

Humans susceptibility to follow social bots

Anders Wessberg
s103477

Benjamin Maksuti
s103449

Kevin Voss Sjøbeck
s103451

1. INTRODUCTION

Have you ever wondered why all your friends on twitter have more followers than you? In this project we will create a twitter bot, and try figure out which people that are more likely to followback and which who are not. Our bot is named Nikki Masti and is a beautiful girl with a dream of becoming a model. To make this project simpler, we only follow people from San Francisco. The bot will try and act like a human being using data machine learning with python and an EC2 server using crontab. We then keep track of who we follow and who follows us back so we can analyse the profiles using NaiveBayesClassifier, and hopefully find out what features that makes people follow us back.


Our bot Nikki Masti is a young female in her twenties, who is on a business trip till the end of the year. She likes to take pictures of herself, and is found to share these on twitter. Also she is a moderate twitter user, who like to help other twitter users, spread their messages by retweeting their tweets. As in contradiction to being a young and inspiring model for younger girls to look up to, she can sometimes be a real devil, especially if she finds fights for somebody over the last pair of popular shoes on discount.


2. INTERVENTIONS

2.1 Our original tweets

In our creation of our original tweets we tried to match the personality of our bots. Furthermore we made two tweets during most of the interventions, and at the times we made two tweets, it was done in such a manner, so the second tweet of the day, would try to follow up on the first tweet of the day. A very good example of this is the tweets of the blackfriday intervention, where we first post that our bot is very excited to go out and shop for dresses and shoes during black friday. During the second tweet we follow up in the first, by posting a picture of the shoes bought, and we capture the personality of our bot, who sometimes can be a real devil while she is shopping. The personality is captured

by the text, where we write "Got these today at half price :D but i had to push in line to get them"

 **Nikki Masti** @MyNikkles · 28. nov.
Time to get some new dresses and some shoes :D #blackfridaystories
7 15

 **Nikki Masti** @MyNikkles · 29. nov.
Got these today at half price :D but i had to push in the line to get to them
#blackfridaystories #sorry



3 12 Vis flere billeder og videoer

2.2 Timing Strategy

Our timing on the tweets, was in such a way, so that we usually made two tweets per day, one in the morning and another tweet once more in the evening. However since the tweets needed to be written manually we saw no reason to create a script which could post it for us. By doing it manually we also gained the advantage of having the tweets look like they were human made.

3. IMPLEMENTATION

3.1 Intervention

For each intervention we had to favourite and retweet to a specific hashtag. This concluded in three features that needed to be implemented. An automated favouriter that favourites all tweets with the specific hashtag, an automated retweeter that retweets the first four tweets in the timeline with the hashtag and retweet 15 tweets that was not made by a bot from the class.

We made a single script to withhold all of these features. Firstly we needed to find the tweets, this was done by the search.tweets with a query of the hashtag, count at 100 and

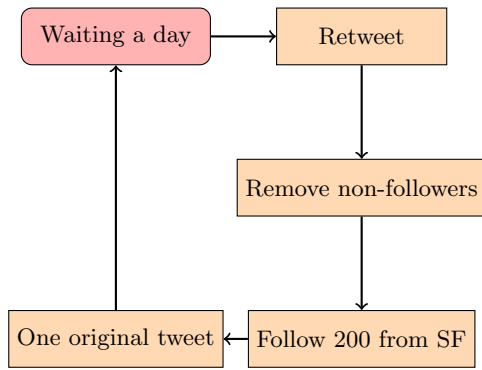


Figure 1: Flowchart of the daily routine

the language at English. We then go through every tweet and first we use the *favorites.create* with the tweets ID, then we use a counter that counts to four, so that the four first tweets get retweeted using the *statuses.retweet* with the ID of the tweet. Lastly we check if the tweet belongs to one from the class. If it does not, we then retweet it again using *statuses.retweet*, and this is done up to 15 times using a counter. We also took advantage of twitter's fault messages, so that if we tried to retweet the same tweet, it would make a fault, and be caught by a try-except and the script would keep on running.

3.2 Daily routine

Our daily routine started with a retweet, here after we removed all who don't follow us back from the day before, then we started following 200 people and lastly we ended the bot's day with an original tweet. The whole routine is shown in Flow Chart 1.

