicts and predict future possible collisions, such as economic conditions and degradation of natural resources like water scarcity.
- *Climate change (Climate)*: datasets relevant to study climate changes and shifts in weather patterns around the world, such as greenhouse gas emissions and El Niño–Southern Oscillation.
- *Equity in education (Education)*: datasets helpful to analyze current education situation around the world, for example, school enrollment rates of different genders and income levels.
- *Economic development (Economics)*: datasets useful to analyze and predict economic growth, such as macroeconomic statistics and international trade statistics.

5.1 End-to-End System Performance

We evaluate the performance of our approach for domain-specific dataset discovery, and compare it with other state-of-the-art domain-specific content discovery systems. We select systems to compare based on following criteria: the system (1) has similar, comparable functionality (e.g., topic-specific crawlers), (2) is open-source, and (3) is actively maintained or at least can be built successfully. Based on these criteria, we selected the following two systems: ACHE [8], focused crawler, and DISCO [43], a domain-specific content discovery system.

To start the search, both systems require a list of seed web pages that are relevant to the topic. For a fair comparison, we use the verified results from our first iteration (keyword search) as the seeds for both systems. We describe baseline methods and our methods below. For each discovery method, we stop it once 50,000 pages are analyzed. For conciseness, we use the following abbreviations for the method names: BD stands for bandit, OL for online learning, KW for keyword, RL for related, BL for backlink and FW for forward.

ACHE: ACHE is an open-source *focused web crawler* [2, 8]. Given a set of seed pages, it aims to maximize the number of relevant pages discovered by prioritizing the unvisited links. We set the number of crawled web pages per domain to 100 so that ACHE visit can visit a larger number of different Web sites. ACHE is effective at finding pages belonging to a given domain; and it does so by leveraging the network topology of specific domains on the Web. However, it lacks the ability to obtain relevant pages that are not connected to pages it has visited – a feature that is important to discover datasets, which are often disconnected.

DISCO: DISCO is an approach to bootstrap *domain-specific content discovery* [21, 43]. It uses a rank-based framework to mimic the way users search on the Web, and combine search-based and crawling-based discovery operations. We use the DISCO bandit search setting which uses a multi-armed bandit algorithm to select a search operator at each iteration among backlink search, keyword search, related search and forward search. For the ranking function, we chose Bayesian sets as it shows good performance and does not require negative samples.

DSDD-BD: this is our system that includes a pre-trained classifier (with online learning disabled) to determine which pages are potential dataset repository entry points, a verification step by crawling potential pages, a ranking function to select top entry pages as seeds, and a multi-armed bandit based approach to select discovery actions that balance the exploitation of accumulated reward and the exploration of future potential.

DSDD-BD+OL: this is our system that has the same settings as the DSDD-BD, but with online learning enabled. The classifier learns from the verified pages at each iteration.

DSDD-KW+OL: this discovery method has the same settings as the DSDD-BD+OL, but instead of a multi-armed bandit approach, the keyword search action is selected at each iteration.

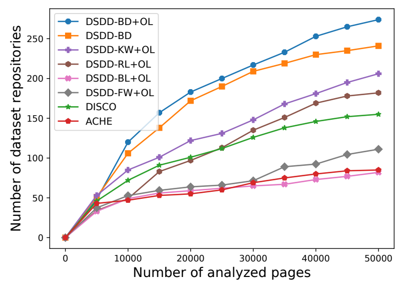
DSDD-RL+OL: this discovery method has the same settings as the DSDD-BD+OL, but instead of a multi-armed bandit approach, the related search action is selected at each iteration.

DSDD-BL+OL: this discovery method has the same settings as the DSDD-BD+OL, but instead of a multi-armed bandit approach, the backward link action is selected at each iteration.

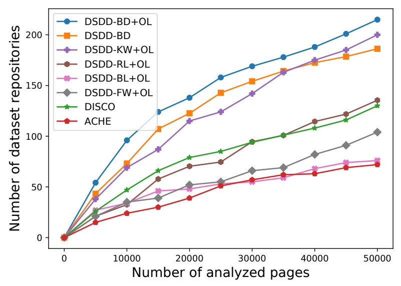
DSDD-FW+OL: this discovery method has the same settings as the DSDD-BD+OL, but instead of a multi-armed bandit approach, the forward crawling action is selected at each iteration.

