Bootstrapping Domain-Specific Content Discovery on the Web

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

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

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

ABSTRACT

The ability to continuously discover domain-specific content from the Web is critical for many applications. While focused crawling strategies have been shown to be effective for discovery, configuring a focused crawler is difficult and time-consuming. Given a domain of interest *D*, subject-matter experts (SMEs) must search for relevant websites and collect a set of representative Web pages to serve as training examples for creating a classifier that recognizes pages in *D*, as well as a set of pages to seed the crawl. In this paper, we propose DISCO, an approach designed to bootstrap domain-specific search. Given a small set of websites , DISCO aims to discover a large collection of relevant websites . DISCO uses a ranking-based framework that mimics the way users search for information on the Web: it iteratively discovers new pages, distills, and ranks them. It also applies multiple discovery strategies, including keyword-based and related queries issued to search engines, backward and forward crawling. By systematically combining these strategies, DISCO is able to attain high harvest rates and coverage for a variety of domains. We perform extensive experiments in four social-good domains, using data gathered by SMEs in the respective domains, and show that our approach is effective and outperforms state-of-the-art methods.

CCS CONCEPTS

• **Information systems → Web searching and information discovery**; *Page and site ranking*; *Content ranking*; Rank aggregation.

KEYWORDS

Focused crawling, Domain-specific Website discovery, Meta search

ACM Reference Format:
Kien Pham, Aécio Santos, and Juliana Freire. 2019. Bootstrapping Domain-Specific Content Discovery on the Web. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313709

1 INTRODUCTION

The ability to continuously discover content relevant to an information domain has many applications, from helping in the understanding of humanitarian crises to countering human and arms trafficking. It is estimated that tens to hundreds of thousands of escort ads are posted every day and about 75% of reported human-trafficking survivors were advertised online at some point [5]. Recently, the US law enforcement has shut down the sex marketplace website backpage.com due to its role in sex trafficking. Criminals also take advantage of online markets to avoid background checks and trade illegal weapons. According to an FBI report, hundreds of thousands of guns slip through the federal background-check system every year [30]. At the same time, many sites advertise weapons for sale – in armslist.com alone one can find hundreds of thousands of weapon-related ads.

By collecting and analyzing domain-specific content from the Web, we can better understand these illegal activities, generate leads, and obtain evidence for investigations. But doing so is challenging. Currently, subject matter experts (SMEs) (e.g., from non-governmental organizations and government agencies) must collaborate with computing experts and data scientists to construct specific solutions to their problems, which include (1) creating machine-learning models to recognize relevant content, and (2) configuring crawlers that retrieve the information at scale. Once these are available, it is possible to continuously discover relevant content by using re-crawling strategies [10, 23, 27]. However, both of these tasks require users to obtain a sufficiently large number of pages and finding these pages is difficult.

In this paper, we address this key problem in bootstrapping domain-specific search: how to effectively discover relevant websites given a small seed set provided by SMEs. A natural approach for discovering relevant websites is to use commercial search engines, such as Google and Bing, and issue keyword queries or search for similar sites using "related" search. This requires SMEs to go through an iterative process in which they search, distill the results, and use the information gathered to formulate additional queries. This process is cumbersome and inefficient for large-scale information gathering tasks. Different approaches have been proposed to tackle this problem. The Domain Discovery Tool (DDT) [15] aims to simplify the process of constructing classifiers for a given domain. To help users discover relevant pages and train a domain classifier, it provides an easy-to-use interface that summarizes search results and helps users formulate new search queries. While the tool is effective, it was designed with a user-in-the-loop philosophy and thus requires substantial user effort to perform the discovery operations and review the results. Methods such as forward and backward crawling [3] and DEXTER [24] have been be proposed to automate discovery. Because they rely on classifiers, they need to be adapted for different domains. Besides, these methods fall short of achieving high precision and recall in domains where relevant websites are not highly connected. To the best of our knowledge, there has been no systematic comparison of these methods for different domains, therefore whether they are effective for website discovery is still an open question.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313709

Several focused crawling and discovery techniques could potentially be adapted for this problem. However, they all rely on the availability of an accurate domain-specific classifier [3, 13, 31]. This is an unrealistic assumption for the many application scenarios where experts must start with a small set of relevant websites , since a small sample is unlikely to be sufficient construct an accurate classifier. These techniques use the classifier to select discovered pages that are relevant, and use content of these pages to train other models that directly control future search, e.g., ordering unvisited URLs and generating search queries. Therefore, a weak classifier would gradually degrade the effectiveness of discovery.

To address this challenge, we propose DISCO, a new method for website discovery that does not depend on an accurate domain-specific classifier. Given a small number of sample pages, DISCO automatically discovers additional pages that can be used both to construct domain classifiers and to serve as seeds for focused crawlers. The strategy adopted by DISCO was inspired by how users follow the search process: it searches for pages on the Web and distills the results in an iterative fashion. Different operations can be used to discover new pages, including keyword-based and related queries issued to search engines, and backward and forward crawling. However, the effectiveness of an operation varies across domains. For example, in our experiments (Section 5), we observed that backlink search performs the best for human trafficking domain, but is much less effective for the others. Based on this observation, unlike previous works [3, 31], DISCO applies multiple search operations. To systematically select the search operation as the discovery progresses, we propose an algorithm that uses the multi-armed bandit strategy [1]. This makes DISCO adaptive both within and across domains, and leads to higher harvest rates and coverage compared to a fixed strategy.

To distill the results, in the absence of a model that recognizes relevant pages, DISCO approximates how users select relevant pages by ranking them. Given a set of discovered pages, it ranks higher pages that are more similar to the input sample set. The challenge here lies in designing a strategy that is effective. We carried out an empirical study of several ranking approaches and found that none of the approaches is uniformly better for all domains. We propose a new ranking function that, by combining the results of multiple functions, produces a robust ranking.

Contributions. Our contributions can be summarized as follows:

- We propose DISCO, a framework that automatically discovers relevant websites requiring only a small set of example websites . To the best of our knowledge, ours is the first approach that does not require an accurate classifier to bootstrap domain discovery.
- To distill results, we propose an ensemble ranking approach that combines multiple independent ranking functions, and show that it consistently attains higher precision than the independent functions.
- We incorporate different search operations in our framework and propose an algorithm that uses the multi-armed bandits strategy to select the best search operator at each iteration. We also perform a systematic evaluation of the performance of the different operations in multiple domains and show that, by using the multi-armed bandits-based algorithm, our method is able to attain high harvest rates.
- We perform extensive experiments in multiple social-good domains using real data provided by SMEs from the corresponding domains. We report experimental results which show that DISCO obtains 300% higher harvest rate and coverage compared to state-of-the-art techniques for domain discovery.

Outline. The remainder of this paper is structured as follows. We discuss related work in Section 2. In Section 3, we define the website discovery and ranking problems and give an overview of the DISCO framework. In Section 4 we describe the ranking and discovery components of the framework. We present the results of our experimental evaluation in Section 5, and conclude in Section 6, where we outline directions for future work.