Each step in the routine was made in a script for themself.

3.2.1 Retweeting

Every retweet that we wanted to make should fit the personality of the bot. We therefore used *search.tweets* to find tweets that used a specific query, as fashion or model OSV. We then compare every tweet and see which have been retweeted the most, and we retweet the same.

3.2.2 Following people from SF

Before we can follow any from San Francisco, we first need to find people living there. We therefore use the twitter stream to retrieve all tweets made within the area of San Francisco, then we go through each tweet to make sure that it is not made by one from the class. Some other measures we make are that it should not be a retweet, because if it is we can't be sure that it was retweeted from San Francisco, and we use our own criteria to check if the tweet was made by a human. One of the criteria is if the person making the tweet has an average of one tweet for the last 20 days.

If they pass we follow them, and save their screen name in

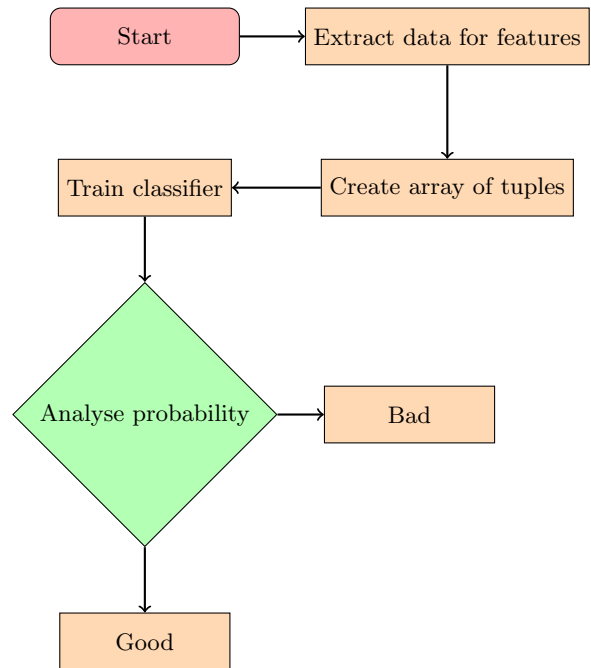


Figure 2: Flowchart of the data analysis

a text file. The streamer continues until 200 people's are followed.

3.2.3 Remove non followers

23 hours and 30 min after we have followed 200 people, we unfollow those who did not follow us back. This is why we save the people's screen name. To see if they have followed us back, we therefore use *friendships.lookup*, but this method only allows 100 screen names each run. This can be done by making two strings each with 100 names that are comma separated. Then we go through each person and check if it says "followed_by" in "connections".

3.2.4 Creating an original tweet

Creating an original tweet is rather simple, we first save all the tweets we want to make later in an array, then we use *statuses.update* to tweet. We choose randomly one of the tweets in the array, and give the latitude and longitude of San Francisco, with a little noise so that it looks that we travel around.

3.3 Machine Learning and data analysis

3.3.1 Features

To use a classifier you need to have a single feature or several, that divides all data into groups. For our classifier we have chosen three features, Friends count and Followers count of the profile, and the month and year the profile

was created. Because you are able to follow and get infinite friends, we set up some criteria's.

- From 0->100
- From 101->250
- From 251->500
- From 501->750
- From 751->1000
- From 1001->2500
- From 2501->5000
- Over 9000¹

3.3.2 Extracting data to train the classifier

To be able to train the classifier, we first need to gather data from two groups of peoples from San Francisco, those who followed us back and those who dint. So from our daily routine, we have stored screen names all of those who followed us back(good), and all of those who dint(bad), and we got 127 good people, and 1108 bad people. As with all tests you need an equal amount of samples, we can therefore only use 127 of the bad peoples.

For each of these 254 people we extract their follower count, their friends count and the month and year of creating their profile. For each profile we use their information to group up the features in a dictionary, and make an array of tuples, consisting of the dictionary and if they were good or bad as a string.

We then train the classifier using `nlk.NaiveBayesClassifier.train` with the array of tuples.

3.3.3 Extracting analytic data

After we made the classifier we started to gather all screen names of those we tried to follow. So we could test how correct our classifier was.