As stated in the solution overview section, our system uses regular expression patterns to determine if a web page contains datasets. We use the same regular expression patterns for all baseline methods. A web page is considered as a dataset page if it contains one or more datasets (Definition 1), and it is considered as a dataset repository entry page if dataset pages can be discovered from it (Definition 2).

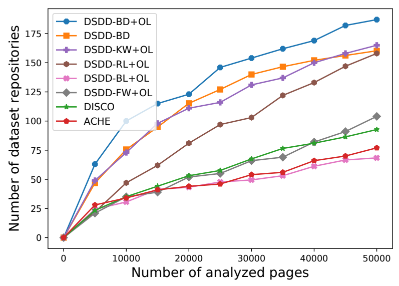
Figure 2 shows the number of dataset repositories discovered versus the number of web pages analyzed during the discovery process for the various systems and configurations. The DSDD-BD+OL configuration performs uniformly better than the others for all domains: it achieves nearly 100% higher harvest rate compared to baseline methods. The plot shows that:



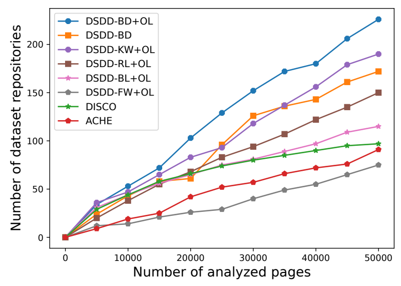
(a) COVID



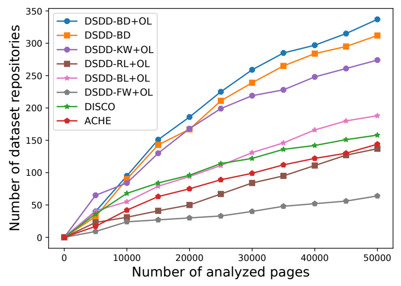
(b) Ethiopia



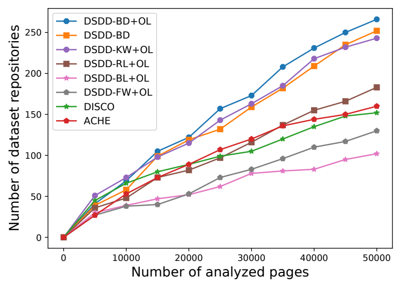
(c) Conflict



(d) Climate



(e) Education



(f) Economics

Figure 2: Harvest rate of dataset repositories for different discovery methods.

- Online learning is effective and results in a higher harvest rate (see the lines for DSDD-BD+OL and DSDD-BD), and
- for all the domains, DSDD-BD+OL consistently outperforms other methods that are limited to a single discovery action (e.g., DSDD-KW+OL).

For most domains, ACHE has the worst performance. Since ACHE relies on the Web network to crawl, it is not able find datasets that are not reachable from the crawling seeds. DISCO performs better than ACHE, but is less effective than our end-to-end system. This can be explained in part by the fact that DISCO lacks the ability to verify dataset repository entry pages and its rank-based model cannot be incrementally improved.

5.2 Recall of Different Discovery Methods

Because it is impractical to obtain all relevant dataset repositories as the ground truth for each domain, we compute the union of all discovered DREPs from all discovery methods as an approximate ground truth. Since DREPs are verified dataset repositories that contain dataset pages, this approximate ground truth can serve as a reasonable measure to compare the coverage of different discovery methods. In addition to the discovery methods introduced in Section 5.1, we include an additional baseline to this experiment:

SEARCH ENGINE: this method directly examines the results of keyword queries returned by a search engine. In other words, this method mimics the simple approach a human would use, without additional steps such as classification, verification and ranking. For example, given '*COVID*' as the domain of interest, this method issues '*COVID data*' and '*COVID datasets*' queries to a search engine and collects the results. These results are then examined by a script to check which of them contain dataset pages.