2 RELATED WORK

Several techniques have been proposed to discover domain-specific web content. These can be categorized in the following two groups: *Search-based Discovery* techniques [24, 31, 32] which rely on search engine APIs (e.g., Google, Bing, and Alexa APIs) to find web pages similar to given keywords or web pages ; and *Crawling-based Discovery* techniques [2, 3, 21], which use the Web link structure to explore new content by automatically downloading and recursively following links extracted from the discovered web pages . Note that while some of these techniques were proposed to discover specific web pages , they can be extended to discover websites as well. To the best of our knowledge, as we discuss below, none of the prior work supports all different discovery techniques, and most require an accurate domain-specific classifier.

2.1 Search-Based Discovery

There are two main search-based discovery operators supported by search engine APIs: *keyword search* and *related search*. Most of the existing discovery techniques use keyword search since related search is only supported by two search engines, Google and Alexa.

Earlier work [11, 21] used both content and Web link structure to compute the relatedness between web pages . Vieira et. al. [31] proposed a system that uses relevance feedback to gather seeds to bootstrap focused crawlers. It submits keyword search queries to Bing; extracts keywords from the result pages classified as relevant for the focus domain; and uses these keywords to construct new search queries. Their system shares some of our goals and their findings support our design decisions. They show that submitting multiple short queries and retrieving a small list of results leads to more relevant pages than submitting long queries and extracting a large number of results. The major drawback of this approach is that, when the classifier is not accurate, the search results quickly diverge from the target domain. Furthermore, the system is specifically designed to use with keyword search, therefore, does not support other search techniques.

Disheng et. al. [24] presented a discovery pipeline that uses keyword search to find websites containing product specifications (e.g., cameras and computers). While they also utilize backward search, they showed that the relevant websites are mainly discovered by searching with product identifications extracted from previous discovery iterations. However, using product identifications as search keyword is specific to product search domain and therefore not applicable to other domains.

Wang et. al. [32] focused on the claim-relevance discovery problem that, if solved, can help to identify online misinformation. They proposed a system that extracts salient keywords from fact-checking articles and uses them to search for candidate documents. Unlike in our search strategy, their search phase is not iterative, since they do not aim to attain high coverage of relevant documents.

2.2 Crawling-Based Discovery

Forward and backward crawling are two types of crawling-based techniques explored in the literature. In forward-crawling, links are extracted from the discovered web pages . Several forward-crawling approaches been proposed to discover domain-specific web pages and websites [4, 7, 8, 13]. The most prominent approach in this group are the *focused crawling* methods that employ online learning policies [4, 7, 20]. They use two classifiers to focus the search: the critic, a classifier that categorizes web pages as relevant or irrelevant to the domain; and the apprentice, a classifier that learns to identify the most promising links in a domain-relevant page in an online fashion. Note that while the apprentice classifier is automatically learned, the critic is a classifier given as input to the system and requires a substantial number of positive and negative examples to be trained on. In contrast, our approach does not require an input classifier, just a small set of relevant pages. Compared to search-based techniques, forward crawling is more scalable, however, it works best for domains where relevant content is well-connected.

In contrast to forward crawling, backward crawling (or reverse crawling) discovers new pages through backlink search. Since the Web is unidirectional, it must be done through search engine APIs that expose results of previous large-scale crawls. Early work proposed the use of backlink search to discover Web content [6, 12]. More recently, Barbosa et. al. [3] proposed a crawling strategy that combines both backward-crawling and forward-crawling techniques to locate bilingual websites. It employs two link classifiers, trained with the content of visited pages and their relevance to the target domain, to determine the visiting orders of the links discovered by the two techniques. As a result, the efficiency of this strategy highly depends on the domain classifier. In addition, this approach has the same limitation we discussed above, in that the crawler becomes ineffective when the classifier is not accurate. In later work, Barbosa [2] proposed a similar crawling architecture for discovering forum websites .

Approaches that focus on real-time discovery [26, 29, 34] have taken advantage of social media platforms (e.g., Twitter) to monitor and track domain-specific content such as misinformation [29], activity related to natural disasters [26], and rumors [34]. Although these techniques mainly target social-media feeds, they could potentially be used to discover websites by extracting links present in the feeds. Discovery from social media could be integrated in our approach as an alternative discovery operation.

3 PROBLEM AND SOLUTION OVERVIEW

3.1 Problem Definition

Website Discovery Problem. Let D represent the domain of interest and w denote a website represented as one of more web pages

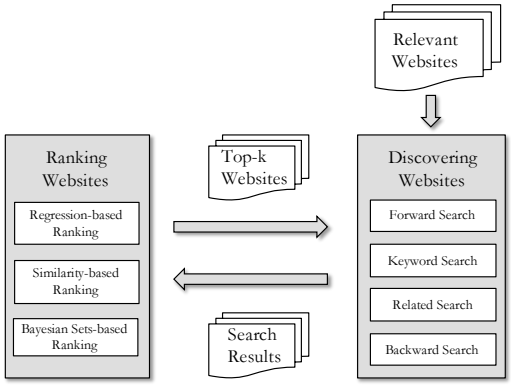


Figure 1: Architecture of DISCO.

. If w is relevant to the domain, we say that $w \in D$. Given a small set of websites $S \subset D$ (seeds), our goal is to efficiently expand S by finding additional websites w that belong to D.

Website Ranking Problem. In the absence of a model that recognizes pages in D, we approximate the problem to make it depend only on the set S of seeds by framing it as a ranking task: given S, the goal is to discover new websites that are *similar* to S. More formally, given set of discovered websites D_+, the problem consists of ranking each site w in D_+ according to its similarity with S – this is equivalent to the likelihood of w being in D. The intuition is that the closer w is to S, the more likely it is that $w \in D$.

By ranking the discovered websites based on their similarity with S, we help to reduce the SMEs effort in reviewing and labeling examples to build a domain-specific classifier and selecting important websites for the re-crawling process. A ranking strategy in this scenario should ideally have two important features: high precision in the top of the ranked list and low false negative in the tail of the ranked list. The former feature is important for the website discovery operation to be effective since the top-ranked websites are fed back into the discovery process. The latter is useful for reducing the SMEs effort of reviewing the final ranked list output by the system: the fewer relevant documents in the tail of the list, the fewer documents need to be reviewed by the SMEs. In our experimental evaluation in Section 5, we use metrics aligned with these desired features to evaluate different ranking functions.

