Computers & Geosciences 49 (2012) 164–169

Contents lists available at SciVerse ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Social.Water—A crowdsourcing tool for environmental data acquisition

Michael N. Fienen [a,*], Christopher S. Lowry [b]

[a] US Geological Survey, Wisconsin Water Science Center, 8505 Research Way, Middleton, WI 53562, USA
[b] University at Buffalo, Geology Department, 411 Cooke Hall, Buffalo, NY 14260, USA

ARTICLE INFO

Article history:
Received 16 April 2012
Received in revised form
14 June 2012
Accepted 18 June 2012
Available online 26 June 2012

Keywords:
E-geoscience
Citizen science
Hydrology
Streamgaging
Python

ABSTRACT

Remote telemetry has a long history of use for collection of environmental measurements. With the rise of mobile phones and SMS text-messaging capacity, many members of the general pubic carry communications equipment in their pockets at all times. Enabling the general public to provide environmental data through text messages has the potential both to provide additional data to scientific projects and also to raise awareness of the projects through participation. Hydrologic measurements – some of which can be made without training, involve a single measurement, and are often made in rural areas – are well-suited to text-message conveyance. Many other environmental measurements are similarly well-suited for this technology. Social.Water is a software package, written in Python, that collects, parses, and categorizes text messages sent to a dedicated phone number, updates a simple database, and posts both graphical results and the database on the Web. Social.Water was designed as the backend to the Crowdhydrology project and is written in an object-oriented design that makes customization and modification straightforward.

Published by Elsevier Ltd.

1. Introduction

Acquisition of field data is an expensive part of most geoscience projects—it is also an opportunity for geoscientists to interact with the public. Lowering the cost of data acquisition in concert with increasing public interaction can provide science benefits to projects from reducing the expense of data collection and through increased public engagement. Allowing citizen-scientists to contribute data to a scientific project is an example of crowdsourcing. Crowdsourcing means obtaining information or analysis from the "crowd" – the general public – and is so-named as an adjunct to outsourcing (Howe, 2006). An example of this approach is the Crowdhydrology project (Lowry and Fienen, in press). In this project, the authors used text messages to obtain stream water levels at multiple sites in upstate New York, USA. Visitors to the sites saw signs posted on water level gages asking that they send a text message with the station number and the water level reading from the gage. These messages are forwarded to an email server where they are parsed by a script to associate measurements with specific gages, and then displayed in near real time on the Web. Between May 2011 and February 2012, nearly 150 measurements from nine locations were submitted by citizen scientists in this way. This work details the software package, Social.Water, which forms the backbone infrastructure for the Crowdhydrology project. Lowry and Fienen (in press) also describe a variety of previous projects using crowdsourcing technology in natural science applications.

Social.Water is a program written in Python, building on the open-source tools that form the protocol for Crowdhydrology. The main objective for Social.Water is to provide a simple, modular, and inexpensive way to enlist the general public in collecting scientific data (stream water levels in the case of Crowdhydrology). The data are transmitted using text message protocols. The data obtained in this way can supplement measurements made by project staff when telemetry or continuous recording are infeasible. A secondary but important outcome of using this protocol is engagement and, in a sense, ownership by citizens who encounter the field sites and contribute information. Enlisting citizen-scientists in data collection efforts dates at least back to the inception of the Audobon Society's Christmas Bird Counts (Wiersma, 2010) in 1900. The recent proliferation of mobile phones and smartphones means telemetry more sophisticated than one dreamt of 20 years ago is in nearly everyone's pocket.

Social.Water depends on text messages forwarded to an IMAP-enabled email account for the transmission of data. An obvious alternative would be the use of a smartphone application such as CreekWatch (IBM, 2012). Smartphone applications allow for automatic geolocation, submission of photographs, and other advantages such as delayed synching when off-network. Recent developments in HTML5 standards also make cross-platform development realistic (Isaac, 2011). However, a goal with Social.Water is to allow transmission of strictly text and numerical data

* Corresponding author.
E-mail addresses: mnfienen@usgs.gov (M.N. Fienen), cslowry@buffalo.edu (C.S. Lowry).

0098-3004/$ - see front matter Published by Elsevier Ltd.
http://dx.doi.org/10.1016/j.cageo.2012.06.015

on a platform that is most commonly available. Despite the popularity of smartphones in the United States, only 46% of adults have them, compared to 87% of adults who have some kind of mobile phone (Smith, 2012). Furthermore, a smartphone app requires users to download and install the app prior to participation. The guiding precept of this project was to lower the barriers to participation as much as possible such that the simplest implementation with the lowest burden on users would be realized.

One challenge in using text messages rather than a dedicated smartphone application is the need to interpret, parse, and categorize the messages to extract the relevant data. In the Crowdhydrology project, instructions to observers were intentionally simple and imprecise. Adaptations to the code, discussed below, were required for successful parsing and categorization of results.

