IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 30, 2022, accepted April 15, 2022, date of publication April 20, 2022, date of current version April 27, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3168977

Real-Time Focused Extraction of Social Media Users

RODRIGO MARTÍNEZ-CASTAÑO, DAVID E. LOSADA, AND JUAN C. PICHEL
CiTIUS, Universidade de Santiago de Compostela, 15782 Santiago de Compostela, Spain

Corresponding author: Juan C. Pichel (juancarlos.pichel@usc.es)

This work was supported in part by the Ministerio de Ciencia e Innovación (MICINN) under Grant RTI2018-093336-B-C21 and Grant PLEC2021-007662; in part by Xunta de Galicia under Grant ED431G/08, Grant ED431G-2019/04, Grant ED431C 2018/19, and Grant ED431F 2020/08; and in part by the European Regional Development Fund (ERDF).

ABSTRACT In this paper, we explore a real-time automation challenge: the problem of focused extraction of Social Media users. This challenge can be seen as a special form of focused crawling where the main target is to detect users with certain patterns. Given a specific user profile, the task consists of rapidly ingesting Social Media data and early detecting target users. This is a real-time intelligent automation task that has numerous applications in domains such as safety, health or marketing. The volume and dynamics of Social Media contents demand efficient real-time solutions able to predict which users are worth to explore. To meet this aim, we propose and evaluate several methods that effectively allow us to harvest relevant users. Even with little contextual information (e.g., a single user submission), our methods quickly focus on the most promising users. We also developed a distributed microservice architecture that supports real-time parallel extraction of Social Media users. This modular architecture scales up in clusters of computers and it can be easily adapted for user extraction in multiple domains and Social Media sources. Our experiments suggest that some of the proposed prioritisation methods, which work with minimal user context, are effective at rapidly focusing on the most relevant users. These methods perform satisfactorily with huge volumes of users and interactions and lead to harvest ratios 2 to 9 times higher than those achieved by random prioritisation.

INDEX TERMS Big data, distributed systems, focused user extraction, supervised learning, information retrieval, real-time processing, social media.

I. INTRODUCTION

Focused Crawling techniques are oriented to extracting web pages that satisfy a given property or topic of interest. This is typically supported by supervised learning technology (e.g., text classifiers) that help to decide what links are worth to explore. Nowadays, it is feasible to run crawling tools that efficiently extract *on topic* contents from massive repositories such as the web or certain Social Media (SM) sources. Focused Crawling was first introduced in [1] and coined in [2]. It has been intensely discussed in the literature [3], [4] and it plays a fundamental role, for example, in building vertical search engines [5]. Nevertheless, to the best of our knowledge, the real-time extraction of SM users that are relevant to a target topic has received little attention. In this paper, we make a first attempt to fill this gap. Many studies in the literature have proposed different alternatives to crawl or extract SM contents. However, the notion of user is often ignored and most existing methods do not try to anticipate which users are worth to explore. We claim that we need real-time technology that rapidly ingests SM contents and, given little context (e.g., the last user's SM submission), moves quickly towards relevant users. We envisage intelligent forms of user crawling that aggregate the history of the target users with little delay and do not overload the system with users that are likely non-relevant. As reported in our Related Work section, some user-oriented SM methods exist but they are oriented to extract samples of users (rather than massively analysing the entire SM site in real-time) or do not implement any form of user prioritisation.

Given a certain SM platform, we formally define the task of *Focused User Extraction* (FUE) and investigate the efficiency and effectiveness issues involved in extracting target users. FUE has important applications in numerous domains. The rapid growth of SM platforms and their content production dynamics demand Big Data solutions able to (1) efficiently

The associate editor coordinating the review of this manuscript and approving it for publication was Massimo Cafaro.

ingest contents in real-time, (2) quickly react to changes, and (3) intelligently anticipate which users are worth to explore.

This task requires methods able to perform user extraction from large repositories of SM data. By smartly prioritising SM users who are of potential interest, we can early identify target individuals and, thus, support a number of risk assessment tasks such as grooming detection, criminal recruitment or early identification of psychological problems. This requires not only large-scale technologies but also smart predictive components that continuously rank users based on real-time evidence (e.g., last SM post).

In popular SM sources, we constantly obtain new pieces of evidence. Each new SM post is typically short and gives only a partial description and context of the author of the post. The user crawling process needs to evaluate this real-time evidence and, for example, decide whether or not to crawl the entire user history. This prediction task resembles when focused web crawlers see a link to a new page and have to decide whether or not to download the linked page. However, in FUE, the full exploration of a given user often requires tens or hundreds of requests to the SM servers and, thus, effective prioritisation of users becomes crucial.

Taking only into account the *partial user context* allows us to optimise the existing resources and extract the maximum number of relevant users per unit of time. In this paper, we propose several user prioritisation methods that guide this exploration process and we study which of these methods lead to high-speed FUE.

In order to extract and process huge amounts of data in real time, it is necessary to design and implement an adequate architecture that can scale up horizontally in a cluster of computers. In this regard, we contribute here by developing an adaptable distributed crawling architecture based on Catenae[1] [10]–[12], Kafka [13] and Docker [14]. The architecture was developed to support FUE but it can be easily adapted to other scenarios such as real-time query-based filtering, topic extraction or summarisation.

Summing up, the paper describes the practical application of user prioritisation methods in real-time intelligent extraction of SM users. The main contributions of this paper are:

- A formal definition of the Focused User Extraction (FUE) task and a systematic analysis of the main effectiveness and efficiency issues involved.
- Several heuristic-based methods that consistently increase the harvest ratio of target users and a comprehensive study of their relative importance under real-time SM experiments. To the best of our knowledge, this is the first study that analyses the ability of a set of user-related variables in user crawling prioritisation.
- A modular SM crawler that performs real-time extraction and exploration of SM communities and user profiles. An experimental evaluation performed on Reddit demonstrates the real-time processing capabilities of the crawler. The competing methods are benchmarked against each other under real-time experiments where the methods run in parallel. In this way, the relative benefits of each prioritisation method are not affected by seasonal factors.
- A Docker-based distributed architecture that can be easily deployed and scales by increasing the number of instances of each module in a cluster of computers (publicly available[2] under a Free Software license).

The paper is structured as follows. Section II describes the methodology followed by our crawler to extract SM contents and rank users following different strategies. Section III discusses the architecture of the system, its modules, connections and the scalability of the proposed solution. Sections IV and V illustrate how different user extraction methods run and evolve under real-time experiments. In Section VI the adaptability of our approach to other Social Media is discussed. Section VII discusses related work and, finally, Section VII contains the main conclusions of this study.

II. METHODOLOGY

A. FOCUSED USER EXTRACTION

Given a profile of interest that defines target users (target user profile), the relevance of each newly detected user has to be estimated. If possible, this needs to be done using minimal contextual information. For example, crawling the entire history of user posts, even if possible, introduces a significant overhead on the user crawling process. And many of those users might be non-relevant. An effective user crawler must process light volumes of user data (e.g., last post) and only make a full extraction of user's data for those users that seem to be on-target. This represents a two-stage process in which SM web pages are constantly explored to extract candidate usernames (e.g., from web pages with the newest threads of user posts) and the user crawler process predicts which candidate users are worth to explore (i.e., which users are fully explored).

The target user profile can be represented with a driving query or with a set of examples (users who fulfil the target profile and which can be used to build a user classifier). Given a Social Media platform, the aim of the task is to extract as many target users per unit of time as possible. Effective algorithms that solve this problem should somehow prioritise those users that are presumably on target and optimise the use of available resources.

There are two main useful metrics to evaluate the performance of this task. First, the harvest ratio (*HR*), or precision, as the number of relevant users extracted divided by the total number of extracted users. A user is deemed as relevant if it is cataloged as so with high confidence by a reference classifier. This reference classifier represents the target profile of users. This means that the entire user profile –concatenation of all their posts–, when passed to the classifier, exceeds a certain threshold of the classifier (prediction probability higher than 0.9 in the case of the logistic regression classifier). A second

[1] https://github.com/catenae

[2] https://github.com/blind-sniper

performance metric is the extraction speed expressed as the number of extracted users per unit of time.

There is a huge number of active users on Social Media. In realistic scenarios, where hardware is limited, extracting and processing everything would be unfeasible due to the massive volume of contents produced in real-time. Furthermore, there are usually some access restrictions imposed by the source (e.g., `robots.txt` constraints). As a consequence, lightweight exploration needs to be performed in order to extract SM users. A crucial issue is how to estimate user's relevance from the little context available. Note also that, even if we have extensive hardware resources and computational power at our disposal, a random or arbitrary exploration of SM users is not pertinent. Such näive extraction of users slows down the identification of relevant users and, thus, cannot support the early identification of target users.

Typically, when exploring a Social Media site looking for new users, candidate usernames can be found on pages that list users' descriptions or on pages that contain a thread of user's posts (e.g., the entry page of a given user community or forum). The available posts often include a title or description, possibly a snippet, information about the author and other relevant social properties (such as the number of replies, votes, etc.). These types of listings serve as seed pages for extracting candidate users. Among the available evidence that can be used to guide the crawler, we have elements such as the title or the text of the last user's post, the date and time of publication, the number of comments, the score or the number of times a post has been liked, the number of times the post has been shared or re-posted, etc. In this paper, we study which of the available pieces of evidence are predictive of relevance in order to guide the crawling process.