3.2 Framework Overview

The design of DISCO was inspired by the iterative process that SMEs usually follow to discover domain-specific websites . They start by submitting queries to a search engine. Then, they review and evaluate the search results based on their knowledge about the domain. The new knowledge they acquire in this process is used to refine and formulate new search queries. In DISCO, we replace the manual evaluation with the website ranking component and use search engine APIs to automate the search operations. Figure 1 depicts the high-level architecture of the DISCO framework. It consists of two main components: *website ranking* and *website discovery*. To incorporate different search engine APIs and ranking techniques, we standardize the input and output of the two components. Specifically, the website discovery component takes a list of

Algorithm 1 Website Discovery

```
1:  procedure Discovery(seeds)
2:      results = ∅
3:      ranked_results = ∅
4:      topk = seeds
5:      while stop condition do
6:          op = select_discovery_operator()
7:          search_results = op.search(topk)
8:          results = results ∪ search_results
9:          ranked_results = rank(results)
10:         topk = get_top_k(ranked_results)
11:     end while
12:     return ranked_results
13: end procedure
```

relevant websites as input and returns a list of discovered websites after performing the search operations. Similarly, the input and output of the website ranking component consist of lists of discovered websites so far and the corresponding list of ranked websites , respectively. This design unifies multiple search operators in a single framework, and hence, it allows us to fairly compare their efficiency across different domains (see Section 5). Algorithm 1 outlines the discovery process carried out by DISCO. It starts with the seeds and iteratively searches, ranks and selects the top results. The *select_discovery_operator()* function in line 6 returns one of the four discovery operators: forward crawling, backward crawling, keyword search and related search.

4 RANKING AND DISCOVERY

In this section, we present the details of the website ranking and discovery components of DISCO. Then, we present a technique to efficiently select the discovery operator based on the multi-armed bandits algorithm.

4.1 Website Ranking

The website ranking problem can be cast as a conventional ranking problem in information retrieval, where S represents the query, and the goal is to rank the websites in D_+ according to its similarity to the query. The query here consists of a very long document containing all the websites in S, which makes the traditional ranking models (e.g., BM25 [25]) less effective. Furthermore, merging all the websites in S can potentially blur the distinct domain-specific features that exist in individual websites . To avoid this problem, we investigate other ranking approaches that consider websites in S individually. We then propose a new ranking function, inspired by ensemble learning methods, that combines the scores of the existing approaches.

Website Representation. The first step of any ranking function design is to represent the candidate documents, which are websites in our context. A website can be represented as a single or multiple web pages . We consider both options and selected the latter for several reasons. To use multiple pages to represent a site, we need to have an effective procedure to select representative pages in the sites. This task is challenging, especially when we do not have ground truth data. Random selection tends to result in a large number of web pages that are not good representative of the site and that lead to noise that negatively affects the ranking. The single-web page representation can help circumvent this issue if we rely on the assumption that every website contains at least one web page that can serve as an effective representative of the website content. As a result, the score of a website can depend only on the score of its most relevant web page discovered. Furthermore, the single-web page representation greatly simplifies the implementation of the framework and improves its performance. For this representation, we do not need a separate process to retrieve and manage additional pages for each newly discovered website , and discovered web pages can be processed independently. This makes the framework ready for parallelization.

In fact, we experimentally compared selection strategies using multiple web pages representation against the single-page representation, and the former led to better rankings than the latter for all domains we considered. We considered two different approaches for the multiple-page representation. One performs a breadth-first-search crawling starting from the home page, which leads to both relevant and irrelevant pages. The other uses in-site search via search engine APIs (i.e., using queries with the format "site:keyword"). This method, however, is keyword-sensitive and tends to retrieve false positive pages.

Several ranking function have been proposed in the literature. In what follows, we describe the ones we considered. Details about their performance are discussed in Section 5.

4.1.1 Regression-Based Ranking. This approach learns the ranking function using regression-based techniques and computes a score for each website in D_+. The challenge is that only positive examples (i.e., S) are given, and thus additional negative examples must be obtained or unconventional techniques such as PU learning [18] or novelty detection [28] must be applied. We investigate three regression techniques in this direction and describe how they can be applied in our scenario.

Binomial Regression. In order to use binomial regression (i.e., logistic regression), we need both positive and negative examples. Therefore, we obtain negative examples by randomly selecting web pages from a public Web corpus. Since the Web corpus is relatively large compared to the number of samples pages, there is only a very small chance of selecting a web page relevant to the domain. Given S and the unlabeled web pages as negative examples, we train a binomial regressor and use it to compute ranking scores.

Positive and Unlabeled Example Learning (PU Learning). A well-known alternative to binomial regression method, given positive and unlabeled examples, is PU learning [18]. Liu et al. [18] proposed biased SVM, a PU learning technique that frames the PU Learning problem as a constrained optimization problem. They experimentally showed that biased SVM outperforms other 2-steps PU learning approaches. Therefore, we choose this technique to learn the ranking function.

Novelty Detection. Another line of research that does not require negative examples is novelty detection [19, 28]. Scholkopf et. al. [28] extend SVM (support vector machine) to learn a function that is positive on the positive examples and negative on its complement.

Intuitively, it computes the minimal region in feature space that encloses the positive examples.

4.1.2 Similarity-Based Ranking. Similarity-based ranking, inspired by the k-nearest neighbors algorithm, computes the ranking score of a website according to the average of its similarities to all websites in S. Let $Sim(w_i, w_j)$ be a similarity function between the websites w_i and w_j, the score of a website $w \in D_+$ is computed as follows:

$$Score(w) = \frac{1}{|S|} \sum_{w_s \in S} Sim(w, w_s) \tag{1}$$

We consider Jaccard and Cosine as similarity functions. These are two well-known similarity measures for textual content that use distinct input representations, which might encapsulate different ranking advantages. We use the vector space model and binary vector to represent websites for computing Cosine and Jaccard similarity respectively. Since these similarity measures use different representations of the websites , they might expose different characteristics in the ranking task. Let x, y be the representation of websites w_x, w_y, the Jaccard index and Cosine similarity between them are computed as follows:

$$Jaccard(x, y) = \frac{x \cap y}{x \cup y} \tag{2}$$

$$Cosine(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{3}$$

4.1.3 Bayesian-Sets-Based Ranking. Bayesian Sets (BS) is a Bayesian inference technique introduced by Ghahramani and Heller in [14] that performs set expansion – it generates high-precision sets of related items. Specifically, given a set of related items that belong to a class C, BS scores a new item by estimating its normalized marginal probability that the item belongs to C. By this definition, BS can be directly applied to our ranking problem by considering the related items as seeds S and class C as domain D. Accordingly, the score of a website w is computed as the probability of w given that S is observed:

$$Score(w) = \frac{P(w|S)}{P(w)} \tag{4}$$

4.1.4 Ensemble Ranking Function. While many alternatives are possible for ranking functions, none of them is uniformly better for all domains, as we experimentally show in Section 5. Thus, ideally, we should be able to combine them. Ensemble learning is a paradigm that combines multiple machine-learning models and has been shown to be effective at reducing their variance. In our context, each ranking function captures a different set of features, hence their combination has the potential to improve the robustness and reduce the variance of the ranking. Inspired by this, we propose a new ranking function that combines the results from multiple ranking functions. Specifically, let $F = \{f_i\}$ be the list of the ranking functions and $f_i(w)$ denote the position of w in the ranked list that w belongs to. The new ensemble score of w is computed as the mean of these position values:

$$Score(w) = \frac{\sum_{f_i \in F} f_i(w)}{|F|} \tag{5}$$

Another option could be to directly consider the similarity scores as $f_i(w)$. However, these scores must be calibrated for the ensemble function to be effective. Our similarity functions do not produce naturally calibrated scores (i.e., probabilities), hence, we decided to use rank positions for combining the results. We also considered using other rank combination methods that have been proposed in previous work, e.g., reciprocal rank fusion [9]. However, we found that the mean function performed comparably well in our case and at same time it was faster.

