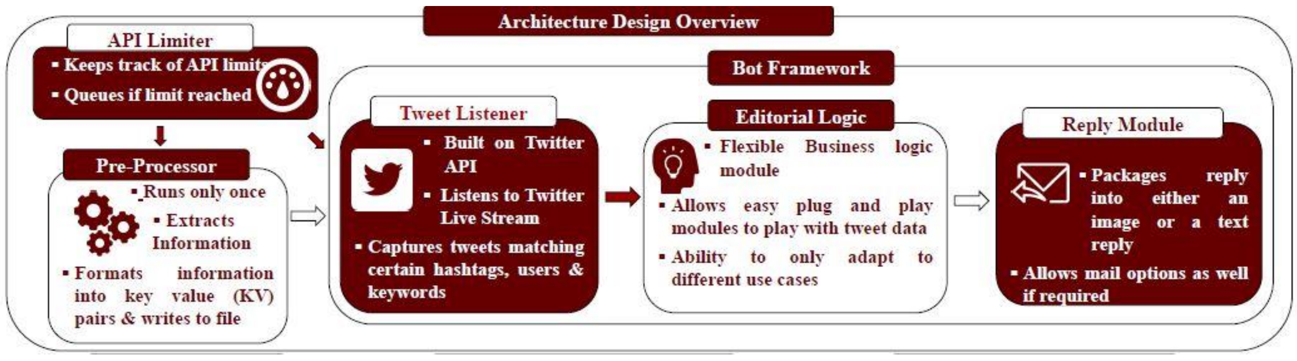
Twitter Bots – Architectural Design Document & Bot description

Abstract

The following document gives a brief overview of the Architecture developed for the creation of bots on Twitter. It highlights the modular framework with description of each module in depth along with its capabilities. It also describes 2 bots – **Protest Bots** and **Custom Search Bot** developed using the same framework. Both depict 2 different approaches of bot creation using the same framework thereby displaying its flexibility. The 2 bots are based on different paradigm – Protest Bot based on the concept of Information dissemination and Custom Search Bot based on the concept of Information aggregation.

Bot Framework

Designing a framework which can be used for building a bot can be made easy by breaking the functionality of the bot into modular components. The basic functionality can be broken into 3 main modules – **Input, Logic and Output**. Building on this, here are the Architectural design decisions taken in designing the framework


Architecture Design Overview
API Limiter
Keeps track of API limits
Queues if limit reached
Pre-Processor
Runs only once
Extracts Information
Formats information into key value (KV) pairs & writes to file
Bot Framework
Tweet Listener
Built on Twitter API
Listens to Twitter Live Stream
Captures tweets matching certain hashtags, users & keywords
Editorial Logic
Flexible Business logic module
Allows easy plug and play modules to play with tweet data
Ability to only adapt to different use cases
Reply Module
Packages reply into either an image or a text reply
Allows mail options as well if required

1) Stream Listener Module (Input): Gathering information on Twitter which serves as input is one of the key tasks of a bot. Thus, the 'Tweet Listener' module serves the purpose by providing a tweakable module where one can specify what you want to listen to. One can listen to multiple #hashtags or particular keywords or tweets from a certain user or even tweets based on certain locations. This module can on itself be useful to just listen to a stream based on certain filters and can help in capturing real-time tweets.

2) Editorial Logic Module: This module is where one can define the business logic of the bot. This comes straight after the Stream Listener Module. The input to this module is the Tweet JSON. The module supports easy tweet parsing and sub-filters which allows further filtering of the tweet data. One can define internal function which can help in further enhancing module functionality.

3) Reply Module: The reply module comes as the last module in the workflow. This module allows for east reply ability to the users. One can reply using the direct @tweet option or even do something more radical such as email the results back to a person. An internal module in the reply module is the **Text-to-Image** convertor module which helps in converting the text data to image (it wraps if it has multiple lines) thereby making it easier to tweet out messages that are more than 140 characters by sending an image as a reply. It also has an internal optional '**Probability'** module which helps in allowing user to sample the reply. One can tune in a probability factor which helps in deciding whether a reply should be sent or not.

The above 3 module defines the bot framework. On top of this framework are 2 other modules that give the framework more functionality.

1) API Limiter: This is a module that keeps a track of the API limits imposed and helps in stopping the bot before it exceeds the API limit. As of now, the most important API limit a bot has to adhere to are imposed by the Twitter API. The bot will time out once it hits the API limit set in the limiter.

2) Pre-processor Module: This module helps in pre-processing any data or information one wants to before the bot is invoked. This could be considered as a subordinate bot which helps in things like scraping information from someplace and outputting it in a file or inputting information to a file once. This is useful for information that needs to be pre-loaded and thus works once per invocation of the bot (it can be called again periodically to gather new information by including logic in the Tweet Listener).

Bots

Based on the above 2 frameworks, 2 bots were built.