B. FOCUSED USER EXTRACTION ON REDDIT

Reddit is a Web platform where users submit content (posts) such as text, images or links and other users can comment and vote for or against them. Comments, at the same time, can also be commented and voted. The platform is subdivided into communities (*subreddits*) focused on specific topics. It is currently ranked in Alexa [15] as the 7th website with more traffic in the United States and 19th globally. The number of average daily active users is higher than 52 million, and there are more than 100,000 active communities [16].

We have chosen Reddit as our reference platform for experimenting with FUE algorithms. This decision was based on several criteria. First, it is one of the three largest Social Media platforms with increasing popularity. Second, the diversity of users and communities make Reddit a perfect place for user mining. Third, its terms of service state their open philosophy and willingness to support external applications or services connected to Reddit. This allows us to easily conduct crawling research on this platform. Furthermore, the lessons learned from our study can be potentially transferred to other social networks.

The crawling process is composed of two main steps. First, the crawler has to choose the next subreddit from which to explore the latest posts and comments. This subreddit exploration step constantly detects new usernames that are candidates to be fully explored. The second step consists of ranking the candidate users based on the available evidence. To meet this aim, a user ranking is built (using a number of user prioritisation methods) and the top user is selected and fully explored (all his/her submissions are collected).

The first stage (subreddit exploration) works with a *subreddit frontier* that stores the subreddit names that have been found so far. Initially, the main frontpage of Reddit is used as a seed to extract some subreddit names that are stored in the frontier. This frontier grows during the crawling process (any web page retrieved from Reddit potentially contains references to unseen subreddits) and the selection of the next subreddit is done randomly from the subreddits available in the frontier.

The second stage consists of choosing the next user to be analysed in depth. When exploring subreddits, new users are discovered and ranked according to different methods (see below). After a user is fully explored (all his/her submissions are retrieved), he/she is marked as processed on the list of known users and, thus, he/she will not be processed again.

Algorithm 1 describes our focused user crawler in pseudocode. There are four main procedures. First, during the **Initialisation**, the Reddit's front page is downloaded. This page is a list of the most recent posts on the platform from any subreddit. The names of the subreddits are collected and used as seeds. In addition, the features of the posts (for example, title or number of comments) are extracted. These features are used to update the ranking of candidate users (according to different methods, as we will explain below). After obtaining the seeds, three tasks are launched in parallel:

- **Community Exploration**. A subreddit is picked at random from the list of known communities. The front page of the selected subreddit is visited, extracting the latest posts and their features. The user ranking is updated by aggregating the new features with the existing ones.
- **User Extraction**. The top user from the user ranking is selected. All the available submissions and comments posted by this user are extracted from the SM platform. Note that this requires several calls to Reddit and, thus, the focused user crawler aims at extracting only those users who are likely relevant.
- **Evaluation**. All the extracted users are queued to evaluate their relevance. To meet this aim, we employ a user classifier. This classification tool analyses the full user information and determines whether or not the user is on-topic. In this way, this evaluation component assesses whether or not the extraction of the entire user history was worthwhile. This evaluation strategy follows [2], [17], where the relevance of the extracted pages is evaluated with a classifier. The main reason to estimate relevance in this way is that there are practical impediments to perform a manual analysis of thousands of texts from hundreds of thousands of users.

Algorithm 1 Pseudo-Code of the Focused User Crawler for Reddit

```
1:  procedure Initialisation
2:      if length of subreddit_list == 0 then
3:          html_code = load_url("https://reddit.com/r/all/new/")
4:          subreddits = extract_subreddits(html_code)
5:          for each subreddit in subreddits do
6:              if subreddit not in subreddit_frontier then
7:                  subreddit_frontier.add(subreddit)
8:          posts_features = extract_posts_features(html_code)
9:          for each post_features in posts_features do
10:             users_ranking.update_user_features(post_features)
11:         do in parallel
12:             Community Exploration()
13:             User Extraction()
14:             Evaluation()
15:
16: procedure Community Exploration
17:     while true do
18:         selected_subreddit = pick_at_random(subreddit_frontier)
19:         html_code = load_url("https://reddit.com/r/" + selected_subreddit + "/new/")
20:         posts_features = extract_posts_features(html_code)
21:         for each post_feature in posts_features do
22:             users_ranking.update_user_features(post_features)
23:
24: procedure User Extraction
25:     while true do
26:         selected_user = users_ranking.get_first_user()
27:         html_code = load_url("https://reddit.com/user/" + selected_user)
28:         user_posts = extract_posts(html_code)
29:         for each user_post in user_posts do
30:             subreddit_frontier.add(user_post["subreddit"])
31:         users_posts_to_evaluate.add(user_posts)
32:
33: procedure Evaluation
34:     while true do
35:         user_posts = users_posts_to_evaluate.get_first()
36:         probability = classification_model.predict(user_posts)
37:         if probability > THRESHOLD then
38:             positive_users.add(user_posts)
39:         else
40:             negative_users.add(user_posts)
```

We now make a brief discussion on the computational complexity involved in the procedures described above:

- Community Exploration: The community exploration procedure is quite lightweight. For each subreddit, it gets the subreddit's page of new contents, extracts some metadata from the available posts and updates some user's features. The time complexity of this procedure is $\mathcal{O}(s)$, where s is the number of communities or subreddits. With proper data storage structures, the cost of choosing a random subreddit from the subreddit frontier (line 18) and updating user's features (lines 21-22) do not add further complexity to the process.
- User Extraction: The computational load of the user extraction procedure grows linearly with the number of users and, for each user, the user's posts available at the user's frontpage need to be crawled and added to the corresponding data structures. This means that the time complexity of this procedure is $\mathcal{O}(u \cdot p_{avg})$, where u is the number of users and p_{avg} is the mean number of posts published in the user's frontpage. Observe that, having proper data storage structures, the costs of extracting the next user from the available list of users (line 26), incorporating the post's subreddit to the subreddit frontier (line 30), and incorporating the user's posts to the

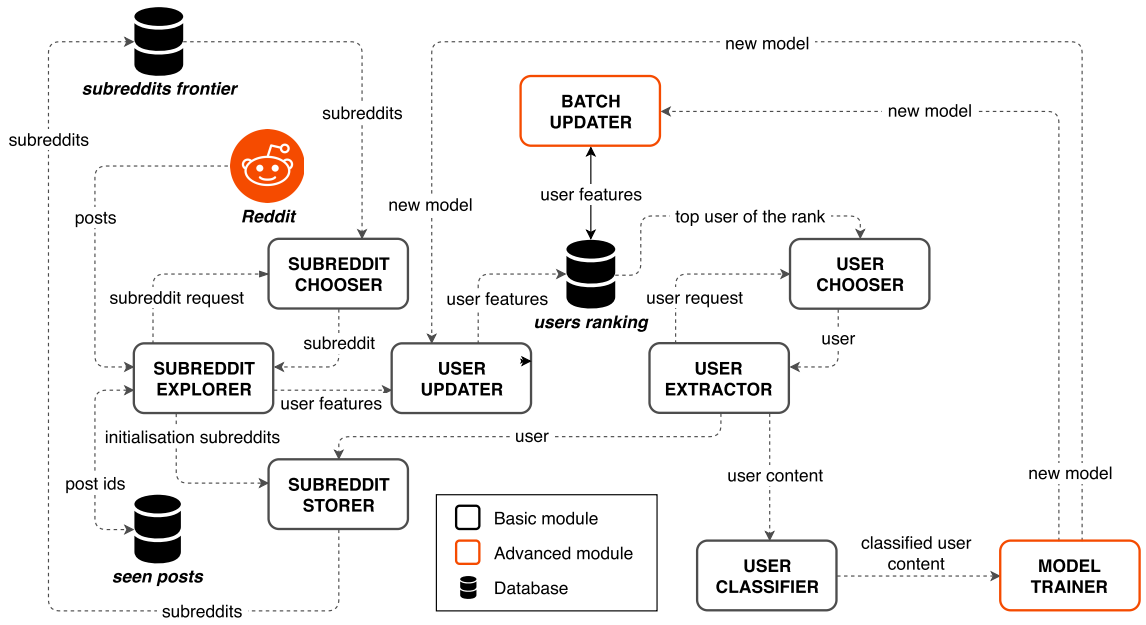

FIGURE 1. Simplified architecture diagram of the real-time focused user crawler for Reddit.

evaluation's data structure (line 31) can be considered negligible.

- Evaluation: This procedure essentially consists of passing users through the classification model. The construction of the classifier from (external) training data is not part of the crawling process (the crawling process simply reads the trained classifier and the text vectorizer from disk). The text vectorizer processes the user's posts and produces a numerical representation that is then passed to the classifier. We work with a lightweight classifier (Logistic regression, see section IV-F) whose prediction cost grows linearly with the number of features. The number of features depends on the number of unique words in the training corpus and it is often larger than the number of words in the user's posts. As a consequence, the time complexity essentially depends on the number of users and the number of classification features: $\mathcal{O}(u \cdot ft)$, where u is the number of users and ft is the number of classification features, which is bounded the size of the vocabulary of the training collection. Again, with proper storage structures, the cost of extracting the first user (line 35) and incorporating the user to the positive or negative lists (lines 38-40) is insignificant.