4.2 Website Discovery Operators

The goal of the discovery operators in our context is to discover new websites , therefore we make use of existing discovery methods in a way that maximizes this objective. We investigate four different discovery techniques: forward crawling, backward crawling, keyword search, and related search. To incorporate these discovery methods in our framework, we designed them in such a way that they take the same input, the top-k most relevant websites , and produce the same output, a list of new websites . In what follows, we describe in detail each of these methods.

Forward Crawling. The forward crawling operator discovers new websites through outlinks extracted from the HTML content of the input web pages . Since the objective is to maximize the number of relevant websites , the operator only returns the outlinks for websites have not been discovered in the previous iterations.

Backward Crawling. The backward crawling operator discovers new websites by performing two operations: backlink search and forward crawling. In backlink search, it calls search engine APIs to obtain backlinks for the input web pages . Since the input is relevant to the domain, the discovered backlinks are likely to serve as hubs pointing relevant pages. This phenomenon has been referred to as co-citation in the literature [16]. Then, we can discover additional relevant websites by extracting outlinks from the hubs and doing forward crawling.

Keyword Search. The keyword search makes use of search engine APIs to search for websites relevant to the keywords. The problem is how to obtain domain-specific keywords from the input websites that lead to more relevant websites . The body text extracted from the input pages contains good keywords that are representative of the domain, however, they often co-occur with terms that may not be representative. Our observation is that the keywords extracted from the metadata tags (i.e., description and keyword) are significantly less noisy than the ones extracted from the body text. For this reason, they are particularly effective for searching for relevant websites . For each page, we extract and tokenize the metadata from the description and keywords tags. We select the most frequent tokens as search keyword candidates. The problem is that, even though many keywords are relevant to the domain, do not comprise sufficient information to retrieve relevant websites . For example, the keywords *pistol*, *texas*, and *sale* are related to the *weapon forums* domain, but more contextual information is needed to build search queries that find additional relevant websites . To address this challenge, we combine these keywords with terms that are good descriptors of the domain. For example, in the *weapon forums* domain, given *gun forum* as seed keyword, the corresponding search terms of the previous keywords are *gun forum pistol*, *gun*

forum texas and *gun forum sale*. Note that we only need one seed keyword for each domain, so this should not add any burden to the SMEs.

Related Search. The related search operator takes a website URL as input and returns a list of related websites . To our knowledge, only Google Search APIs and Alexa APIs support this operation. Depending on the nature of the data that these search platforms possess and their underlying algorithms, the results can be vastly different. Nevertheless, it appears to be a natural fit for our objective of discovering relevant websites . Note also that different search providers can be easily included as different search operators.

4.3 Discovery Operator Selection

In the previous section, we have described four different operators for discovering relevant websites , the challenge now is how to select an operator at each iteration in Algorithm 1 that maximizes the rate of discovering new relevant websites .

This is a non-trivial task for several reasons. First, it is not clear which search operator performs the best across multiple domains. In our experiments, described in Section 5, backward crawling performs better than other operators for website discovery in the Human Trafficking domain, but it performs poorly in other domains. Furthermore, favoring only search operators that are known to be good can leave the discovery process stuck in a connected region of the Web. On the other hand, using different discovery operators can diversify the discovery process, and eventually improve the rate of harvesting relevant websites .

To address this problem, we propose a strategy to balance the trade-offs between different discovery operators using multi-armed bandits (MAB), particularly UCB1 algorithm [1]. MAB is known as an effective means to solve the exploitation and exploration problem. The goal of this strategy is to select bandit arms with the highest accumulated reward and to penalize the ones that were overly operated in the past. To accomplish this, we need to define two variables in the UCB1 algorithm: the bandit arm (*op*) and the reward of its operation (μ_{op}). We model each discovery operator as a bandit arm. Assume that each time t, when an arm *op* is selected, it discovers $n_{op,t}$ websites . According to the UCB1 algorithm, the score of *op* at time t is defined as:

$$Score_{op,t} = \overline{\mu_{op}} + \sqrt{\frac{2ln(n)}{n_{op,t}}} \tag{6}$$

In this equation, n is the total number of websites retrieved by all operators. At each iteration, the algorithm selects the operator with highest score to perform the discovery.

The remaining challenge is to define the reward of an operator so that it is proportional to the number of relevant websites it discovered. Although we do not know the relevance of the discovered websites , we can approximate them by their ranking scores computed by the website ranking component (Section 4.1). Let pos_i denote the position of a website i in the ranked list. The normalized reward of i is $1 - \frac{pos_i}{len}$, where *len* is length of the ranked list. By this definition, if i is ranked at the end or top of the list, its reward is 0 or 1 respectively. Since the goal is to maximize the number of new relevant websites , we assign 0 reward to a website that was already discovered in previous iterations, regardless of its position. As a result, the reward of i is redefined as $(1 - \frac{pos_i}{len}) * I_i$, where I_i takes binary values:

$$I_i = \begin{cases} 0 & \text{website } i \text{ was already discovered} \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

If a discovery operator *op* returns k websites , its reward μ_{op} is defined as follows.

$$\mu_{op} = \frac{\sum_{i=1}^{k}(1 - \frac{pos_i}{len}) * I_i}{k} \tag{8}$$

We use this reward to compute the score of an operator at time t ($Score_{op,t}$) in the equation 6. After each round of search, we recompute this score and select the operator with highest score for the next round.

5 EXPERIMENTAL EVALUATION

In this section, we evaluate the effectiveness of our discovery framework and ranking functions through extensive experiments using real-world datasets. We use the following domains for the experiments:

- Human Trafficking (HT): classified ads and other escort-related websites which often contain adult ads. Prominent example websites in this domain are `backpage.com`, `craigslist.com`, and `adultsearch.com`.
- Weapon Forums (Forum): discussion forums about firearms and weapons.
- Weapon Marketplaces (Market): marketplace or classified ads websites for firearms and weapons.
- Stock Promotions (SEC): websites that post stock promotion alerts. This domain contains information that can support investigations of micro-cap fraud.

