Setting up a Text Message Announcement System

October 22, 2015

At this year's DubHacks, we used a text announcement system to send important information to over 700+ attendees, mentors, and sponsors throughout the hackathon. As attendees were working in separate classrooms spread across multiple buildings, the mass-SMS system was particularly effective in notifying everyone of upcoming events. This post explains how to quickly setup and use Batch SMS, the open source project I developed to send large amounts of text announcements.

Sections

Design Overview

The code for Batch SMS can be found on GitHub at csu/batch_sms. It is written in Python and uses the Twilio API for sending text messages (but it is written in a way such that Twilio could easily be swapped out for an equivalent service).

Twilio *rate limits message sending to one text per second* per phone number, so the system is also capable of load balancing a text across multiple "from"

numbers (the numbers you own, from which the texts are being sent).

While we do want to distribute the texts across multiple “from” numbers, we still want each receiving number to always receive texts from the same sending number. Thus, the system maintains associations between “from” numbers and “to” numbers and persists them in a database. Any database supported by SQLAlchemy can be used with the system (this includes SQLite, MySQL, PostgreSQL, and MariaDB).

Installation

First, install all necessary requirements:

- Python 2
- pip (for installing Python dependencies)
- A SQLAlchemy-compatible database (SQLite is sufficient)

Now, get the code by cloning the git repository:

```
git clone https://github.com/csu/batch_sms.git
cd batch_sms
```

Install the necessary Python dependencies (optionally, enter a Python virtual environment before doing this):

```
pip install -r requirements.txt
```

Now you’re good to go!

Basic Usage

Create a new Python script or enter the interactive Python REPL. Now enter the following:

```
from batch_sms import BatchSMS
from batch_sms import AssociatedBatchSender
from batch_sms import TwilioSender
from secrets import TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN

sender = TwilioSender(TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN)
batch_sender = AssociatedBatchSender(sender)
client = BatchSMS('test.db', batch_sender, auto_associate=True)
```

Here's what this does:

- The first few lines `import` what we need from the Batch SMS module.
- We then create a `TwilioSender`, which knows how to send messages using Twilio.
- We then create an `AssociatedBatchSender`, which knows how to send large numbers of messages while obeying to-from number relationships (we pass it the `TwilioSender`, which it will use for the actual message sending).
- The components are separated in this way so you could swap out the `TwilioSender` for another object that implements the same interface (e.g. if you wanted to use another API besides Twilio to do the actual SMS sending).
- Finally, we create a `BatchSMS` object which will manage data persistence.
- The `auto_associate` option will automatically associate newly added "to" numbers with "from" numbers. It attempts to distribute "to" numbers as evenly as possible across "from" numbers.

Now, we can create a "subscription list" which consists of a list of recipients. Give it a name, like `Hackers`:

```
sub_id = client.create_subscription_list('Hackers')
```

Add your "from" numbers from which you will send messages:

```
client.add_from_number('+15005550006')
```

Now add your “to” numbers and include them in the subscription list you just created:

```
client.add_to_number('+15005550010', subs=[sub_id])
client.add_to_number('+15005550011', subs=[sub_id])
client.add_to_number('+15005550012', subs=[sub_id])
client.add_to_number('+15005550013', subs=[sub_id])
```

We can now send a message to all recipients in the “Hackers” subscription list like so:

```
client.send_to_subscription(sub_id, 'Hello World')
```

If you want some feedback printed to `stdout` (or want to do anything after the messages are sent), you can pass in a callback:

```
def callback(payload):
    print payload

client.send_to_subscription(sub_id, 'Hello World', callback=callback)
```

Normalizing Numbers

Depending on how you’re collecting phone numbers from recipients, you may end up with phone numbers in all different formats or numbers that are entirely invalid. I wrote a simple Python script that normalizes numbers from a `csv` file into the `+1XXXYYYZZZZ` format that Twilio likes while discarding invalid phone numbers (e.g. ones of improper length).

Unsubscribing

Conveniently, Twilio already handles unsubscribing/opt-out and resubscribing for us. Users can text STOP, UNSUBSCRIBE, and other commands to opt-out of notifications. Users can text START and YES to resubscribe. For the full list of commands included by Twilio, see their article on the topic.

Incoming Messages

By default, Twilio will not do anything with incoming messages and will reply to them with a message telling you to setup your incoming message handler. I quickly threw together this short PHP script that will receive incoming messages, forward them to another number (e.g. my phone), and then reply to the sender of the message with a generic response (e.g. “email the DubHacks team with questions”).

Conclusion

All in all, the system was very effective at DubHacks. It took around one minute to send out 700+ text messages load balanced across five sending numbers. The success rate for message sends was high and it felt great being able to hit enter on my keyboard and seeing hundreds of people appear at an activity or event shortly after.

I wrote this whole thing in one go. If you find any typos or errors, please contact me. Feel free to message me with any questions or concerns.

Comments

Karan Goel • 6 years ago

Is there a queue or built-in rate limiting? It'd be nice to send a bunch of texts to the library and then it send the texts with necessary delays and what not.

1 • Reply • Share ›

Christopher Su Mod Karan Goel • 6 years ago

It sends the requests to Twilio as quickly as possible, but then Twilio queues texts and sends them according to the rate limit.

• Reply • Share ›