  In terms of space complexity, the size of subreddit frontier is bounded by the number of available subreddits (less than 200k) and, in any case, we only need to store a string for each subreddit (the name). The users' features and posts demand more space but the crawling process, if needed, can remove the features and posts of the users who have been already passed to the classifier.

  Note also that community exploration, user extraction and evaluation run in parallel (lines 11-14) and, furthermore, the operation of each of these three procedures can be easily parallelized. As argued in the next section, depending on the load of each procedure, we can add more computational resources (e.g., to do parallel extraction of multiple users).

III. REAL-TIME CRAWLING ARCHITECTURE

The focused user crawler is composed of multiple microservices, designed to facilitate the scalability of the system when running on a cluster of computers. Modules are interconnected to form an execution graph built on Catenae [10]–[12], a Python library for easy development of scalable stream execution graphs. Catenae uses the Kafka message broker to interconnect different microservices. Catenae-based systems can scale up horizontally by increasing the number of instances of any microservice without further configuration.

Our architecture permits the deployment of this focused crawler in infrastructures where dynamic resources can be configured, such as computing clouds (e.g., Amazon Web Services, Microsoft Azure, Google Cloud Platform, Alibaba Cloud). In addition, when few hardware resources are available, the crawler can scale up or down without interrupting its execution. Due to this distributed architecture, the focused user crawler can be simply scaled by augmenting the number of instances (e.g., if there is a bottleneck in one of the modules). The main modules and connections of the system are shown in Figure 1. The modules of the focused user crawler for Reddit are explained below:

- **Subreddit Explorer**. At this stage, subreddits are requested to the Subreddit Chooser through RPC (Remote Procedure Call). When a subreddit is assigned, this module explores it by extracting the last posts and

comments. For each submission, it extracts the available features and sends them to a User Updater instance (for example, if the crawler prioritises users based on the number of votes of the user submissions, then any new update on the user's votes is handled by the User Updater). When initialising the system, the Subreddit Explorer module extracts the subreddits of the texts available under `/all`, a pseudo-community that holds together the posts from all communities. Note that, at the beginning of the process, there is no knowledge on which subreddits are more promising (potentially contain more on-topic users) and, thus, we need to start from a general page of the platform.

- **Subreddit Chooser**. This module is responsible for selecting a new subreddit to be sent to the subreddit explorers. In this work, we experiment with a random selection of subreddits but the platform can easily implement more sophisticated selection methods.
- **User Updater**. The User Updater is responsible for updating the user features that guide the focused crawling process. For example, it averages, for every user, the number of comments in his/her posts. A full description of user features and related requirements to the User Updater module are given in Section IV.
- **User Extractor**. The User Extractor instances request users to the User Chooser module and extract the maximum possible number of user texts.[3] The extracted content is sent to the User Classifier, which evaluates the performance. Additionally, the subreddits discovered during this process are sent to the Subreddit Storer.
- **User Chooser**. This module handles the selection of new users to be processed (according to the user prioritisation process that guides the focused crawler). User extractors invoke the User Chooser module through RPC once they finish the extraction of a user. Since the new users are requested only when needed, the user selection is coherent with the updated ranking state in every moment. The User Chooser assigns a unique user for each user extractor. It also marks the selected users as processed in order to avoid repeated processing.
- **User Classifier**. The main goal of this module is to evaluate the performance of the focused user crawler. It receives all the available texts for every extracted user, classifies the user and stores the probability provided by the user classifier.
- **Subreddit Storer**. In this phase, the previously unseen subreddit names are stored in the database. It receives new subreddits from a Subreddit Explorer instance during the initialisation and from the User Extractor instances.
- **Stats Dumper**. This module is connected to all the other modules and stores events with information related to the real-time execution. Every event is timestamped, so the behaviour of the system in several dimensions can be easily analysed during or after the execution. This module is not represented in Figure 1 due to its number of interconnections.
- **Model Trainer**. This module is responsible for training the user classifier. In our initial experiments, the user classifier is built once (from external data) and remains unchanged thereafter.
- **Batch Updater**. This module is needed for one of the advanced prioritisation methods detailed in Section V. It updates user-related data after learning new user classification models.

MongoDB [18] is used as a storage system and for managing the rankings. Aerospike [19], a memory-based key-value store, is used to avoid processing repeated posts during the crawling process.

IV. BASIC PRIORITISATION METHODS

In our attempt to early identify target users during crawling, we have defined several methods to guide the FUE process. Standard focused crawlers employ a number of strategies when, for example, they get a new link and they need to predict whether it is worth to download the linked web page. In FUE, the available evidence is different and, thus, we need to design new strategies for prioritising the users that are found. These methods work with a number of features that are available when browsing the SM website. The methods considered are:

A. RANDOM

A random (RND) selection of the next user. This is a näive baseline used for comparison against more sophisticated methods.

B. AVERAGE SCORE OF USER'S TEXTS

In SM platforms, users' interactions are often scored, voted or liked. The *popularity* of user's posts and comments might be a valuable clue to guide FUE. For example, a focused user extraction process might be interested in early identifying those users emitting highly influential contents. In Reddit, the score of a post reflects the utility and quality of the content within its context (the subreddit). The interaction of the users determines the total score of a post (users can add, *upvote*, or subtract, *downvote*, a point to each post or comment). The score of a Reddit's post or comment is calculated by summing the positive and negative votes. Depending on the focus of the crawler, this feature might be indicative of target users (e.g., a focused extraction of offensive users might benefit from the existence of many negative votes). During the crawling process, this average score is accumulated for each user. Note that this is computed from the user texts seen so far (web pages downloaded) and, thus, it is usually an incomplete view of the overall score for this user.

Let U be the set of candidate users and $S_u = \{S_{u_{t_1}}, S_{u_{t_2}}, \ldots\}$ be the set of scores for each observed text of a

[3] In our experiments, we respect Reddit's `robots.txt` and, thus, each request gets a maximum of two hundred texts (posts and comments).

given user u. Let $\overline{S}_u$ be the average score of the known texts of a given user. We propose two different ways to determine the chosen user (c_{user}). These two variants select the next user based on the highest or lowest average score, respectively. The first approach builds (and constantly updates) a ranking of users by decreasing average score and extracts the top user. This will be referred to as Highest Average Score (HAS). Similarly, the Lowest Average Score (LAS) builds a ranking of users by increasing average score.

$$c_{user} = \arg\max_{u \in U} \overline{S}_u \quad (1)$$

$$c_{user} = \arg\min_{u \in U} \overline{S}_u \quad (2)$$

C. AVERAGE NUMBER OF COMMENTS IN USER'S TEXTS

This feature also weights users based on the impact of their texts. In this case, we employ the average number of comments associated to the posts of a given user. Users with long threads of comments to their posts might be particularly relevant for a given FUE process. Let C_u be the number of comments for each observed text of a given user u. Let $\overline{C}_u$ be the average number of comments of the known texts of a given user u.

Again, we define two approaches that build descending and ascending rankings with the average number of comments of every user: Highest Average Comments (HAC) and Lowest Average Comments (LAC), respectively. The ranking is updated every time new data is available. When the system needs to extract a new user, the top user of the ranking is selected.

$$c_{user} = \arg\max_{u \in U} \overline{C}_u \quad (3)$$

$$c_{user} = \arg\min_{u \in U} \overline{C}_u \quad (4)$$

D. TIME-BASED USER SELECTION

The post time of user's texts might also be useful for discovering target users. For example, a FUE process aiming at extracting users showing signs of psychological problems (e.g., depression) could benefit from time-related trends (e.g., users might make more submissions during the night because sleeping problems is one of the symptoms of depression). To encode this feature, we extract the hour when each observed text was posted, we average these hours for each user and we compute how close the average is to four selected times (0h, 6h, 12h and 18h). Let $T = \{0, 6, 12, 18\}$ be the set of selected hours. Let H_u be the set of hours for the seen texts of a given user u. Let $\overline{H}_u$ be the average hour when the seen texts of a given user were posted. We have defined four time-related prioritisation methods: Near 00:00 UTC±00:00 ($N00$) gives more weight to users that post near midnight, Near 06:00 UTC±00:00 ($N06$) gives more weight to users that post near 6 in the morning, and so on.

$$c_{user} = \arg\min_{u \in U} min(|t - \overline{H}_u|, 24 - |t - \overline{H}_u|), \ where\, t \in T \quad (5)$$

E. CLASSIFIER-BASED PRIORITISATION

A focused crawler is often guided by a driving query or classifier. Given the text observed from each user, a natural way to guide the FUE process is either to compute the matching between the driving query and the user's text or to pass the user's text to the classifier that guides FUE. At any point, the user's texts available to the crawler are a subset of the user's history of posts but, still, this partial representation of the user might be highly valuable to early identify target users. For example, the first time that the crawler sees a user, it probably reads a single comment or post from the user. This resembles when a focused crawling of web pages sees the first reference (link) to a new web page. In such a case, a single anchor text and the associated URL are the only pieces of evidence available but the focused crawler can still estimate topicality with respect to the target profile. In FUE, we can maintain (and constantly augment) a textual representation of each user. This consists of the concatenation of the titles and comments' bodies of the posts observed for each user. Let D be the set of incremental user's representations. Given D, we can pass it to the classifier and obtain a probability estimation of the user being on-target, $P_u(D)$.