Let M denote the set of discovery methods

$$M = \{\text{DSDD-BD+OL, DSDD-KW+OL, DSDD-RL+OL, DSDD-BL+OL, DSDD-FW+OL, SEARCH ENGINE}\}$$

R_m denote the DREPs discovered by method $m \in M$, R_{union} denote the union of discovered DREPs by each method, and $R_{intersection}$ denote the intersection of DREPs discovered by the target method R_t and DREPs discovered by all other methods. More formally, we have:

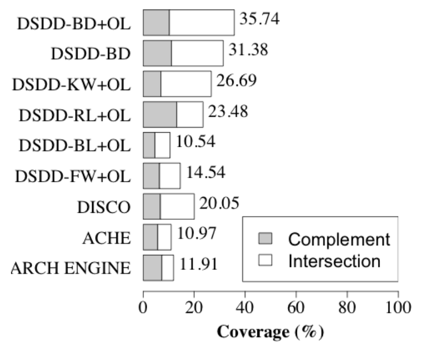
$$R_{union} = \bigcup_{m \in M} R_m \tag{5}$$

$$R_{intersection} = R_t \bigcap \{\cup_{m \in M \text{ and } m \neq t} R_m\} \tag{6}$$

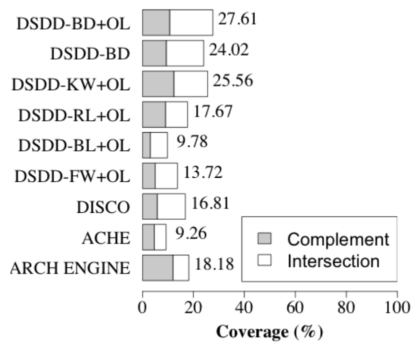
$$intersection = \frac{|R_{intersection}|}{|R_{union}|} \tag{7}$$

$$complement = \frac{|R_t| - |R_{intersection}|}{|R_{union}|} \tag{8}$$

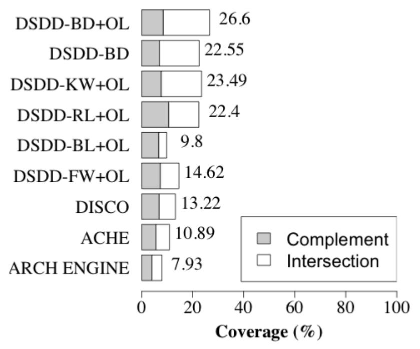
Figure 3 shows the coverage of different discovery methods in all domains. The *intersection* (in white) represents the percentage of related dataset repositories that are also discovered by all other methods, and the *complement* (in gray) represents the percentage of related dataset repositories that are discovered only by the corresponding method. A higher intersection value means the corresponding method attains a higher coverage. Each method explores different parts of the Web, and different discovered candidates result in a different set of new (discovered) DREPs, which can increase



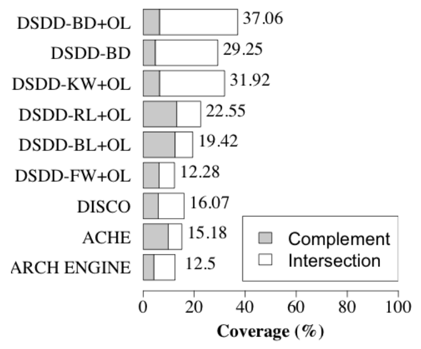
(a) COVID



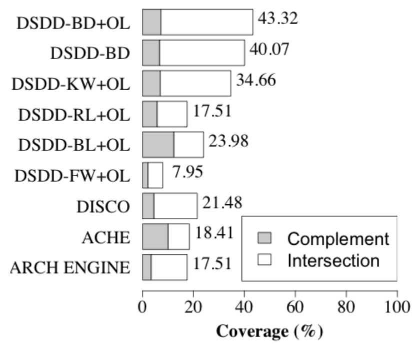
(b) Ethiopia



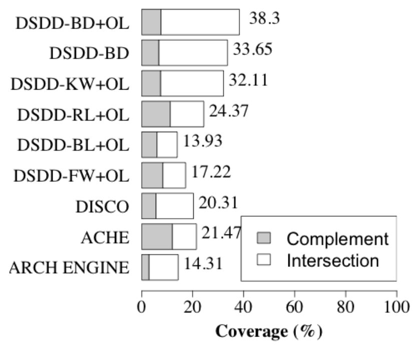
(c) Conflict



(d) Climate



(e) Education



(f) Economics

Figure 3: Coverage for different discovery methods. A larger intersection value means that the corresponding strategy is able to explore a wider range of the Web.

this discrepancy. As a result, no single method achieves absolutely high coverage. In general, the DSDD-BD+OL method consistently achieves the largest intersection among all domains. This suggests that although each discovery action explores a different region of the Web, the DSDD-BD+OL method is able to attain higher coverage by combining multiple actions using the multi-armed-bandits-based approach.