We selected these domains for the following reasons. First, since they are different in nature, they allow us to evaluate the robustness of our proposed techniques and baselines. Most importantly, we had access to the seed websites and the training examples manually labeled by SMEs in these domains. The SMEs were analysts from government agencies and worked with us in collecting and reviewing relevant content. The number of seed websites in HT, Forum, Market and SEC domains are 54, 103, 23, and 18 respectively. These are representative websites that contain a significant number of pages relevant to the domains. These seed websites are used for the evaluation of the ranking methods in 5.2 and for the input of the DISCO framework in Section 5.3. As training examples (described in Table 1), we used a bigger set of websites, labeled as relevant and non-relevant by SMEs in HT, Forum and Market domains. We use these examples to create ground-truth classifiers for the evaluation of the DISCO framework, which we present in Section 5.3.

It is worthy of note that although we were able to obtain ground-truth to train these classifiers, labeling the pages required a very time-consuming process. In addition, the labeling process was not a trivial task for the SMEs, who are familiar with the domain but not with the intricacies of learning. For example, they did not know the impact of labeling some confusing cases would have in the final model. An important motivation for this work is to devise an easier-to-use and less time-consuming process to solve this problem.

5.1 Evaluation Metrics

5.1.1 Website Ranking Metrics. Section 3.1 presented two desired characteristics of good ranking functions: high precision in the top of the ranked list and low false negative in its tail. In order to measure these qualities, we use the following metrics:

P@k: computes the percentage of the relevant websites in the top-k websites of the ranked list. This metric captures the quality at the top of the list.

Mean Rank: computes mean of the ranking positions of the relevant websites in the ranked list.

$$MR = \frac{1}{n} \sum_{w_i \in D_+^*} position(w_i) \tag{9}$$

We do not normalize the rank by the length of the ranked list because we want to keep the absolute values for easy interpretation. Also, in our experiment, the length of the ranked lists in the experimental domains are similar, therefore absolute values are still valid for cross-domain comparison.

Median Rank: position of the relevant website whose position after ranking is the midpoint of all the relevant websites .

5.1.2 Website Discovery Metrics. We measure the discovery efficiency using *Harvest Rate* and *Coverage* metrics. In the following metric definitions, let D_+^* be the set of discovered websites that are in the domain D.

Harvest Rate: the ratio between the size of D_+^* and the size of D_+. The Harvest Rate is a standard metric that has been widely used in literature to measure the efficiency of focused crawlers.

Coverage: we define coverage as the percentage of websites in D that are discovered.

5.2 Website Ranking Evaluation

For each domain, we randomly select 10,000 websites from DMOZ directory[1] as negative candidates. Since the domains are rare and the size of DMOZ is large (about 4M websites), there is only a small chance that the selected examples are relevant to the domain. Therefore, we can reliably consider these as negative candidates. For each domain, we split the seed websites into train and test sets randomly. We merge the test set into the set of negative examples, creating a new set named the *candidate set*. Then, we rank the candidate set using the ranking functions described in Section 4.1. We compute the evaluation metrics based on the positions of the test set in the ranked list.

JACCARD, COSINE: These ranking functions compute the similarity between two websites using JACCARD and COSINE similarity index, respectively.

BS: This ranking function uses Bayesian sets technique. We represent each website as a term-frequency vector as opposed to the binary vector used in [14].

ONECLASS: This ranking function uses novelty detection (i.e., one-class SVM [28]).

BINOMIAL: This ranking function uses binomial regression. We use logistic regression because it is fast and known for performing well on text data. To obtain the negative examples, we randomly select websites from DMOZ. The number of selected examples was equal to the number of positives examples. For the implementation of both *ONECLASS* and *BINOMIAL*, we used scikit-learn [22].

PUL: This ranking function uses biased SVM [18]. We use SVM light[2] as an implementation of biased SVM. We also use DMOZ to obtain the unlabeled examples for learning this function.

ENSEMBLE: This is our ranking function that combines all the above ranking functions, except *PUL*, whose performance showed a high variability across different domains.

Figures 2, 3, and 4 compare the ranking functions using different metrics: *P@k*, mean rank and median rank. Note that we select a k that is equal to the number of relevant websites (test set) in the candidate set. By doing this, we basically normalize the *P@k* values to range [0, 1]. The figure shows that the ENSEMBLE ranking function achieves better performance than the individual ranking functions do in most of the cases. This confirms that the combination of different ranking functions is able to reduce the variance and increase the robustness of the overall ranking. We also observe that *BS* consistently performs better than other individual ranking functions in median rank metric, i.e., it rarely places a relevant website to the end of the list, which is one of our desired features. Another observation is that the precision in SEC domain is relatively low (66.67%) compared to the others. This can be explained by the fact that in this domain, relevant websites might contain a mix of topics that are less distinguishable. Another reason is that this domain has smallest number of seeds among other domains.

Influence of the Number of Seed Websites . To understand how the number of seed websites impacts the effectiveness of the ranking, we conduct a ranking evaluation with different number of seed websites . Figure 5 shows the *P@k* corresponding to number of seed websites . We notice that the precision improves when we increase the number of seed websites up to 13 and stabilizes after that. This finding suggests that this number of seeds is sufficient to achieve a good ranking performance. It also reveals that probably there is still room to improve ranking performance, especially for cases in which a larger number of seed websites is available. Given recent advances in text representation using dense vector representations and document distance computation in the embedding space [17, 33], this is a promising area for future work. While this is a limitation of our ranking approach, it is still suitable for our context, where we assume to have only a small list of seeds as input.

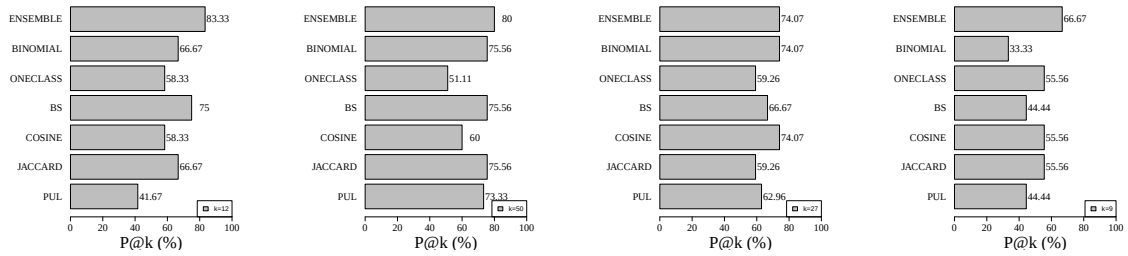
5.3 Discovery Evaluation

In this section, we assess the effectiveness of DISCO and compare it to state-of-the-art discovery strategies [3, 4, 31] (baselines). Since the baselines require a page classifier to operate, we train one classifier for each domain using seed websites and unlabeled examples as training data. Specifically, we consider seed websites as positive examples, and randomly select pages from DMOZ to use as negative examples. For training the classifiers, we use an implementation of SVM from the scikit-learn library [22].