We define two approaches to select target users: Highest Classification Probability (HCP) builds a ranking of users by decreasing probability, whereas Lowest Classification Probability (LCP) builds a ranking of users by increasing probability. LCP is supposed to perform poorly since this prioritisation makes little sense. However, LCP serves as a safe check in our empirical study.

$$c_{user} = \arg\max_{u \in U} P_u(D) \quad (6)$$

$$c_{user} = \arg\min_{u \in U} P_u(D) \quad (7)$$

Table 1 summarises all prioritisation methods described above.

F. EXPERIMENTAL RESULTS

In order to make fair comparisons among the proposed methods, we have designed an experimental framework where all crawling variants run in parallel. In this way, we can fairly evaluate their relative merits on identifying target users and we avoid undesirable biases from seasonal effects that might affect the comparison.

The server where all the experiments in the paper were conducted has the following characteristics:

- CPU: 2 x Intel® Xeon® CPU E5-2630 v4 @ 2.20 GHz - 20 cores (40 threads)
- Memory: 12 x Hynix 32 GiB DIMM DDR4 @ 2400 MHz (384 GiB)
- Disk: Toshiba MD04ACA400 4 TB @ 7,200 RPM, 64 MB cache
- Internet bandwidth: ∼400/200 Mbps (download/upload)

The first experiment lasted seven days. During this period, the total number of unique extracted users was higher than 110, 000 (655 per hour) for each prioritisation method

TABLE 1. Basic prioritisation methods.

Acronym	Category	Description
RND	Random	Random Selection of the next user
HAS	Score (Pos-Neg votes)	Selects user with the Highest Average Score
LAS	Score (Pos-Neg votes)	Selects user with the Lowest Average Score
HAC	Comments	Selects user with the Highest Average # of Comments
LAC	Comments	Selects user with the Lowest Average # of Comments
N00/N06/N12/N18	Time-based	Select user whose avg submission time of day is the closest to 0h, 6h, 12h and 18h
HCP	Classifier-based	Selects user whose submissions yield the Highest Classifier Probability
LCP	Classifier-based	Selects user whose submissions yield the Lowest Classifier Probability

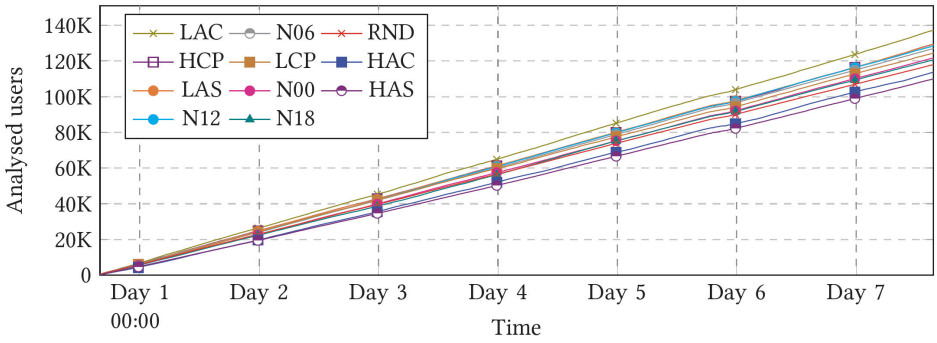

FIGURE 2. Number of extracted users by each focused user crawling method.

(see Figure 2). Note that the extraction performance can be highly scaled, while maintaining the politeness of the crawler, by mainly launching more instances of the User Extractor module, which is the main bottleneck.

The crawling variants HAC and HAS are slower at extracting users. This is due to the way in which users are prioritised. These two variants extract users that have a high number of comments or points. These users tend to be highly active on the platform and, thus, when the crawler extracts them, the retrieval of all their submissions is costly.

Let us now evaluate the ability of these prioritisation strategies in identifying target users. To meet this aim, we use a user classifier built from external training data. More specifically, we built a binary classifier from a sample of Reddit users that predicts if a given user is likely to have signs of depression. We worked with a collection on depression and natural language use [20]. The classifier is a Logistic Regression model with L1 regularisation implemented in Python with *scikit-learn*. It was built with a training set of 486 users (83 positives, 403 negatives) where users were represented with a single document consisting of the concatenation of all their writings. In order to decide if a user is on-target, we established a threshold equal to 0.9 since previous experiments demonstrated that this setting leads to high precision [20].

Figures 3 and 4 plot the harvest ratio of different prioritisation strategies and, thus, it helps to compare the relative merits of the proposed metrics in early identifying the target users. To facilitate readability, the methods have been separated into two independent plots but the random strategy is shown in both graphs. The highest harvest ratio is achieved by HCP: 2.8 times better than the second best (N06) and 4.9 times better than RND during the first day. On the last day these numbers decay to 1.4 and 1.8, respectively.

After the first day, four of the proposals behave better than random. This number increases to five after the third day. At this point, many strategies behave similarly since most of them have already explored a high number of users. The most promising strategies are HCP, N06, HAC and N00 since they perform considerably better than random. On the other hand, HAS, N12, N18, LAC, LAS and LCP can be discarded.

Observe that HCP effectively exploits the little pieces of textual evidence available. These small extracts seem to be indicative of user relevance. Note also that none of the other good performers employ the texts written by users to prioritise them.

V. ADVANCED PRIORITISATION METHODS

Given the results discussed above, we designed three new FUE methods. The first is based on re-training the classifier that guides the HCP crawler (from new tagged users obtained with *pseudo-training data*). A second method performs a fusion of two user rankings (HCP and HAC). Finally, the third method combines HCP, N06 and HAC in a hierarchical way. This section presents these new methods and discusses the associated changes in the FUE architecture.

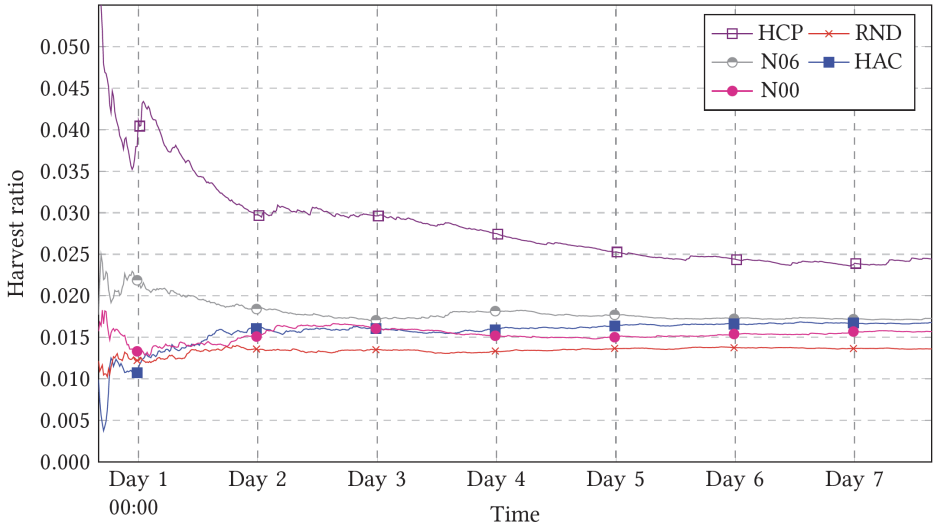

FIGURE 3. Harvest ratio of some focused user crawling methods.

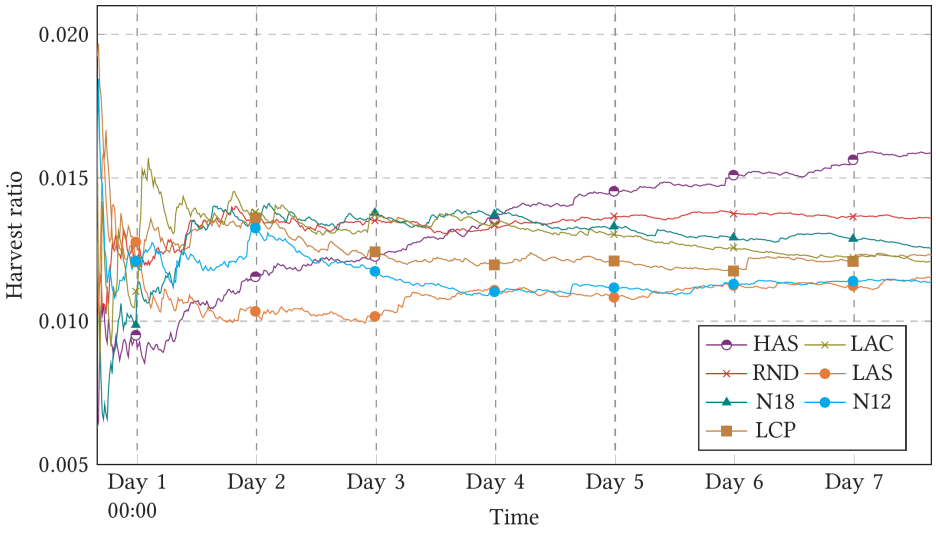

FIGURE 4. Harvest ratio of some (discarded) focused user crawling methods.

