DoucheTag: Twitter Harassment Detector

By: Chris Jepeway, Amitabha Karmakar, Ricardo Frank Barrera

Contents

[Overview](#h.gjdgxs)

[Justification](#h.30j0zll)

[Tech Stack](#h.1fob9te)

[Tech Specs:](#h.3znysh7)

[Initial Performance Evaluation](#h.tyjcwt)

# Overview

Our group created a service to identify potential twitter harassers. DoucheTag is built upon several technologies so that we can score a stream of tweets and use our Latent-Dirichlet Analysis Model to classify as likely harassment or not. The tech stack uses Kafka for the storage, Spark for streaming analytics, calls Twitter APIs to stream, analyze. This allows us to stream and persist tweets in case we need to update

## Justification

Twitter harassment is alive and real. Recent prominent examples are listed below:

• GamerGate: <https://en.wikipedia.org/wiki/Gamergate_controversy>

• see Wired: <http://www.wired.com/2014/05/fighting-online-harassment/>

This issue is prevalent and serious and Twitter is actively fighting harassment. (Twitter's abuse policy: <https://support.twitter.com/articles/20169997> )

# Data Sources

Our data is only using the Twitter streams to build the harassment model. The Twitter Tweet JSON has many fields and a lot of information such as:

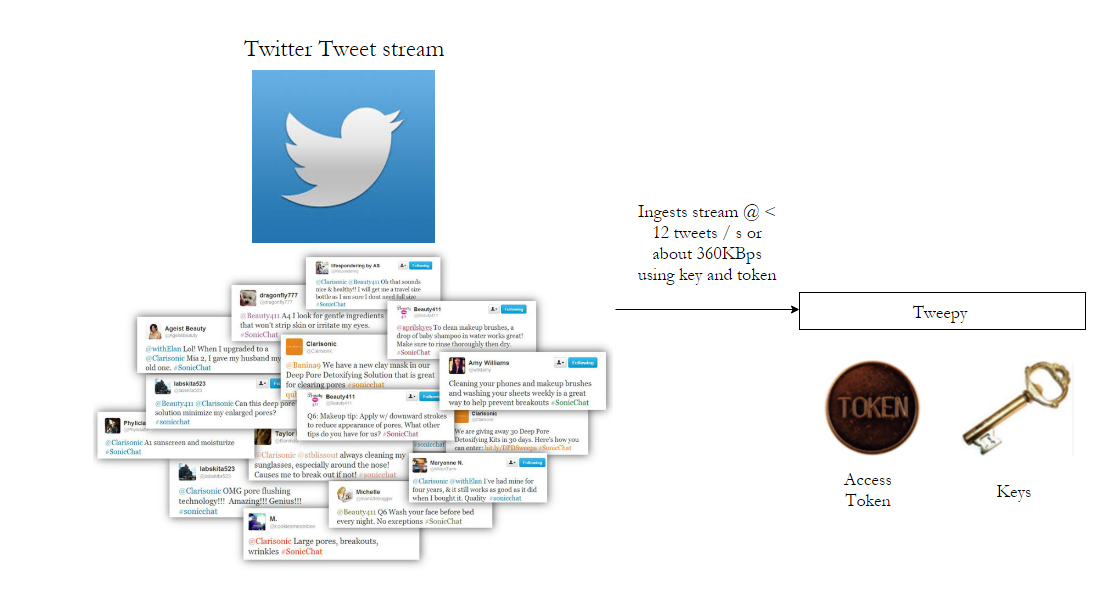
* Coordinates
* Created\_at
* Favorited
* **Text**
* etc

We’re only using the text portion of the tweet for the LDA model.

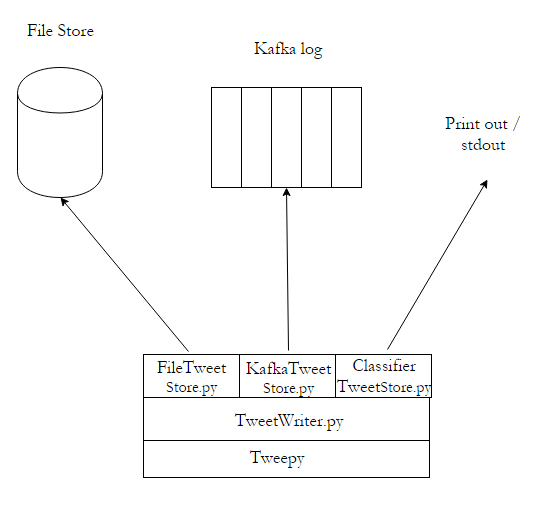
# Architecture

first version

Twitter stream ingestion using Tweepy with Twitter App Token and Key



Stream storage in one of several options (local, Kafka, or print)



Classification:

# Data Acquisition

Data acquisition is done via Tweepy.

1. Tweepy.stream establishes the streaming session using the auth credentials
2. StreamListener receives the messages
3. Incoming tweets are written to the File store and the Kafka log

# Initial Performance Evaluation

## Tech Specs:

* Data Velocity
  + 100k Tweets / 3GB per day; 12 Tweets per second at about 36KBps
  + Each Tweet is about 3KB in size with a pull rate of about 12 per second (the max rate)
* Data Acquisition
  + We will pull random samples from the Twitter stream until we become interested in a specific user because of a high-scoring tweet. A separate part of our Storm App will then pick multiple tweets from the user, score the account, and deciding to block or not.
* Data Analysis
  + The initial version will use a basic sentiment analysis dictionary and perform a generic L1 scoring approach and simply threshold
  + Once everything is in place, we will consider more sophisticated approaches to classify users
  + The JSON will be parsed for tweet content to score, and user id for identification, which can be used for targeted pulls if interest increases
  + Users will then be added to a blocktogether list which people can subscribe to for blocking
  + More sophisticated approaches may require us hand-tagging tweets, and for that we may use Amazon Mechanical Turk to scale.
* Data visualization
  + We will not be leveraging visualization for the project. Our visualization is basically twitter block list.
* Testing on Networks
  + LAN: Wifi @ 54 Mibps
  + WAN : Residential U-verse
    - speedtest.net : 15Mbps down, 2 Mbps up
* System Latency
  + We do not have end-to-end latency estimates yet, but our goal is to minimize the time to identifying a harasser after an initial high-scoring tweet comes to our attention
* Scalability
  + Our simple approach using sentiment dictionary values should scale well, but the more accurate and sophisticated classifier may not scale easily due to the cost of tagging tweets manually for training.
* Machine Learning / Sentiment Analysis
  + We have used LSI topic modeling for doing similarity analysis of incoming tweets to existing harassing tweets.
  + We initially tried to use gensim’s LDA implementation to make the classifier learn topics, but online training didn’t work on that. Hence we had to shift to LSI.
* Scalability
  + Our simple approach using sentiment dictionary values should scale well, but the more accurate and sophisticated classifier may not scale easily due to the cost of tagging tweets manually for training.

# Architecture Diagram