To compute the discovery metrics, we train one ground-truth page classifier for each domain using the positive and negative examples manually labeled by SMEs. Note that these classifiers are only used for evaluation purposes. Although not ideal, these classifiers are the closest to the real ground truth that we can possibly

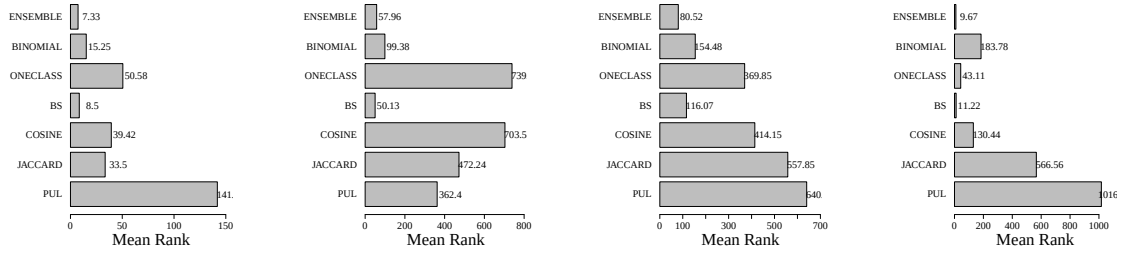
[1] http://dmoz-odp.org/

[2] http://svmlight.joachims.org/



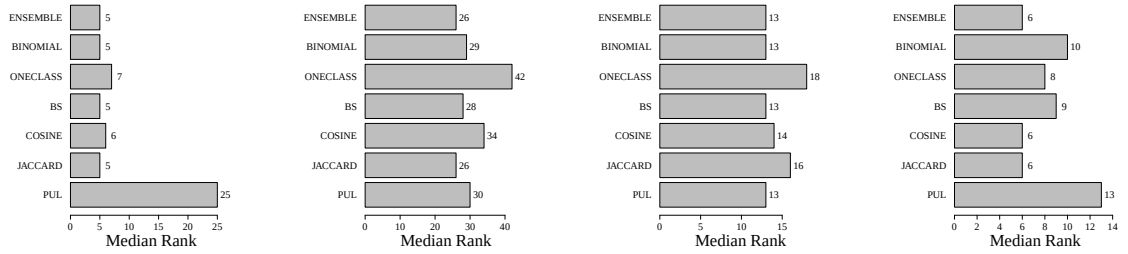
(a) Market Domain (b) Forum Domain (c) HT Domain (d) SEC Domain

Figure 2: Comparison of different ranking functions using P@k metric in all domains. Higher values are better.



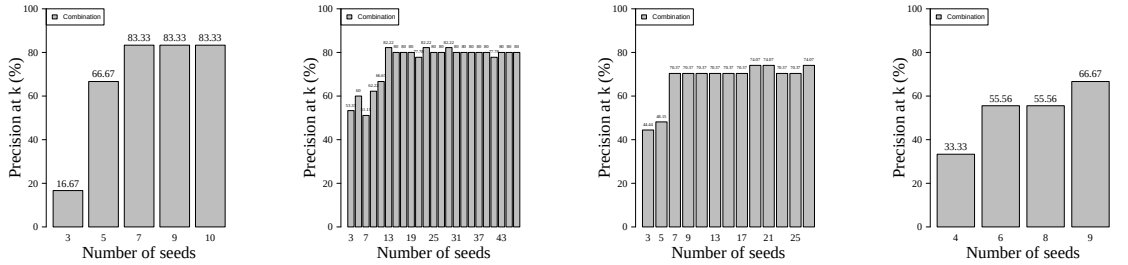
(a) Market Domain (b) Forum Domain (c) HT Domain (d) SEC Domain

Figure 3: Comparison of different ranking functions using mean rank metric in all domains. Lower values are better.



(a) Market Domain (b) Forum Domain (c) HT Domain (d) SEC Domain

Figure 4: Comparison of different ranking functions using median rank metric in all domains. Lower values are better.



(a) Market Domain (b) Forum Domain (c) HT Domain (d) SEC Domain

Figure 5: Experiment showing the influence of the number of seed websites . When the number of seeds increases, the precision also increases until it reaches a plateau. In general, less than 15 examples are required to attain high precision.

Table 1: Statistics of the training data and accuracy

	HT	Market	Forum
# of positive examples	968	68	103
# of negative examples	7117	169	256
Accuracy (5-fold CV)	0.97	0.96	0.84

obtain. Nevertheless, they provide a scalable and consistent way to evaluate the discovery results. Table 1 summarizes the training data and cross-validation accuracy of these classifiers. The accuracy ranges from 84% to 97%, which are sufficiently reliable for evaluating discovered websites . For each method, we stop the algorithm when it retrieves 50,000 web pages . In fact, all the baselines stop before reaching this limit. In the following, we describe baselines and different configurations of DISCO framework used in the evaluation.

We used the following approaches as baselines:

ACHE: ACHE is an adaptive focused crawler described in [4]. We used an open-source implementation available online.[3] This implementation uses an online-learning crawling policy [7] to prioritize the unvisited links. Since ACHE is designed to maximize the number of relevant web pages instead of relevant websites , we limit the number of crawled web pages per website to 10 to ensure it will visit more websites . While a smaller limit would yield higher rate of new websites , it would potentially lead to missing relevant websites that exist but would not have been discovered because some web pages would never been visited. Indeed, our experiment with a smaller limit results in a lower harvest rate and the crawler also stops earlier due to running out of links to crawl.

BIPARTITE: This crawling strategy was proposed by Barbosa et. al. [3] and, similar to ACHE, we set the page-per-site limit to 100.

SEEDFINDER: We implemented the SeedFinder algorithm proposed by Vieira et. al. [31]. SeedFinder uses the keyword search operator to discover new web pages . It then extracts new keywords from the content of the pages classified as relevant by the page classifier to derive new queries using relevance feedback.

We used the following search operators for DISCO:

KEYWORD: DISCO used with the keyword search operator. For a fair comparison, this method uses the same settings as SEEDFINDER (i.e., initial keywords and the number of search results returned by search APIs).

RELATED: DISCO used with the related search operator. Since the objective is to discover websites , we use host name of the input URL to form the search query. For example, given the URL `http://example.com/path.html`, the query is represented as `related:http://example.com`.

FORWARD: DISCO used with the forward crawling operator.

BACKWARD: DISCO used with the backward crawling operator.

BANDIT: DISCO used with the multi-armed bandits algorithm for selecting the discovery operator.

Implementation Details. For KEYWORD and SEEDFINDER methods, we use "gun classified", "gun forum" and "adult escort" as seed keywords for the Market, Forum and HT domains respectively. We selected these keywords since they represent well these domains and we observed highly relevant results by searching these keywords on Google. To obtain new keywords, we tokenize the textual content extracted from the description and keywords tag, then select the most frequent keywords to use in the next search iteration. In each iteration, we select at most 20 new keywords. All methods start with the seeds provided by SMEs (also used in the ranking experiment). We limit the number of results retrieved by keyword search, related search and backlink search to be 50, 50, 5, respectively. We set k in Algorithm 1 to 20. DISCO uses ENSEMBLE as the ranking function because it outperforms other ranking functions and is consistent across multiple domains. We use MOZ API[4] for backlink search, Bing Web Search API[5] for keyword search, and Google Custom Search API[6] for related search.