HCP With Pseudo-Training: With HCP, the crawling system makes two types of user classifications. The *evaluation classifier* (see Section II) is the core evaluation tool of the FUE process and assesses whether or not user extraction was effective. The *prediction classifier* categorises the users based on the partial information available (e.g., last post in a recently crawled page), yielding a confidence score that is used for user prioritisation. With the standard HCP method discussed above, these two classifiers are the same and their classification model is built once (from the training set) and never changes thereafter. However, the classification model of the predictive classifier could be updated as we extract and process users. More specifically, those users that are chosen, fully explored, and classified by the evaluation classifier can be incorporated as pseudo-training examples. The main idea of HCP with pseudo-training consists of updating the prediction classifier based on augmenting the training examples with new examples and their *pseudo-labels*. Every five minutes, a new prediction classifier is built from the original training set plus the new *pseudo-examples* available. We experimented with three variants. With HCP-0.9-0.1 only those users classified with a probability higher than 0.9 or lower than 0.1 are incorporated (as positive or negative, respectively).[4] With HCP-0.5-0.5 we employ the standard threshold of the classifier (0.5) and, thus, any user classified above or below the threshold is incorporated into the training set as positive or negative, respectively. We also

[4]Note that we work with Logistic Regression classifiers and, thus, we have access to probabilities associated to the estimations.

TABLE 2. Advanced prioritisation methods.

Acronym	Category	Description
HCP-u-d	HCP+pseudo-training	HCP variant that regularly updates the classifier with new pseudo-training instances (users classified with probability higher than u are considered as positive, while users with probability lower than d are considered as negative).
HCP-u	HCP+pseudo-training	HCP variant that regularly updates the classifier with new pseudo-training instances (users classified with probability higher than u are considered as positive; pseudo-negatives examples are not incorporated into the training sets).
HCP-N06-HAC	Hierarchical combination	Ranks users by HCP, breaks ties with N06 and if ties still exist it resolves them with HAC.
FSN	Fusion	Ranks users by HCP and HAC (two rankings) and selects the user with the highest average position in the two ranked lists.

Algorithm 2 Focused User Crawler. Batch Procedures

```
1: procedure Batch Training
2:     while true do
3:         classification_model.train(positive_users, negative_users)
4:         Batch Updater()
5:         sleep(SLEEP_TIME)
6:
7: procedure Batch Updater
8:     for each user in users_ranking.get_users() do
9:         user.update_proba()
```

experimented with HCP-0.9, which only incorporates pseudo-positive examples (those with a probability higher than 0.9). Our hypothesis is that this injection of *pseudo-training* data might improve the chances of identifying target users.

HCP-N06-HAC: This strategy combines the three best-performing methods in a hierarchical way. First, the users with the highest HCP score are selected. Since there are usually ties, these are resolved by selecting those users with the highest N06 score. If we still have ties on N06 scores, then the HAC score is used to resolve them. In the unlikely event of ties in all scores, the user is chosen at random.

Fusion: (FSN). This prioritisation proposal merges the rankings of HCP and HAC by averaging the positions of the users in these two rankings. This fusion approach continuously analyses and merges the top 100 users of the two base rankings.

Table 2 briefly sketches the advanced prioritisation methods described above.

A. EXTENSION OF THE ARCHITECTURE

HCP with pseudo-training needs to update the prediction classifier. To meet this aim, two additional tasks are needed: Model Trainer and Batch Updater. The first builds a new classifier based on the original training data plus new pseudo-examples. Once a new classification model is built, the second task makes an update on all users (the classifier has changed and, thus, the predictions on the available user's texts need to be changed):

- **Model Trainer**. It receives the classification result from User Classifier and decides if the user will be added to the training dataset (based on the confidence score). Every five minutes, the classification model will be trained (but only if new examples were added). The new model is spread to the instances that update the user rankings. When a new model is trained, the User Updater instances are invoked through RPC so that they can update the model.
- **Batch Updater**. According to the HCP method, the ranks of the users are updated following the newly trained model.

These tasks run in parallel completing the system architecture shown in Figure 1. The Batch Updater is only activated when a new model is available. These new procedures are sketched in Algorithm 2.

B. EXPERIMENTAL RESULTS

In a second experiment, the new proposals and the best performers of the first experiment were run for four days. In Figure 5, we plot the cumulative number of extracted users for each method. The extraction capacity of the fusion proposal is significantly lower than the capacity of the other ones. This outcome is due to the heavy and repetitive task of merging rankings. The existing bottlenecks could be easily avoided by incorporating more instances of the User Chooser module, but extraction performance was not the main goal of these experiments.

In Figure 6, we can observe that the most solid choice is the fusion method (FSN). FSN is outperformed by

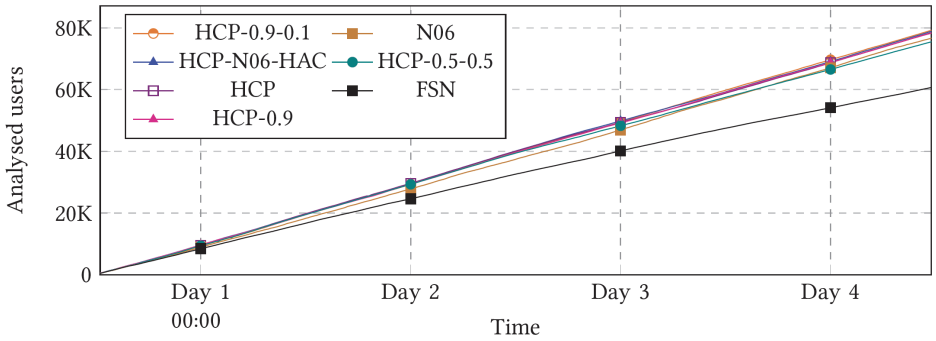

FIGURE 5. Second experiment. Number of extracted users by each focused user crawling method.

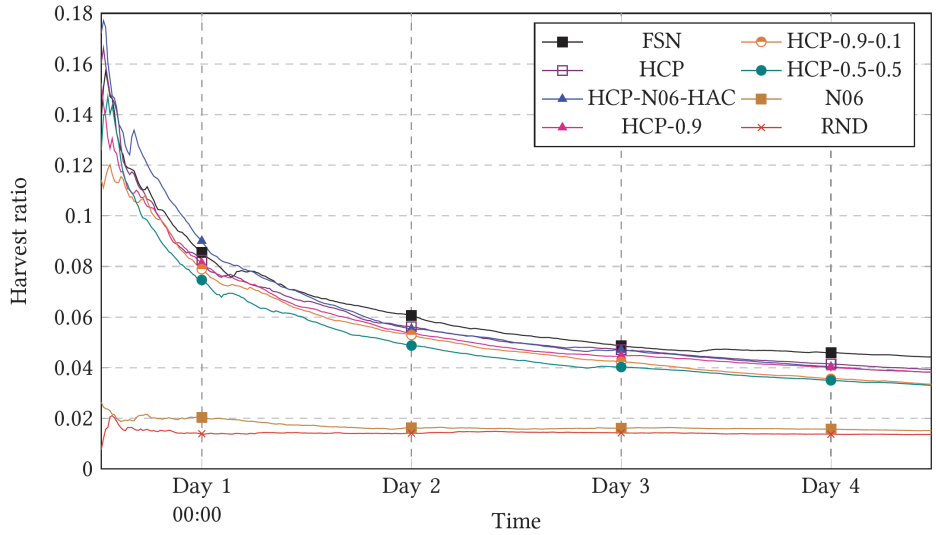

FIGURE 6. Ratio of users classified as positive over the total for the heuristic proposals of the second experiment with a threshold of 0.9.

HCP-N06-HAC during the first day of the experiment but, after that, FSN becomes the most valuable source of relevant users. At the end of the experiment, FSN yielded an overall harvest ratio that is 20.7% better than the ratio achieved by HCP-N06-HAC and 12.9% better than the ratio achieved by HCP. The harvest ratio of HCP was 6.9% better than HCP-N06-HAC. These results suggest that if we need a sustained rate of extraction of on-topic users then FSN, which merges HCP and HAC, should be our method of choice. However, if the crawler is employed for quickly extracting a sample of relevant users, then HCP-N06-HAC should be preferred. During the first day, the HCP-N06-HAC method performs 12.7% better than FSN, which behaves comparably to HCP for the first 24 hours. This second experiment also revealed that pseudo-training was ineffective at improving the harvest ratio of the HCP crawler. None of the pseudo-training variants performed better than the standard HCP, which never updates the prediction classifier.

Table 3 helps to further analyse the HCP-N06-HAC method. In many cases, this method extracts the same users as HCP. However, as shown in the table, the number of ties in the HCP scores is not negligible. HCP breaks these ties randomly, while HCP-N06-HAC resorts to the second level user score (N06) and, if required, to the third level user score (HAC). Such a smart tie-breaking approach was productive in early identifying target users. HCP-N06-HAC outperformed all the other methods during the first day of the experiment. After the first 24 hours, this crawling approach descended from a harvest ratio of 0.18 to levels around 0.07.

