Explanation Twitter Bot @OwenLussac

Abstract

The purpose of this project is to model a human-like twitter bot that could easily interact with the twitter community without being framed as a robot. Thus, it would enable researchers to experiment a lot on Twitter with a programmed bot. To be able to do so I am going to analyze on the literature the behavior towards tweeting things such as frequency of tweets. Then I am going to develop content wise issues and show two technics of generating tweets: free-context sentence with bigrams model and summarization of press articles. What's more, a part of it will deal with extra tweeting activities such as direct messages or following behavior.
To keep in mind that making the bot as human as possible, I had to give him attributes such as birth date, interests and occupations. As a result, I kept consistency in the tweets he makes.

Development

Libraries used: Numpy, Twitter (https://github.com/sixohsix/twitter), Random, NLTK

- **Creation of a Twitter profile**: https://twitter.com/OwenLussac
  Owen Lussac, 37 following and 1 follower, Profile pic of young white boy. Born in 1991 January the 5th and student at Imperial College London. Interested in Technology and startups.

- **Authentication**: Creation of App via https://apps.twitter.com/app/9025682/keys and get Token, consumer key and consumer secret key

- **Time to post or not**: First part of the program is to know whether or not it is time for the bot to tweet.

Inputs :

- Poisson law to compute time to post
- n: random number between 0 and 99 to compare to the probabilistic law
- Time to generate n : here I assume that it is 20 minutes, no information found on literature for that
- Occurrency: 13 tweets a day from 9am to 10 pm. Intuitive assumption

To compute this, I chose to use Poisson Law (poisson.py). This is very used in Parisian supermarkets to know whether or not they should switch the cashiers regarding the arrival of clients.
To compute Poisson law, I've also define Factorial function (fact.py).

As I've seen on many articles, and it actually makes sense, people tweet in average once an hour during 13 hours from 9am to 10pm. As a matter of fact, they tweet in average 13 tweets a day. Not necessarily every hour but on the whole day it happens to be once an hour in average. As a result, it was obvious not to only compute a timer that post a tweet every 60 minutes but to make it seems more human and less automatic. I opted out for a Poisson law and the program is actually taking a random number every 20 minutes and compare it to the probability of my law (with parameter the variance equals to 13 (13 hours * 1 tweet per hour). Every time it is greater it continues with the rest of the programming towards posting behavior and otherwise it does nothing.

In the whole day it will normally be 13 tweets (not during the night obviously) but at « random time of the day ». Improvements of this would be to analyze if student like Owen tweet more during weekdays or weekends.

- **What to post? (Post.py)**

Here we have several options and it can actually be unlimited option of tweeting behavior due to unlimited complexity of human being. I'll stick to programming 3 of them:

  - <u>Human-being sentences (Tweet_conten1.py)</u>

Here I am calculating on database of tweets (the calculus should be done on a more comprehensive database) the probabilities of appearance of a word considering the previous word. In a nutshell, I am calculating the probabilities to have such combination of words (characteristic of a human being) and then observing patterns. I could not make it work yet and for the moment the tweet that could result from this would not make sense but the combination of the words would tend to be as if a human wrote it. To solve the problem of the coherence and the syntax of the tweets, a large database has to be involved. This is basically computing a n-gram model where a word in position i depends on the words in the previous n-1 positions.
In my work, the sentence is generated using **NLTK library** and nltk.bigrams function.

To generate these tweets, we have to formalize a grammar. In our case, we could analyze thousands of tweets and take out general sentence architecture but for this project I am for now consider an intuitive grammar which is:

Sentence = Verb + Determinant + Noun + '!' + link to an article

```
S -> VP NP    Starting state S
NP -> Det N
PP -> P NP
VP -> 'Check'
Det -> 'this'
Det -> 'it'
N -> 'article'
N -> 'link'
N -> 'news'
P -> 'here'
```

Tweet_content1 returns then a context-free sentence generated through NTLK grammar model + a link to an article. As it is context-free, we could imagine any kind of tweets, as long as the grammar is enhanced with more words and syntaxes.

- Summarizing press articles

Tweet_content2 will summarize an article by quoting it or just giving important words of the article. While tweet_content1 generates free-context sentences, here, it has to be contextualized since the tweet has to be about a specific article.
First choice would be to use a python script summarizing an article (that's chat I did for my bot). A good example of summary tool programmed in Python is here.

The other would be to use Twitter Card tool available on their Development website: https://dev.twitter.com/cards/overview

David Walsh explains on it works here: http://davidwalsh.name/twitter-cards

- **Extra activities on Twitter besides posting**

  Indeed, besides tweeting, users have other occupations on Twitter such as retweeting, follow, reply, put in favorites, etc. I will concentrate for my bot on one of them which is:

- Sentiment analysis:

The bot will analyze the tweets of his timeline very one hour (time of activity and thus period at which a human looks at his feed) and by an analysis of the tweets will be able to DM the user shaving tweeted nice things. This idea came to my mind reading an article by Columbia researchers : http://www.cs.columbia.edu/~julia/papers/Agarwaletal11.pdf

We could train a Bayes classifier for this and compare words of the tweets to the classifier. The work consisted in, first, separating each word of a tweet and then analyze each word to see if they are positive or negative (using the classifier). At the end I chose to compare the counts of positive words and of negative words. This is a very basic debut and to make it even more efficient we should focus more on the syntax of the sentence.

Conclusion

We have created a bot that has some human characteristics and that interacts with others considering their tweets.
Yet, there are others criterion that should be taken into account such as:

- Coherence of the tweets: we should program something that is able to take into consideration the previous tweets of Owen and do text mining on them in order to tweet every time with the same syntax characteristic of Owen's behavior while writing or even keeping the same sense of humor and ton (if he is used to tweet with exclamation points or not). This would involve to program a function that analyze previous tweets and take them in a dictionary that grow with the amount of tweets tweeted.

- Topics of tweets: if Owen tweets about technology and is fond of it, we should be consistent enough and not make him tweet something about candies or cook.

- Consistency on time: make sure we keep the same pace of tweeting and not making Owen tweets ten times a day during weekdays and never during weekends for a month and do the contrary next month. It would make people doubt on its human behavior.

In general, the whole work is on giving relief to Owen as a human and attributing him large set of criterion such as 'funny', 'curious', etc.

Resources

- http://stackoverflow.com/questions/5957495/where-and-how-can-i-install-twitters-python-api

- http://www.dototot.com/how-to-write-a-twitter-bot-with-python-and-tweepy/

- http://python.jpvweb.com/mesrecettespython/doku.php?id=loi_poisson

- http://www.cs.columbia.edu/~julia/papers/Agarwaletal11.pdf

- http://meta-guide.com/software-meta-guide/100-best-github-automatic-summarization/

- http://streamhacker.com/2010/05/10/text-classification-sentiment-analysis-naive-bayes-classifier/

- http://nltk.org/book/ch02.html