5.3.1 Comparing Harvest Rates. We use the ground-truth classifiers to classify the discovered web pages . A website is considered as relevant if it contains at least one relevant web page . Figure 6 and Figure 7 show the number of discovered relevant web pages and websites versus the number of discovered web pages over time. Overall, DISCO with the best configuration (BANDIT) obtains at least 300% higher harvest rate compared to other baselines. Also, BANDIT consistently outperforms other configurations in all the three domains in terms of discovered websites . However, if we measure the harvest rate by relevant web pages , ACHE performs closely to the DISCO strategies in the Market and Forum domains. The reason is that ACHE's objective is to maximize the number of relevant web pages by exploiting relevant websites that are already discovered. However, it quickly reaches a plateau when fewer relevant websites are found. Another reason that ACHE performs well in HT domain is that websites in this domain are highly connected. This also explains why backlink method performs better in this domain than the others. SEEDFINDER and BIPARTITE strategies stop early, before reaching 5,000 pages in all the domains, due to running out of links to crawl.

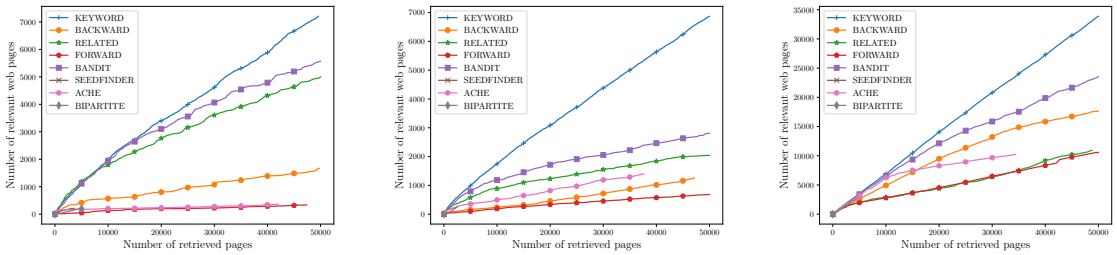
5.3.2 Comparing Coverage. Since it is impractical to obtain the real ground truth for each domain (i.e., all the relevant websites in the domain), we use the union of discovered websites from all discovery methods as an approximation for the ground truth. By doing this, while the estimated coverage can be largely different from the real value, it serves as reasonable relative measure to compare different methods. Figure 8 shows the coverage of all discovery methods in the considered domains. For each method, we compute the percentage of relevant websites that are also discovered by all other methods, which we denominate as *intersection* in the plots. The *complement* region represents the percentage of relevant websites that are discovered only by the corresponding method. The BANDIT strategy consistently attains the largest intersection among all methods. This suggests that each operator searches a different region of the Web, and by adaptively combining them, the BANDIT strategy is able to explore a broader portion of the Web.

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

[4] https://moz.com/products/api

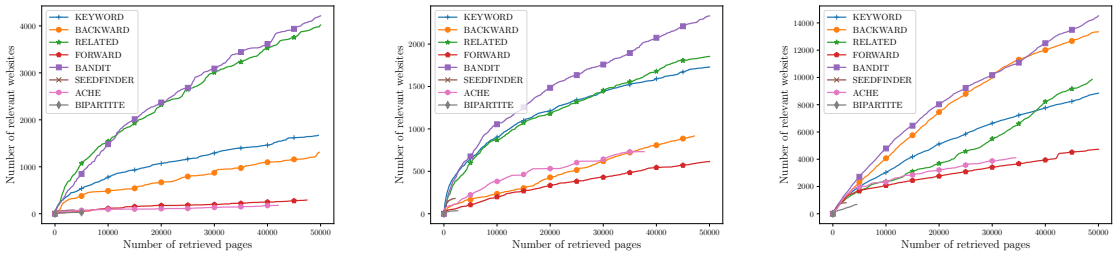
[5] https://bit.ly/2KYAzu2

[6] https://developers.google.com/custom-search/



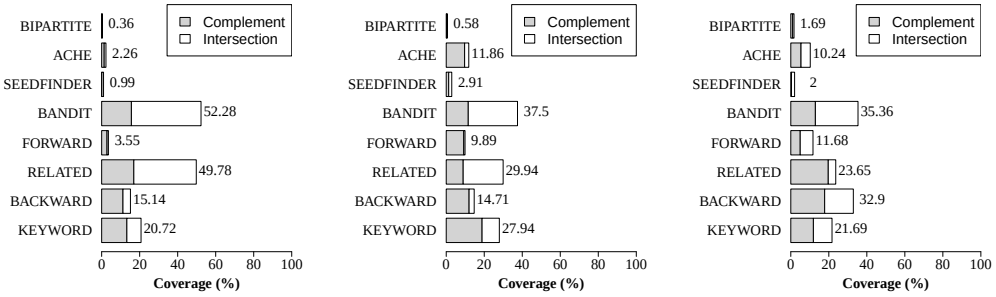
(a) Market Domain (b) Forum Domain (c) HT Domain

Figure 6: Comparison of harvest rate of relevant web pages for different discovery methods.



(a) Market Domain (b) Forum Domain (c) HT Domain

Figure 7: Comparison of harvest rate of relevant websites between for discovery methods.



(a) Market Domain (b) Forum Domain (c) HT Domain

Figure 8: Comparison of coverage for different discovery methods.

6 CONCLUSION

In this paper, we propose DISCO, a new approach that helps bootstrap domain discovery: given a small number of sample sites in a domain, it automatically expands the set. DISCO employs an iterative crawling procedure that combines multiple search operators and, as proxy to select relevant pages, it ranks the search results based on their similarity to the sample sites. We perform extensive experiments using real-world data from different social-good domains and show that our framework is effective and attains significant gains in harvest rate compared with state-of-the-art methods. The experimental results suggest that DISCO can be effective at lowering the burden on users that need to create page classifiers and discover information on the Web at scale. For example, DISCO could be used to both automatically provide additional seed pages for DDT [15] and to rank the results of user-specified search operations.

While DISCO is effective for the domains we experimented with, it has limitations. Our ranking approach is only suitable for the domains for which using n-gram features are effective and the given seed websites are representative for the domains. Additionally, keyword search is very sensitive to the seed keywords, which can be hard to select for some domains. In future work, we would like to explore image-based search for this problem, which can be

potentially applied to image-rich domains, such as weapon ads and human trafficking.