TABLE 3. Number of ties in the HCP and N06 scores for the HCP-N06-HAC method. The table also reports the total number of ties and the ratio of ties with respect to the number of users.

	HCP	N06	Total	Ratio	Extracted Users
Day 1	483	76	559	0.056	9,920
Day 2	2,410	1,096	3,506	0.117	29,922
Day 3	3,836	1,248	5,084	0.101	50,119
Day 4	5,571	1,581	7,152	0.103	69,382

Figure 7 plots the number of on-target users detected by each FUE method. There are no major differences among the methods tested, but the graphs reveal the main temporal trends of extraction of relevant users. The detection of

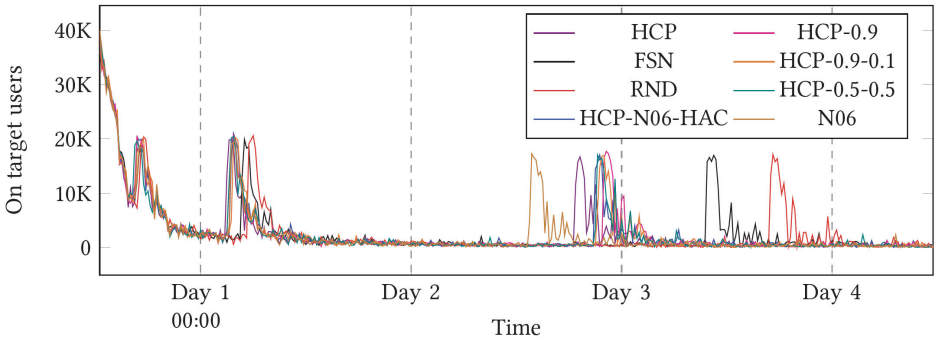

FIGURE 7. Number of new on-target users detected by each crawling method.

on-target users peaks during the first day. Initially, all target users are unexplored and the FUE methods can work with a large sample of potentially relevant users. The initial peaks suggest that the strategies are effective at quickly guiding their exploration to relevant users. After day 1, the curves become flattered and new on-target users are detected in periodic waves (e.g., because new evidence is found about previously existing users or because some users become active and, thus, their usernames are revealed to the crawler). The peaks are associated to new interactions by on-target users. In SM, user interactions tend to be clustered around certain time points and communities (e.g., new discussion threads become highly active). Note also that, after Day 3, the peaks become lower. This suggests that most relevant communities of users have been explored and the pattern of identification of new on-target users is smoother.

In summary, HCP-N06-HAC and FSN were the most effective user crawling methods. HCP-N06-HAC shows a solid behaviour during the first day, when the number of new (previously unknown) users is large and, potentially, there are many ties. As the hours pass and the number of new users stabilises, a more sophisticated method, FSN, which merges the HCP and HAC ranks of the users, is required.

VI. DISCUSSION

In focused user crawling, it is crucial to speed up the process of extracting on-target users. For example, in early risk applications (e.g., detecting criminal recruiters or identifying individuals showing signs of psychological disorders), it is important to maximise the number of users processed per unit of time and quickly focus the analysis on users who are potentially relevant. In this way, effective and efficient crawlers can support multiple user screening tasks and facilitate preventive measures.

Depending on the application domain, such screening actions and the associated interventions need to be designed with full consideration of proper ethical guidelines. For example, as Neuman [21] suggests, screening of signs of depression on Social Media could be employed by health agencies to economically screen massive samples of the population, before more accurate, albeit more expensive, steps can be taken. Given the subject's permission, a screening system, powered by user crawlers such as those developed here, may extract signs of depression in SM texts and, if concerning symptoms are detected, the system may inform the subject and offer the opportunity to complete an online questionnaire (or the subject might be advised to consult a medical expert). In any case, the development of solid focused user crawling solutions can impact a number of areas including cybercrime detection, expert search, and health, just to name a few. With these new technologies, the work of experts, authorities and other stakeholders can be eased through the effective and rapid filtering and prioritisation of users who likely match a given pattern of interest.

Our proposal focused on Reddit as the reference source for experimentation but the lessons learned here are potentially applicable to other SM platforms. This is due to the modularity of our crawling architecture and the commonalities among SM sources. For example, it would be relatively easy to transfer this research to other social networks in which text-based posts are common and openly accessible.

User features, such as the time of posting or the probability assigned by the classifier, can be applied on most SM platforms. Furthermore, microblogging platforms, such as Twitter or Mastodon, would give us the opportunity of exploiting additional features such as certain hashtags, words or geotags (instead of subreddits). Additionally, we could design new prioritisation methods from variables such as the number of likes/favourites, the number of retweets/boosts of a tweet/toot or the number of replies. New crawlers oriented to these platforms can be easily implemented in our architecture and the crawling system can be connected to API services of SM, where available.

Note also that the most effective prioritisation method is a text-based approach that predicts user's relevance based on the user's available interactions. This type of textual evidence exists in nearly all SM platforms and, thus, there is potential to apply our results to many other SM sources.

VII. RELATED WORK

Our work is related to several lines of research, which include real-time crawling, focused web crawling, and user

TABLE 4. Related papers organised by topic and main attributes.

Real-Time Crawling / Social Media Sampling				
Paper	**Source**	**Task**	**Method**	**Type**
[28]	Twitter	Event Monitoring	Recall-oriented adaptive crawling that identifies new event-related keywords	Real-Time
[22]	Twitter	Event Monitoring	Keyword-based Adaptive Crawler	Real-Time
[29]	Flickr/Youtube	Crawling SM pages	User profiles used as ranking criteria for guiding the crawling process	Not reported
[23]	Twitter and Census Data	Drug Abuse Monitoring (community-focused)	Keyword-based & Deep Learning (CNN & LSTM)	Near Real-Time
[24]	Weibo.com	Social Network Crawling	User-based model to extract fresh contents	Real-Time
[25]	Facebook	Social Network Crawling	User-guided method to prioritize crawling contents	Real-Time
[26]	Xiami/YouTube/ Flickr	Content Characterisation	Graph-based Sampling	-
[27]	Twitter	Estimating SM bias	Network-based approach	-
[30]	Flickr	Multimedia Indexing	Feedback-based approach	-

Focused Web Crawling			
Paper	**Source**	**Task**	**Method**
[31]	WebKB	Web Page Categorisation (predict the topic of a linked webpage)	SVM (features based only on URL)
[32]	Common Crawl	Crawling Structured Data (data-rich webpages)	Online Learning and Bandit-based methods
[33]	Web	Focused Web Crawling	Evolutionary crawler (exploitation+exploration)
[34]	Web	Web and Social Media Focused Crawling	Integration of SM streams & Web to continuously guide the crawler towards fresh and relevant content

User or Community Extraction/Crawling			
Paper	**Source**	**Task**	**Method**
[35]	Twitter	User Crawling	Multi-Armed Bandit Approach
[36]	Tumblr	User Crawling	Random walk, centrality and link features
[37]	Medical Support Forums	Ranking User Influence	Link-based and content-based approach
[38]	Epinions/Flickr	User Behaviour Prediction	Neural model
[39]	egoFacebook	Community Crawling	Branch & Bound network algorithm to extract socially tight communities
[40]	Flickr	Community Discovery	Low-rank matrix approach
[41]	Twitter	Campaign Extraction	Content-driven graph-based approach

Risk Analysis & Social Media				
Paper	**Source**	**Task**	**Risk**	**Method**
[42] (survey)	Twitter/Facebook	User Classification	Depression/PTSD	Regression or Classification
[43] (survey)	Twitter/Reddit/ Facebook/Tumblr/...	User Classification	Depression/PPD/Eating Disorders/PTSD/...	Multiple (supervised, unsupervised, ...)
[44]	Twitter	User Classification	Online Radicalisation	KNNs & SVMs
[45]	Formspring.me/ MySpace	Post Classification	Cyberbullying	Genetic algorithms & Fuzzy rules

extraction. The following subsections summarise the work that has been done in these areas and emphasise the differences with our approach. Table 4 provides a global view of related papers and their main attributes.

A. REAL-TIME CRAWLING OF SOCIAL MEDIA

There are several studies in the literature that propose crawlers to support real-time extraction and analysis of SM data. For example, in [22], the authors present a system that monitors real-time events on Twitter using an adaptive crawler. This crawler was guided by periodically updated keywords, which were tagged as relevant for a given event. In [23], a monitoring system to detect drug abuse risk behaviours from SM was described. This system worked in near real-time. In [24], the authors optimised a social network crawler based on the posting frequency of their users. In this way, the crawling process could crawl most of the newly produced content with limited resources (and taking into account the access restrictions of the SM source). In [25], a user-guided Social Media crawling method was proposed. The goal was not to crawl the entire SM platform (or extract the full set of users) but instead to obtain a sample of posts or submissions that are statistically representative of the entire dataset. The approach is based on utilising user-based evidence to decide in which order the SM contents should be collected. In [26], several methods are described to build

an unbiased sample of social media content to characterise the vast amount of the existing content given the networks' large-scale nature and query imposed limitations. Zhou and colleagues [27] were instead interested in measuring SM bias and proposed a hybrid sampling strategy that considers network information.