1) Protest bot: The protest bot functions as an information dissemination bot. The motive is to find people who are tweeting about facts that happened on the current day and then send them more facts. These facts are extracted each day from Wikipedia. Following are the modular functionality

a) Pre-Processor Module: This module helps in scraping the facts from Wikipedia. Here, all the events that happened on the day are extracted from Wikipedia. There is a filter internally with an array of keywords. Each event is then parsed to find a keyword from the above list and if present, the event is added to a file. In the end of the scraping, we are left with all the events of the day from Wikipedia that have at least one keyword from the filter set. All of the facts are written to a file which will be further used by the Editorial logic module.

b) Tweet Listener: In this case, the tweet listener listens to a particular hashtag (eg. #ThisDayInHistory) using the Twitter streaming API and on each emit of a Tweet with the hashtag, it passes the Tweet JSON data to the Editorial logic module.

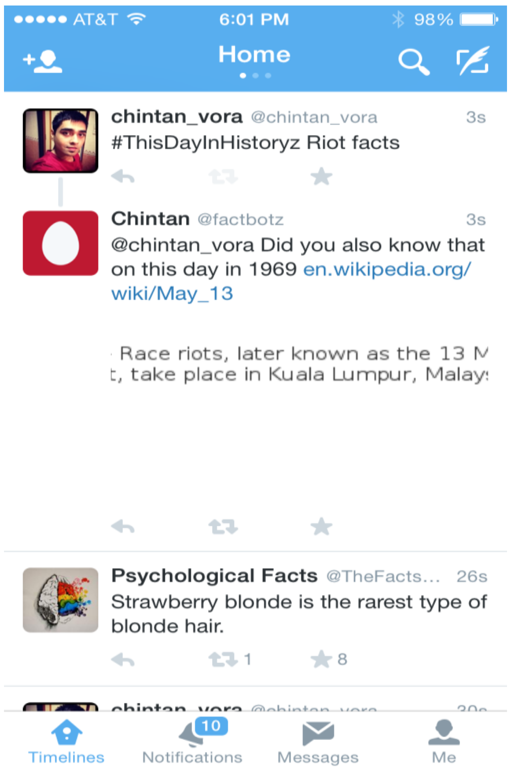
c) Editorial Logic: This module gets the Tweet data as an input. It has an internal probability module where a given threshold is set. It firstly generates a random number. If the number is less than the threshold, further operations are allowed else we return the control back to the Tweet Listener. It is then followed by an internal sub-filter which further allows one to filter the tweet on the body text (If it contains a word from a set of keywords, then only will it allow the control to move forward). If the probability criteria are met & the text criterion is met, then the logic module retrieves a random fact from the input file of facts generated by the pre-processor module. This fact is then sent to the Reply module.

d) Reply Module: The reply module gets a fact as the input along with the Tweet data. The module has an internal Text-to-Image convertor which converts text data to image. This image is then tweeted out back to the original user using Twitter update status API.

e) API limiter: The API limiter acts as a gatekeeper. It regularly checks if the API limit isn't crossed (more than 150 API calls per 15 minutes) and if the criteria is exhausted, it halts the bot for 15 minutes.

Issues faced: Creating the bot had a lot of challenges namely working with a stringent API limit. This was the biggest factor which led to design changes. Along with the API limit, the inability to reply a message with more than 140 characters lets to creation of new approach of converting the text to an image.

Retrieving facts in the pre-processing phase also had a couple of challenges since as of now, the bot just scrapes the Wikipedia page due to the lack of decent official Wikipedia API.



chintan_vora @chintan_vora 3s
#ThisDayInHistoryz Riot facts

Chintan @factbotz 3s
@chintan_vora Did you also know that on this day in 1969 en.wikipedia.org/wiki/May_13

Race riots, later known as the 13 M
t, take place in Kuala Lumpur, Malay

Psychological Facts @TheFacts... 26s
Strawberry blonde is the rarest type of blonde hair.
1 8

In the above figure the bot is listening to the hastag **#ThisDayInHistory** & internal filter is looking for keyword ‘**riot**’. Once both are found, the bot replies with a fact in the form of an image.

2) Custom Search Bot: If a person wants any information, their first step is to ‘Google it’. Google also provides a powerful implementation of Custom Search Engines where users can build their own list of target sites and Google will search only on those sites. This allows a more narrow focused way of searching for news. The motive of the Custom Search Bot was to find leaks from the Apple supply chain. The implementation included adding all the websites of the suppliers of Apple products (list provided by Apple) into a custom search engine (CSE). There is also the ability to tweak the File Format of the results and the amount of time since the search result popped in Google indexes. Once the pre-processing is set, the bot waits for users to Direct Message the query it on Twitter. Once a message arrives, the bot will extract the query and send it to the CSE. The CSE results will be then emailed to a registered email id. The bot is built on the same framework mentioned above:

a) Pre-Processor module: Here, the pre-processing is manually adding the list of sites to a Custom Search Engine. Also, once the list is added, one needs to get the Custom Search Engine ID (available in the CSE developer console) and add it to the input.txt file. Along with that, user also can add in various parameters such as the file type (eg. pdf,pptx etc) and the number of days/months window of the result (d1 or m1 or y1 for 1 day/month/year). All the data is populated in the input.txt file which will be read by the logic module.