Acknowledgments. This work was partially supported by the DARPA MEMEX and D3M programs, and NSF award CNS-1229185. 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] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47, 2-3 (May 2002), 235–256. https://doi.org/10.1023/A:1013689704352

[2] Luciano Barbosa. 2017. Harvesting Forum Pages from Seed Sites. In *Web Engineering*, Jordi Cabot, Roberto De Virgilio, and Riccardo Torlone (Eds.). Springer International Publishing, 457–468.

[3] Luciano Barbosa, Srinivas Bangalore, and Vivek Kumar Rangarajan Sridhar. 2011. Crawling Back and Forth: Using Back and Out Links to Locate Bilingual Sites. In *Proceedings of the 5th International Joint Conference on Natural Language Processing, IJCNLP*. 429–437.

[4] Luciano Barbosa and Juliana Freire. 2007. An Adaptive Crawler for Locating hidden-Web Entry Points. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*. ACM, New York, NY, USA, 441–450. https://doi.org/10.1145/1242572.1242632

[5] Vanessa Bouché. 2018. Survivor Insights: The Role of Technology in Domestic Minor Sex Trafficking. https://www.wearethorn.org/survivor-insights/.

[6] Soumen Chakrabarti, David A Gibson, and Kevin S McCurley. 1999. Surfing the Web backwards. *Computer Networks* 31, 11 (1999), 1679 – 1693. https://doi.org/10.1016/S1389-1286(99)00042-0

[7] Soumen Chakrabarti, Kunal Punera, and Mallela Subramanyam. 2002. Accelerated Focused Crawling Through Online Relevance Feedback. In *Proceedings of the 11th International Conference on World Wide Web (WWW)*. ACM, New York, NY, USA, 148–159. https://doi.org/10.1145/511446.511466

[8] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. 1999. Focused Crawling: A New Approach to Topic-specific Web Resource Discovery. *Comput. Netw.* 31, 11-16 (May 1999), 1623–1640. https://doi.org/10.1016/S1389-1286(99)00052-3

[9] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, 758–759. https://doi.org/10.1145/1571941.1572114

[10] Anirban Dasgupta, Arpita Ghosh, Ravi Kumar, Christopher Olston, Sandeep Pandey, and Andrew Tomkins. 2007. The Discoverability of the Web. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*. ACM, New York, NY, USA, 421–430. https://doi.org/10.1145/1242572.1242630

[11] Jeffrey Dean and Monika R Henzinger. 1999. Finding related pages in the World Wide Web. *Computer Networks* 31, 11 (1999), 1467 – 1479. https://doi.org/10.1016/S1389-1286(99)00022-5

[12] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. 2000. Focused Crawling Using Context Graphs. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 527–534.

[13] Martin Ester, Hans-Peter Kriegel, and Matthias Schubert. 2004. Accurate and Efficient Crawling for Relevant Websites. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB)*. VLDB Endowment, 396–407.

[14] Zoubin Ghahramani and Katherine A Heller. 2006. Bayesian Sets. In *Advances in Neural Information Processing Systems (NIPS)*, Y. Weiss, B. Schölkopf, and J. C. Platt (Eds.). MIT Press, Cambridge, MA, USA, 435–442.

[15] Yamuna Krishnamurthy, Kien Pham, Aecio Santos, and Juliana Freire. 2016. *Interactive Exploration for Domain Discovery on the Web*. 64–71.

[16] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. 1999. Trawling the Web for Emerging Cyber-communities. In *Proceedings of the Eighth International Conference on World Wide Web (WWW)*. Elsevier North-Holland, Inc., New York, NY, USA, 1481–1493.

[17] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From Word Embeddings To Document Distances. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Francis Bach and David Blei (Eds.), Vol. 37. PMLR, Lille, France, 957–966.

[18] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. 2003. Building Text Classifiers Using Positive and Unlabeled Examples. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, Washington, DC, USA, 179–187.

[19] Markos Markou and Sameer Singh. 2003. Novelty Detection: A Review - Part 1: Statistical Approaches. *Signal Processing* 83 (2003), 2003.

[20] Robert Meusel, Peter Mika, and Roi Blanco. 2014. Focused Crawling for Structured Data. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*. ACM, New York, NY, USA, 1039–1048. https://doi.org/10.1145/2661829.2661902

[21] Tsuyoshi Murata. 2001. Finding Related Web Pages Based on Connectivity Information from a Search Engine. In *Poster Proceedings of 10th International Conference on World Wide Web (WWW)*. 18–19.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[23] Kien Pham, Aécio Santos, and Juliana Freire. 2018. Learning to Discover Domain-Specific Web Content. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, New York, NY, USA, 432–440. https://doi.org/10.1145/3159652.3159724

[24] 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. *Proc. VLDB Endow.* 8, 13 (Sept. 2015), 2194–2205. https://doi.org/10.14778/2831360.2831372

[25] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (April 2009), 333–389. https://doi.org/10.1561/1500000019

[26] Jakob Rogstadius, Maja Vukovic, C. A. Teixeira, Vassilis Kostakos, Evangelos Karapanos, and Jim Laredo. 2013. CrisisTracker: Crowdsourced social media curation for disaster awareness. *IBM Journal of Research and Development* 57, 5 (2013), 4:1–4:13. https://doi.org/10.1147/JRD.2013.2260692

[27] Aécio Santos, Bruno Pasini, and Juliana Freire. 2016. A First Study on Temporal Dynamics of Topics on the Web. In *Proceedings of the 25th International Conference Companion on World Wide Web*. 849–854. https://doi.org/10.1145/2872518.2889291

[28] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. 1999. Support Vector Method for Novelty Detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, USA, 582–588.

[29] Chengcheng Shao, Giovanni Luca Ciampaglia, Alessandro Flammini, and Filippo Menczer. 2016. Hoaxy: A Platform for Tracking Online Misinformation. In *Proceedings of the 25th International Conference Companion on World Wide Web (WWW)*. 745–750. https://doi.org/10.1145/2872518.2890098

[30] Matt Stroud. 2015. FBI data show thousands of gun sales beat checks. https://docs.house.gov/meetings/JU/JU00/20151117/104114/HHRG-114-JU00-20151117-SD007.pdf.

[31] Karane Vieira, Luciano Barbosa, Altigran Soares da Silva, Juliana Freire, and Edleno Moura. 2016. Finding seeds to bootstrap focused crawlers. *World Wide Web* 19, 3 (2016), 449–474. https://doi.org/10.1007/s11280-015-0331-7

[32] Xuezhi Wang, Cong Yu, Simon Baumgartner, and Flip Korn. 2018. Relevant Document Discovery for Fact-Checking Articles. In *Companion Proceedings of the The Web Conference 2018*. 525–533. https://doi.org/10.1145/3184558.3188723

[33] Lingfei Wu, Ian En-Hsu Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J. Witbrock. 2018. Word Mover's Embedding: From Word2Vec to Document Embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 4524–4534.

[34] Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and Resolution of Rumours in Social Media: A Survey. *ACM Comput. Surv.* 51, 2, Article 32 (Feb. 2018), 36 pages. https://doi.org/10.1145/3161603