Wang and Nasraouni [28] proposed a Twitter crawling approach where they can dynamically detect and monitor new hashtags related to an initial set of keywords during a live event. In [29], the authors analysed SM user profiles for guiding the crawling of posts. This methodology speeds up the crawling process of relevant posts but requires analysing the profile of each newly detected user in a FIFO fashion.

Some teams focused on how to extract specific media from SM platforms. For example, Leung and colleagues [30] proposed an adapted search engine architecture aimed at indexing multimedia resources (e.g., images or videos). The estimation of relevance relies on user's ratings as evaluated by a community of users. By incorporating this form of user feedback, this approach allows to gradually discover and index certain properties of media resources.

B. FOCUSED WEB CRAWLING

Focused Web Crawling techniques aim to extract web pages that satisfy a given pattern, and therefore, are related to the FUE task introduced in this paper. In [31], Kan analysed the performance of web page categorisation using exclusively the URL of the web page in order to determine the relevance of its content. In [32], Meusel and colleagues predicted data-rich web content based on the context of the page and other metadata available from previously extracted pages. Their approach was a combination of online learning and a bandit-based selection process. In [33], a framework to fairly evaluate focused crawlers was proposed. Such a framework was employed to further design new evolutionary crawlers that combine exploration and exploitation. These are examples of studies that effectively crawl on-topic contents guided by predictive features. Our paper represents an attempt to build similar effective and efficient solutions for extracting Social Media users that are relevant to a given profile of interest.

There are several studies where Social Media is used to improve the performance of a focused crawler. However, the vast majority of them do not focus on retrieving relevant users but, instead, focus on extracting relevant Web pages or documents. In [34], the authors proposed a focused crawling approach that analyses the interlink between the Web and SM content in order to guide the crawler towards fresh and relevant content.

For a more general discussion of web crawling, categories of crawling and their main challenges the reader can refer to the survey by Kumar and colleagues [4].

C. USER OR COMMUNITY EXTRACTION/CRAWLING

A few studies in the literature explored user extraction tasks that are similar to ours. In [35], Gisselbrecht and colleagues implemented a multi-armed bandit approach that guides a real-time capture towards users that are most likely to produce data on a given topic. Our approach is based on the study of a set of prioritisation methods that lead to quick identification of target users, while Gisselbrecht's approach needs to listen to the users in real-time (for a given period of time) in order to assign a reward. Another important difference is that Gisselbrecht and colleagues strictly analysed real-time data while we performed a historical analysis of every selected user in order to assess user relevance.

Focusing on the detection of hate-promoting users and discovering their communities, Agarwal and Sureka [36] implemented a focused user crawler for Tumblr. First, a number of texts are analysed to determine if they are relevant. If so, metadata from those posts are collected, feeding up a social graph. This graph is used to guide the crawling process through stochastic exploration. Their approach makes further analysis of the information available and, in this way, extremist communities can be undercovered. This study represents a case of focused user crawling where the focus is extremism and the method only analyses users related to those who have been already classified as extremists. Their goal is to undercover hidden extremist communities and they do not perform an ordered extraction or prioritisation of the most relevant users. Furthermore, their approach is computationally expensive because a substantial amount of user texts are needed to guide the crawler.

Some teams proposed other user-oriented tasks, such as ranking user influence or user behaviour prediction that are intrinsically different from the FUE task explored in our paper. For example, Tang and Yang [37] were interested in measuring a user's influence on other users in online healthcare forums and, to that end, user influence was estimated from users' reply relationships, conversation content and response immediacy. Li and colleagues [38] designed a neural approach to predict user behaviour (e.g., social link behaviour or consumption behaviour).

Hsu and colleagues [39] proposed a network-based algorithm that supports crawling community-aware data. Their approach is not oriented to support a massive analysis of Social Media users but, instead, it focuses on how to obtain valuable community information for research purposes. The proposed solution identifies socially tight communities by optimising *user willingness*, which estimates what users are willing to contribute their data. Related to this, Zhuang et al [40] presented a low-rank matrix recovery technique-oriented to discovering communities from SM.

Rather than identifying users or communities, some researchers aimed at extracting certain types of behaviour. For example, Lee and colleagues [41] proposed a campaign extraction method. Their approach, which is content-based and graph-based, identifies and extracts campaigns (e.g., coordinated spamming, promotional campaigns or political ''astro-turfing'') in large-scale social media.

FUE techniques could be applied to support health-related applications, such as detection of users with signs of the onset

of mental illnesses (e.g., depression, anorexia or dementia). As a matter of fact, depression and behavioural disorders, such as irritability, can constitute part of the symptomatology of neuro-degenerative mental diseases such as Alzheimer, Huntington or Creutzfeldt-Jakob. FUE can also assist in analysing users massively, searching for threats related to online harassment or terrorist radicalisation. Our FUE crawling methods can, in general, be employed to detect users showing any behavioural pattern identifiable by their public activity on SM. There are numerous works of SM analytics that target risks such as mental illnesses [42], [43], radicalisation [44] or harassment [45]. However, these studies are oriented to classification or predictive aspects and they are not concerned with how to efficiently extract target users from massive streams of user data.

D. FINAL REMARKS

Table 4 presents a structured view of the studies described above. Our paper represents a distinctive contribution where the latest (unseen) user posts are analysed, and the corresponding authors are ranked according to several proposed prioritisation methods. New users are analysed from little context and, thus, this represents a lightweight crawling method that rapidly explores communities in the quest of relevant users. Our experimentation, performed on Reddit, demonstrates that the exploration is efficient and effective.

VIII. CONCLUSION

In this paper, we have approached (and formally defined) the task of Focused User Extraction (FUE), which has received little attention in the literature. In FUE, the main goal is to extract as many relevant users as possible, being guided by a minimal context (i.e., only using the information available at the pages where the usernames are initially observed). As argued above, most previous studies on extracting SM users were not explicitly concerned about efficiency and about how to quickly harvest the most relevant users.

As a case study, we have implemented a focused crawler oriented to an early risk task. We proposed a set of user prioritisation methods that proved to work well with limited user contexts. In a first experiment, we have determined that HCP is a robust FUE method that only requires the last textual interactions written by the users. In a second experiment, we have found that two variants of the basic FUE methods can further improve the harvest ratios. We have also shown that our FUE methods perform satisfactorily with huge volumes of users and interactions. In terms of harvest ratios, the crawlers initially detect many on-target users and, next, the harvest ratios tend to decay. This is a natural consequence of the fact that all users are initially unexplored and, thus, at the beginning of the process, the chances of extracting relevant users are higher. Our experiments also revealed that, as the crawlers proceed, smarter methods, which combine multiple signs, are required to further mine on-target users.

Despite the simplicity of the methods tested, their harvest ratios are high. In the future, we plan to study more sophisticated and formal ways to prioritise users. The methods proposed here can also serve as an input to bootstrap other future focused user crawlers. We have used Reddit to perform our experiments but our contributions are potentially applicable to other SM platforms.

REFERENCES

[1] F. Menczer, "ARACHNID: Adaptive retrieval agents choosing heuristic neighborhoods for information discovery," in *Proc. Mach. Learn.-Int. Workshop Then Conf.*, San Mateo, CA, USA: Morgan Kaufmann, 1997, pp. 227–235.

[2] S. Chakrabarti, M. van den Berg, and B. Dom, "Focused crawling: A new approach to topic-specific web resource discovery," *Comput. Netw.*, vol. 31, nos. 11–16, pp. 1623–1640, May 1999.

[3] B. Novak, "A survey of focused web crawling algorithms," in *Proc. SIKDD*, vol. 5558, 2004, pp. 55–58.

[4] M. Kumar, R. Bhatia, and D. Rattan, "A survey of web crawlers for information retrieval," *WIREs Data Mining Knowl. Discovery*, vol. 7, no. 6, Nov. 2017, Art. no. e1218. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1218

[5] G. Almpanidis, C. Kotropoulos, and I. Pitas, "Combining text and link analysis for focused crawling—An application for vertical search engines," *Inf. Syst.*, vol. 32, no. 6, pp. 886–908, Sep. 2007.

[6] Twitter. (2021). *Q1 2021: Letter to Shareholders*. Accessed: Jul. 6, 2021. [Online]. Available: https://investor.twitterinc.com/files/doc_financials/2021/q1/Q1'21-Shar%eholder-Letter.pdf

[7] Twitter. (2021). *Twitter Annual Reports*. Accessed: Jul. 6, 2021. [Online]. Available: https://investor.twitterinc.com/financial-information/annual-reports/

[8] I. L. Statistics. (2021). *Twitter Rates*. Accessed: Jul. 6, 2021. [Online]. Available: https://www.internetlivestats.com/twitter-statistics/##rate

[9] Reddit. (2021). *Press Release*. Accessed: Jul. 6, 2021. [Online]. Available: https://redditinc.compress

[10] R. Martínez-Casta no, J. C. Pichel, D. E. Losada, and F. Crestani, "A micro-module approach for building real-time systems with Python-based models: Application to early risk detection of depression on social media," in *Advances in Information Retrieval*. Cham, Switzerland: Springer, 2018, pp. 801–805.