5.3 Discovered Dataset Pages

In the previous sections, we focused on the discovery of dataset repository entry pages. Here, we evaluate the effectiveness of different methods in discovering pages that link to datasets (i.e., dataset pages) and their relevance to the target domain. We also study the availability of dataset metadata which is the main dataset discovery mechanism used by web dataset search engines [10].

Number of Dataset Pages. As outlined in Algorithm 1, when the number of analyzed pages reaches a preset limit (e.g., 50,000 in our experiments), DSDD stops the iterative discovery process and starts crawling for dataset pages from collected dataset repositories. DSDD also stops the in-repository crawling process after certain number of pages are retrieved (also 50,000 in our experiment). Table 1 lists the number of dataset pages discovered by different discovery methods. For conciseness, we abbreviate SEARCH ENGINE as SE, and remove the prefix *DSDD* for the variants of our method (e.g., DSDD-FW+OL is listed as FW+OL in the table). The DSDD-BD+OL strategy consistently retrieves the largest number of dataset pages for all domains.

Quality of Dataset Pages. To measure the quality of discovered dataset pages, we define the *domain specificity* metric. Given a domain of interest D, *domain specificity* is the ratio of dataset pages *related* to D over the total number of dataset pages. To reduce the amount of manual labeling work required to compute this metric, we estimate it by labeling randomly sampled pages from the list output by Algorithm 1 (which is ranked with respect to the domain-specific query). More specifically, we split the result list at different percentiles (25%, 50%, and 75%), generating four disjoint parts. For each part, we randomly select a sample of pages (with sizes from 10% to 15%) and manually check how many of them are related to the given domain. Finally, we sum the different parts to calculate the *cumulative domain specificity* at different list positions, as shown in Table 2. We can see that for all domains, DSDD achieves a high cumulative domain specificity, with most domains having similar values, except for Ethopia, whose results are noisier.

Table 1: Number of dataset pages retrieved by different discovery methods.

	SE	ACHE	DISCO	FW+OL	BL+OL	RL+OL	KW+OL	BD	BD+OL
COVID	1756	771	1076	1526	916	1880	1859	2089	**2454**
Ethiopia	878	326	496	822	633	920	1188	1104	**1424**
Conflict	614	305	370	782	537	779	838	928	**1117**
Climate	1173	816	1281	1721	1622	2074	2169	2138	**2548**
Education	2284	1283	1824	2630	2124	2582	2628	2911	**3164**
Economics	1947	1192	1513	2163	1839	2507	2409	2477	**2858**

Table 2: Cumulative domain specificity for all domains.

	Top 25%	Top 50%	Top 75%	All
COVID	0.93	0.87	0.83	0.73
Ethiopia	0.82	0.76	0.66	0.61
Conflict	0.94	0.91	0.81	0.77
Climate	0.88	0.82	0.78	0.71
Education	0.90	0.86	0.81	0.74
Economics	0.85	0.81	0.76	0.70

Table 3: Percentage of discovered web pages that contain dataset metadata markup annotations.

Domain	Percentage
COVID	35.7%
Ethiopia	15.6%
Conflict	13.3%
Climate	12.8%
Education	10.9%
Economics	11.7%

Metadata and Markup Language Description. To detect the presence of dataset metadata on web pages, we followed the examples in Google Dataset Reference [26] and wrote a script to detect dataset metadata markup in all discovered dataset pages. We report the fraction of dataset pages that provide metadata markup descriptions in Table 3.