In the remainder of this paper, we discuss the details of implementation of the Social.Water code, review the application to the Crowdhydrology project, and provide conclusions and future plans.

2. Social.Water code

Social.Water is written using an object-oriented approach in Python (van Rossum, 2012) version 2.7.2. Because Social.Water is designed in an object-oriented way, the main code in `sw_driver.py` is only a few lines that initialize an object to contain the information in the code and call methods that perform actions on that object. Several dependent files must be in the path for `sw_driver.py` to access them. The classes used by Social.Water are in `social_water.py`. Two other dependent scripts are in `fuzz.py` and `process.py`. These two scripts contain functions from the fuzzy-search algorithm fuzzywuzzy (Cohen, 2011) discussed below. In the remainder of this section, we discuss the mechanics of Social.Water implementation. The code is meant to balance generality with code maintenance and customization. As a result, some customization is required to deploy Social.Water on projects other than Crowdhydrology. Fig. 1 illustrates the general process flow of Social.Water. The entire Social.Water code, along with ancillary code and datafiles, are installed on a server and run through a cron script every 5 min. The runtime is typically less than a second and the majority of times the cron script executes, Social.Water logs into an email account, detects that no new messages are present, and immediately exits.

An initial requirement to implement Social.Water is to forward text messages to an Internet Message Access Protocol (IMAP, Internet Engineering Task Force, 2003b)-enabled email account. We chose Google Voice (voice.google.com) because it is free and can be set up with a dedicated phone number serving the sole purpose of receiving text messages and forwarding them to the email account. Social.Water is run on a server and, using IMAP, checks a free email account every 5 min, parses new messages to determine if they contain valid water level measurements associated with known gages, updates a simple flat database, and displays a graphical result on an HTML page.

Information required to initialize the class `email_reader` is a user name for an email account, an obfuscated password, and the scope of the email search ("UNSEEN" only reads messages marked as unread and "ALL" reads all messages in the email account). For the Crowdhydrology project, we used a Gmail account (gmail.google.com) as a free, IMAP-enabled email client—if a different email client is desired, `email_reader.login()` would need to be updated. The password is obfuscated using the python `base64` module implementing RFC 3548 (Internet Engineering Task Force, 2003a). This obfuscation is not meant to be highly secure, but it prevents a non-human hack from obtaining the password in plain text from the codebase. Nonetheless, we recommend using a sacrificial, dedicated email account for this purpose to avoid any conflict with private data.

After initialization, `email_reader.login()` is called to connect to the IMAP server. `email_reader.checkmail()` checks to see if new messages, based on the email_scope outlined above, are present in the IMAP account. If no new messages are present, Social.Water exits because no new work is required of it. If new messages are present, `email_reader.parsemail()` pulls the date stamps and message bodies from the new messages, only considering new messages with the text "SMS from" in the subject line. This prevents spam, advertisements, or any other non-data email messages from being further considered.

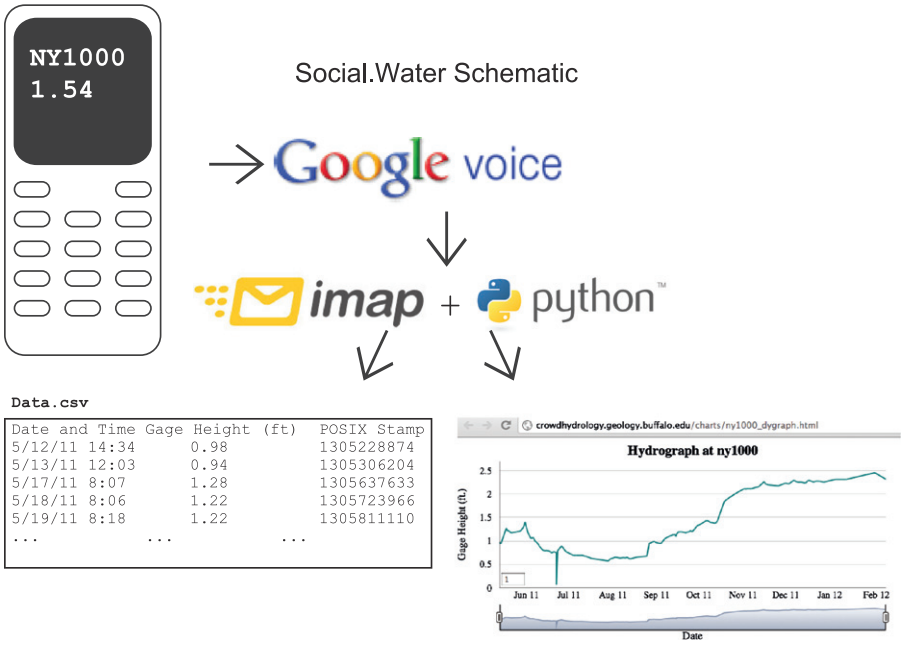

Fig. 1. Schematic representation of the Social.Water program flow from text message, forwarded to email, parsed, and displayed as a data table and graphically.