[11] R. Martínez-Castaño, J. C. Pichel, and D. E. Losada, "Building Python-based topologies for massive processing of social media data in real time," in *Proc. 5th Spanish Conf. Inf. Retr.*, Jun. 2018, p. 18.

[12] R. Martínez-Castaño, J. C. Pichel, and D. E. Losada, "A big data platform for real time analysis of signs of depression in social media," *Int. J. Environ. Res. Public Health*, vol. 17, no. 13, p. 4752, Jul. 2020.

[13] (2017). *Apache Kafka*. Accessed: Feb. 2022. [Online]. Available: https://kafka.apache.org/

[14] (2021). *Docker*. Accessed: Feb. 2022. [Online]. Available: http://www.docker.com/

[15] (2021). *Reddit on Alexa*. Accessed: Feb. 2022. [Online]. Available: https://www.alexa.com/siteinfo/reddit.com/

[16] (2021). *About Reddit*. Accessed: Feb. 2022. [Online]. Available: https://www.redditinc.com/

[17] S. Chakrabarti, K. Punera, and M. Subramanyam, "Accelerated focused crawling through online relevance feedback," in *Proc. 11th Int. Conf. World Wide Web (WWW)*, 2002, pp. 148–159.

[18] (2021). *MongoDB*. Accessed: Feb. 2022. [Online]. Available: https://www.mongodb.com/

[19] (2018). *Aerospike*. Accessed: Feb. 2022. [Online]. Available: https://www.aerospike.com/

[20] D. E. Losada and F. Crestani, "A test collection for research on depression and language use," in *Proc. CLEF*, 2016, pp. 28–39.

[21] Y. Neuman, Y. Cohen, D. Assaf, and G. Kedma, "Proactive screening for depression through metaphorical and automatic text analysis," *Artif. Intell. Med.*, vol. 56, no. 1, pp. 19–25, Sep. 2012.

[22] A. T. Hadgu, S. Abualhaija, and C. Niederée, "Sover! Social media observer," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 1305–1308.

[23] H. Hu, N. Phan, X. Ye, R. Jin, K. Ding, D. Dou, and H. T. Vo, "DrugTracker: A community-focused drug abuse monitoring and supporting system using social media and geospatial data (Demo Paper)," in *Proc. 27th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2019, pp. 564–567.

[24] R. Guo, H. Wang, M. Chen, J. Li, and H. Gao, "Parallelizing the extraction of fresh information from online social networks," *Future Gener. Comput. Syst.*, vol. 59, pp. 33–46, Jun. 2016.
[25] F. Erlandsson, P. Bródka, M. Boldt, and H. Johnson, "Do we really need to catch them all? A new user-guided social media crawling method," *Entropy*, vol. 19, no. 12, p. 686, Dec. 2017. [Online]. Available: https://www.mdpi.com/1099-4300/19/12/686
[26] P. Wang, J. Zhao, J. C. S. Lui, D. Towsley, and X. Guan, "Fast crawling methods of exploring content distributed over large graphs," *Knowl. Inf. Syst.*, vol. 59, no. 1, pp. 67–92, Apr. 2019.
[27] Y. Zhou, R. Ji, J. Su, and J. Yao, "Uncovering media bias via social network learning," *ACM Trans. Intell. Syst. Technol.*, vol. 12, no. 1, pp. 1–12, Feb. 2021.
[28] X. Wang, L. Tokarchuk, F. Cuadrado, and S. Poslad, "Exploiting hashtags for adaptive microblog crawling," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Aug. 2013, pp. 311–315.
[29] Z. Zhang and O. Nasraoui, "Profile-based focused crawler for social media-sharing websites," in *Proc. 20th IEEE Int. Conf. Tools With Artif. Intell.*, Nov. 2008, pp. 317–324.
[30] C. H. C. Leung, A. W. S. Chan, A. Milani, J. Liu, and Y. Li, "Intelligent social media indexing and sharing using an adaptive indexing search engine," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, pp. 1–27, May 2012, doi: 10.1145/2168752.2168761.
[31] M.-Y. Kan, "Web page classification without the web page," in *Proc. 13th Int. World Wide Web Conf. Alternate Track Papers Posters (WWW Alt.)*, 2004, pp. 262–263.
[32] R. Meusel, P. Mika, and R. Blanco, "Focused crawling for structured data," in *Proc. 23rd ACM Int. Conf. Conf. Inf. Knowl. Manage.*, Nov. 2014, pp. 1039–1048.
[33] F. Menczer, G. Pant, and P. Srinivasan, "Topical web crawlers: Evaluating adaptive algorithms," *ACM Trans. Internet Technol.*, vol. 4, no. 4, pp. 378–419, Nov. 2004.
[34] G. Gossen, E. Demidova, and T. Risse, "ICrawl: Improving the freshness of web collections by integrating social web and focused web crawling," in *Proc. 15th ACM/IEEE-CS Joint Conf. Digit. Libraries*, Jun. 2015, pp. 75–84.
[35] T. Gisselbrecht, L. Denoyer, P. Gallinari, and S. Lamprier, "Whichstreams: A dynamic approach for focused data capture from large social media," in *Proc. 9th Int. AAAI Conf. Web Social Media*, 2015, pp. 130–139.
[36] S. Agarwal and A. Sureka, "Spider and the flies : Focused crawling on Tumblr to detect hate promoting communities," 2016, *arXiv:1603.09164*.
[37] X. Tang and C. C. Yang, "Ranking user influence in healthcare social media," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 4, pp. 1–21, Sep. 2012, doi: 10.1145/2337542.2337558.
[38] J. Li, L. Wu, R. Hong, K. Zhang, Y. Ge, and Y. Li, "A joint neural model for user behavior prediction on social networking platforms," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 6, pp. 1–25, Dec. 2020, doi: 10.1145/3406540.
[39] B.-Y. Hsu, C.-L. Tu, M.-Y. Chang, and C.-Y. Shen, "On crawling community-aware online social network data," in *Proc. 30th ACM Conf. Hypertext Social Media*, Sep. 2019, pp. 265–266, doi: 10.1145/3342220.3344937.
[40] J. Zhuang, T. Mei, S. C. H. Hoi, X.-S. Hua, and Y. Zhang, "Community discovery from social media by low-rank matrix recovery," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 4, pp. 1–19, Jan. 2015.
[41] K. Lee, J. Caverlee, Z. Cheng, and D. Z. Sui, "Campaign extraction from social media," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 1, pp. 1–28, Dec. 2013, doi: 10.1145/2542182.2542191.
[42] S. C. Guntuku, D. B. Yaden, M. L. Kern, L. H. Ungar, and J. C. Eichstaedt, "Detecting depression and mental illness on social media: An integrative review," *Current Opinion Behav. Sci.*, vol. 18, pp. 43–49, Dec. 2017.
[43] E. A. Ríssola, D. E. Losada, and F. Crestani, "A survey of computational methods for online mental state assessment on social media," *ACM Trans. Comput. Healthcare*, vol. 2, no. 2, pp. 1–31, Mar. 2021.
[44] S. Agarwal and A. Sureka, "Using KNN and SVM based one-class classifier for detecting online radicalization on Twitter," in *Proc. Int. Conf. Distrib. Comput. Internet Technol.*, Cham, Switzerland: Springer, 2015, pp. 431–442.
[45] B. S. Nandhini and J. I. Sheeba, "Online social network bullying detection using intelligence techniques," *Proc. Comput. Sci.*, vol. 45, pp. 485–492, Jan. 2015.

RODRIGO MARTÍNEZ-CASTAÑO received the B.Sc. degree in computer science and the M.Sc. degree in big data and data analysis technologies from the University of Santiago de Compostela, Spain. He is currently pursuing the Ph.D. degree. His research interests include big data technologies, distributed systems, information retrieval, and blockchain.

DAVID E. LOSADA is currently an Associate Professor in computer science and artificial intelligence with CiTIUS, University of Santiago de Compostela, Spain. His current research interests include a wide range of information retrieval (IR) and related areas, such as early risk detection, text mining, IR evaluation, IR probabilistic models, summarization, novelty detection, and sentence retrieval. He is an Active Member of the IR Community and he regularly serves on the Program Committee of prestigious international conferences, such as SIGIR or ECIR. In 2011, he was recognized with the ACM Senior Member Award.

JUAN C. PICHEL received the B.Sc. and M.Sc. degrees in physics from the University of Santiago de Compostela, Spain, and the Ph.D. degree in computer science from the University of Santiago de Compostela, in 2006. He was a Visiting Postdoctoral Researcher with the University Carlos III de Madrid, Spain, and the University of Illinois at Urbana–Champaign, USA. He also worked as a Researcher and the Project Manager with the Galicia Supercomputing Center, Spain. He is currently an Associate Professor with CiTIUS, University of Santiago de Compostela. His research interests include parallel and distributed computing, big data technologies, programming models, and software optimization techniques for emerging architectures.