The results show that different domains have substantially different percentages of metadata coverage. For instance, the presence of metadata markup in the COVID domain is more than two times larger than in the domain with the lowest coverage (Education). We believe this difference may be explained by (1) the fact that most dataset pages in the *COVID* domain are new (since the global outbreak started in early 2020), and (2) the incentives for publishing datasets with metadata markups were created by the release of Google Dataset Search in 2018, which only adds to their index datasets found in web pages that provide dataset metadata markup [10]. By providing metadata markup, data publishers are able to improve the discoverability and visibility of their datasets. As a result, more dataset pages in the *COVID* domain are annotated with markup language than other domains that have datasets that have been published over a longer period of time.

Nonetheless, note that the percentage of pages that provide metadata markup is still relatively low in all domains. Our findings suggests that dataset search engines that rely solely on metadata markup for dataset discovery are likely to miss many potentially useful datasets for many domains.

6 CONCLUSION

In this paper, we propose DSDD, a system that enables domain-specific dataset discovery on the Web. Given a set of domain keywords, DSDD automatically discovers related datasets and data repositories. DSDD introduces methods to iteratively discover, classify and verify potential dataset repository entry pages, as well as rank the verified pages to select new seeds. We adapt a multi-armed bandit algorithm to balance the exploration and exploitation trade-offs in the selection of discovery actions, and by using online learning to incrementally improve the classifier that identifies the entry pages for data repositories, the approach automatically adapts to different domains. Our experiments on different social-good domains demonstrate the effectiveness of our framework compared to state-of-the-art strategies: DSDD attains high coverage and accuracy with little user input.

Determining the relevance of individual datasets, including data quality, value, and suitability to a problem is a challenging problem [41, 44, 58, 60], for which a completely automated solution is unlikely to exist. However, as users explore data collections through a dataset search engine, there is an opportunity to obtain feedback about the quality and relevance of datasets. In future research, we would like to investigate how to capture and leverage this feedback to improve discovery. Also, while DSDD can easily detect duplicate URLs, detecting duplicate datasets is difficult as datasets can be large and it is infeasible to download and parse their contents at crawling time. In the future, we plan to explore methods for detecting different URLs with similar text [3, 7, 29, 31].

ACKNOWLEDGMENTS

This work was partially supported by the DARPA D3M program and NSF award OAC-1640864. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF and DARPA.

REFERENCES

[1] Tarek Amr Abdallah and Beatriz de La Iglesia. 2014. URL-based web page classification: With n-gram language models. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management.* Springer, 19–33.
[2] ACHE. [n.d.]. ACHE Focused Crawler. https://github.com/ViDA-NYU/ache. (accessed August 20, 2021).
[3] Amit Agarwal, Hema Swetha Koppula, Krishna P Leela, Krishna Prasad Chitrapura, Sachin Garg, Pavan Kumar GM, Chittaranjan Haty, Anirban Roy, and Amit Sasturkar. 2009. URL normalization for de-duplication of web pages. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM).* 1987–1990.
[4] Tarfah Alrashed, Dimitris Paparas, Omar Benjelloun, Ying Sheng, and Natasha Noy. 2021. Dataset or Not? A study on the veracity of semantic markup for dataset pages. In *International Semantic Web Conference (ISWC 2021).*

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

[6] World Bank. [n.d.]. World Bank Open Data. https://data.worldbank.org. (accessed August 20, 2021).

[7] Ziv Bar-Yossef, Idit Keidar, and Uri Schonfeld. 2009. Do not crawl in the DUST: Different URLs with similar text. *ACM Transactions on the Web (TWEB)* 3, 1 (2009), 1–31.

[8] Luciano Barbosa and Juliana Freire. 2007. An adaptive crawler for locating hidden-web entry points. In *Proceedings of the International Conference on World Wide Web (WWW)*. 441–450.

[9] Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. 2016. Variations of the similarity function of textrank for automated summarization. *arXiv preprint arXiv:1602.03606* (2016).

[10] Dan Brickley, Matthew Burgess, and Natasha Noy. 2019. Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. In *The World Wide Web Conference*. 1365–1375.

[11] Sonia Castelo, Rémi Rampin, Aécio Santos, Aline Bessa, Fernando Chirigati, and Juliana Freire. 2021. Auctus: A Dataset Search Engine for Data Discovery and Augmentation. *Proceedings of the VLDB Endowment* 14, 12 (2021), 2791–2794.