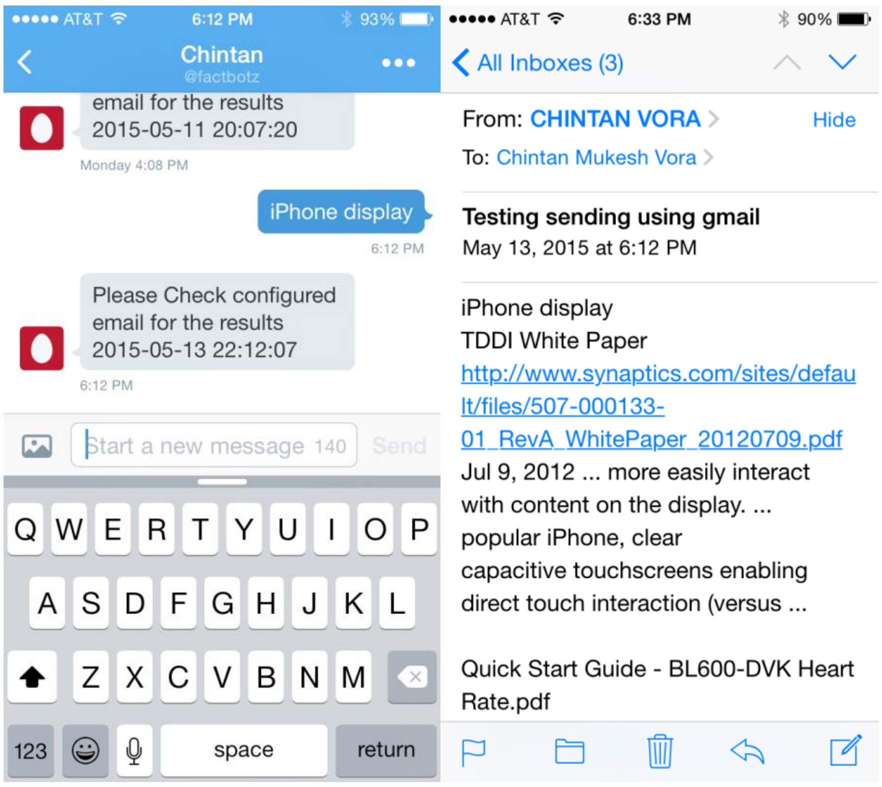
b) Listener Module: The Listener module in this case is configured to listen to direct messages from different users. Thus, the bot monitors its own account. When someone messages the bot directly, it considers the message as a query. This is sent to the Editorial Logic module.

c) Editorial Logic: In this case, the logic module extracts the information from input.txt (which was filled in the pre-processing stage) and gets all the fields that are not null. It then packages the fields into a string which will be a GET request call to the Google Search API. The API call returns the data which is buffered temporarily and sent to the reply module.

d) Reply Module: In this case, the reply module works as an email dispatcher. Here, the reply string received from the Editorial Logic module is sent to a pre-configured email id. Thus, all the search results are sent to an email id for easier access.

e) API Limit module: In this case, the API limits are set in the code itself so that it doesn't exceed the limitation imposed. It is done by the Twitter Bot which queues the queries into a buffer and only replying to them with 1 minute windows. Also, the queue has a max buffer of 150 messages and any more will be disregarded. The assumption in this case is that no one will put in more than 150 queries in 15 minutes.

Issues faced & things learnt: The Custom Search Engine (CSE) of google is still in BETA and thus has a lot of bugs. Sometimes, the GET request just fails and returns a 'Back-End Error'. Also, the search results themselves are questionable since it doesn't seem to bring out all the results. As the number of sites increased, in few cases the search results went down which denotes that the CSE isn't indexing properly and applying weights to the list of sites included (giving more preference to ones added latter). It is advisable to add in few sites (less than 20 if possible) and thus decrease the Sound to Noise ratio. Looking for things is difficult as always and such a bot is useful for getting focused results which is a behaviour like following curated RSS feeds but for search results.



email for the results
2015-05-11 20:07:20
Monday 4:08 PM

iPhone display
6:12 PM

Please Check configured
email for the results
2015-05-13 22:12:07
6:12 PM

From: CHINTAN VORA
To: Chintan Mukesh Vora

Testing sending using gmail
May 13, 2015 at 6:12 PM

iPhone display
TDDI White Paper
http://www.synaptics.com/sites/default/files/507-000133-01_RevA_WhitePaper_20120709.pdf
Jul 9, 2012 ... more easily interact with content on the display. ... popular iPhone, clear capacitive touchscreens enabling direct touch interaction (versus ...

Quick Start Guide - BL600-DVK Heart Rate.pdf

In the above case, the query is a Direct message to the bot on Twitter. The search results are then sent to a pre-configured email id set. Please note, the configuration has been set to return only results of the PDF format hence the result seen in this case is a PDF.