[12] Soumen Chakrabarti. 2009. Focused Web Crawling. In *Encyclopedia of Database Systems*, Ling Liu and M. Tamer Özsu (Eds.). Springer US, 1147–1155.

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

[14] Soumen Chakrabarti, Kunal Punera, and Mallela Subramanyam. 2002. Accelerated focused crawling through online relevance feedback. In *Proceedings of the International Conference on World Wide Web (WWW)*. 148–159.

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

[16] New York City. [n.d.]. NYC Open Data. https://data.cityofnewyork.us. (accessed August 20, 2021).

[17] William W Cohen. 2003. Improving a page classifier with anchor extraction and link analysis. In *Advances in Neural Information Processing Systems*. 1505–1512.

[18] Guilherme T de Assis, Alberto HF Laender, Altigran S da Silva, and Marcos André Gonçalves. 2008. The impact of term selection in genre-aware focused crawling. In *Proceedings of the 2008 ACM symposium on Applied Computing*. 1158–1163.

[19] Jeffrey Dean and Monika R Henzinger. 1999. Finding related pages in the World Wide Web. *Computer networks* 31, 11-16 (1999), 1467–1479.

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

[21] DISCO 2021 (accessed May 2, 2021). The DISCO system. https://github.com/kienpt/site_discovery_public.

[22] Hai Dong and Farookh Khadeer Hussain. 2012. Self-adaptive semantic focused crawler for mining services information discovery. *IEEE Transactions on Industrial Informatics* 10, 2 (2012), 1616–1626.

[23] Martin Ester, Hans-Peter Kriegel, and Matthias Schubert. 2004. Accurate and efficient crawling for relevant websites. In *Proceedings of VLDB*. 396–407.

[24] Mohamed MG Farag, Sunshin Lee, and Edward A Fox. 2018. Focused crawler for events. *International Journal on Digital Libraries* 19, 1 (2018), 3–19.

[25] Koraljka Golub and Anders Ardö. 2005. Importance of HTML structural elements and metadata in automated subject classification. In *International Conference on Theory and Practice of Digital Libraries*. Springer, 368–378.

[26] Google [n.d.]. Google Search Central Documentation, Advanced SEO, Dataset. https://developers.google.com/search/docs/advanced/structured-data/dataset. (accessed August 20, 2021).

[27] Min-Yen Kan and Hoang Oanh Nguyen Thi. 2005. Fast webpage classification using URL features. In *Proceedings of the ACM International Conference on Information and Knowledge Management*. 325–326.

[28] Martin Klein, Lyudmila Balakireva, and Herbert Van de Sompel. 2018. Focused crawl of web archives to build event collections. In *Proceedings of the 10th ACM Conference on Web Science*. 333–342.

[29] Hema Swetha Koppula, Krishna P Leela, Amit Agarwal, Krishna Prasad Chitrapura, Sachin Garg, and Amit Sasturkar. 2010. Learning url patterns for webpage de-duplication. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*. 381–390.

[30] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. 1999. Trawling the web for emerging cyber-communities. *Computer networks* 31, 11-16 (1999), 1481–1493.

[31] Jyoti G Langhi and Shailaja Jadhav. 2018. Parallel Crawling for Detection and Removal of DUST Using DUSTER. In *International Conference on Computing Communication Control and Automation (ICCUBEA)*. IEEE, 1–5.

[32] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2020. *Mining of massive data sets*. Cambridge university press.

[33] Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Cambridge university press.

[34] James G March. 1991. Exploration and exploitation in organizational learning. *Organization science* 2, 1 (1991), 71–87.

[35] Filippo Menczer, Gautam Pant, and Padmini Srinivasan. 2004. Topical web crawlers: Evaluating adaptive algorithms. *ACM Transactions on Internet Technology (TOIT)* 4, 4 (2004), 378–419.

[36] Robert Meusel, Peter Mika, and Roi Blanco. 2014. Focused crawling for structured data. In *Proceedings of the ACM International Conference on Conference on Information and Knowledge Management*. 1039–1048.

[37] Tsuyoshi Murata. 2001. Finding Related Web Pages Based on Connectivity Information from a Search Engine. In *WWW Posters*.

[38] NASA [n.d.]. NASA Open Data Portal. https://data.nasa.gov. (accessed August 20, 2021).

[39] City of Chicago. [n.d.]. City of Chicago | Data Portal. https://data.cityofchicago.org/. (accessed August 20, 2021).

[40] Gautam Pant, Padmini Srinivasan, Filippo Menczer, et al. 2002. Exploration versus Exploitation in Topic Driven Crawlers.. In *WebDyn@ WWW*. 88–97.

[41] Jian Pei. 2020. A survey on data pricing: from economics to data science. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[42] Kien Pham, Aécio Santos, and Juliana Freire. 2018. Learning to Discover Domain-Specific Web Content. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*. 432–440.

[43] Kien Pham, Aécio Santos, and Juliana Freire. 2019. Bootstrapping Domain-Specific Content Discovery on the Web. In *The World Wide Web Conference*. 1476–1486.

[44] Leo L Pipino, Yang W Lee, and Richard Y Wang. 2002. Data quality assessment. *Commun. ACM* 45, 4 (2002), 211–218.

[45] United Nations World Food Programme. [n.d.]. Ethiopia country brief. https://www.wfp.org/countries/ethiopia. (accessed May 2, 2021).

[46] Xiaoguang Qi and Brian D Davison. 2006. Knowing a web page by the company it keeps. In *Proceedings of the 15th ACM international conference on Information and knowledge management*. 228–237.

[47] Xiaoguang Qi and Brian D Davison. 2008. Classifiers without borders: incorporating fielded text from neighboring web pages. In *Proceedings of the ACM SIGIR Conference*. 643–650.

[48] Xiaoguang Qi and Brian D Davison. 2009. Web page classification: Features and algorithms. *ACM computing surveys (CSUR)* 41, 2 (2009), 1–31.

[49] Disheng Qiu, Luciano Barbosa, Xin Luna Dong, Yanyan Shen, and Divesh Srivastava. 2015. Dexter: large-scale discovery and extraction of product specifications on the web. *Proceedings of the VLDB Endowment* 8, 13 (2015), 2194–2205.

[50] R Rajalakshmi and Chandrabose Aravindan. 2018. An effective and discriminative feature learning for URL based web page classification. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 1374–1379.

[51] Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

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

[53] Amany M Sarhan, Ghada M Hamissa, and Heba E Elbehiry. 2015. Feature Selection algorithms based on HTML tags importance. In *International Conference on Computer Engineering & Systems (ICCES)*. IEEE, 185–190.

[54] Sergej Sizov, Martin Theobald, Stefan Siersdorfer, Gerhard Weikum, Jens Graupmann, Michael Biwer, and Patrick Zimmer. 2003. The BINGO! System for Information Portal Generation and Expert Web Search.. In *CIDR*.

[55] Seán Slattery and Tom Mitchell. 2000. Discovering test set regularities in relational domains. In *ICML*. 895–902.

[56] Socrata. [n.d.]. The Socrata Open Data API. https://dev.socrata.com. (accessed August 20, 2021).

[57] Márcio LA Vidal, Altigran S da Silva, Edleno S de Moura, and João MB Cavalcanti. 2006. Structure-driven crawler generation by example. In *Proceedings of the ACM SIGIR Conference*. 292–299.

[58] Richard Y Wang and Diane M Strong. 1996. Beyond accuracy: What data quality means to data consumers. *Journal of management information systems* 12, 4 (1996), 5–33.

[59] Lars Werner, Stefan Böttcher, and Ralph Beckmann. 2005. Enhanced information retrieval by using HTML tags. In *Proceedings of the International Conference on Data Mining (ICDM)*. Citeseer, 24–29.

[60] Haifei Yu and Mengxiao Zhang. 2017. Data pricing strategy based on data quality. *Computers & Industrial Engineering* 112 (2017), 1–10.

[61] Yi Zhang and Zachary G Ives. 2020. Finding Related Tables in Data Lakes for Interactive Data Science. In *ACM SIGMOD Conference on Management of Data*. 1951–1966.

[62] Erkang Zhu. [n.d.]. Find Open Data. https://github.com/findopendata/findopendata. (accessed May 2, 2021).