So for each screen name we would extract their friends count, followers count and month and year of creating their profile and put in a dictionary. Hereafter use `classifier.prob_classify` with the dictionary as parameter, and then get the probability for that to be a good person. We can then use that probability to set a limit for how sure the classifier should be, before we save their screen name as someone who would follow us or not.

4. SUSCEPTIBILITY ANALYSIS

4.1 Statistics

Now that we have data to classify and the classifier, we are able to classify each profile, on how probable the person will follow back. We do this by going through the list of people we have gathered, then we set up 9 probabilities going from 55% to 95%. We do not care about probabilities under 55%, because we want the classifier to make better qualified guesses than a normal human being.

We then store the number of people the classifier have guesses, as who would follow back(good), and who would not(bad).

¹It is actually over 5000

Then we check all of the profiles on how many followed back, to see how many times the classifier made a wrong guess. In the end we summarize how many times the classifier made a right and wrong guess.

Probability	Good		Bad	
	Guess	Wrong	Guess	Wrong
95%	4	3	954	108
90%	14	11	944	106
85%	48	42	910	103
80%	78	67	880	98
75%	129	112	829	92
70%	208	179	750	80
65%	267	231	691	73
60%	327	284	631	66
55%	385	337	573	61

We then divide the number of wrong with the guess, for both good and bad.

Probability	% of wrong	
	Good	Bad
95%	75	11.3
90%	78.6	11.2
85%	87.5	11.3
80%	85.9	11.1
75%	86.8	11.1
70%	86.1	10.7
65%	86.9	10.6
60%	86.7	10.5
55%	87.5	10.6

Some other numbers we are able to look at are how precise the classifier are, this is done by creating a test set. After we create the test set we use `nlk.classify.accuracy(classifier, test_set)` to get the accuracy of the classifier. But the data we normally only use for the classifier, will be divide so that 70% go to the training set for the classifier and 30% goes to the test set. We run this 100 times, with the data being shuffled each time and take the average in the end. This tells us that the classifier is 51.603% accurate, which is not a lot, but still 1.5% better than a normal guess.

4.2 Theory

Which tools from the class (network science, natural language processing, machine learning) have you used?

To classify the data, we use a simple naive bayes classifier. However by using this classifier we make the assumption that all of the features are independent of each other. By independent of each other we mean, that there shouldn't be any correlation between the chances of the different features. However in our case we expect to have a correlation between the followers and how many other users the user follows. We choose to disregard this correlation and make the assumption, that all of our features are independent. This assumption can be made because we are dealing with real people.

Creating features are the most important when creating a classifier, without features that utilize the date the most,

your classifier won't give you a great answer. In our case we use followers and friends count, and the date of the profile creation. We also wanted to use the age, the gender and name of the person using the profile. But these are not shared through Twitter, but could have been good to have as features.

4.3 Analysis

Before we can talk about which probability, with our dataset, is the most efficient one, we need to look at which is worst, getting a wrong with those we thought was good or getting a wrong with those we thought was bad. Getting a wrong when we think they are good, means that for each wrong out the 200 persons we follow a day, will not follow us back, and therefore we will be getting less people following us a day. Getting a wrong when we think they are bad, means we won't follow them, even though they would follow us back, this number needs to be as low as possible, because these would be a potential follower lost forever.

This is why it is more important to have a low wrong in bad, than a low wrong in good. The percentage of wrong in good is rising when we lower the probability, this makes a lot of sense, because we allow more risky judgement. But lowering the probability also means a lowering of the percentage of wrong in bad which is a good thing. The lowest percentage of wrong in bad is when the classifier is at a 60% probability. Also at this classifier probability the wrongs in good is lower than the 65,75 and 85%. So with our dataset of 254 people, the 60% classifier gives the lowest loss of people who would follow us, but also lets 86.7% wrong through the filter.

The reason we look at how many times this classifier makes a wrong is because of its low accuracy, that was 51.6%. But if we looked at how many times it made it right, then we would have to only follow those that have a 90% probability, because there is a 25% change of being right, and it's the highest of them all. But this would only be 1 person out of 958, so we would need to run the daily routine 5 days to get 1 follower.

This is because of the low data set of only 127 good and 127 bad people in our data set, and that the range of which the features put the people in, is way too unspecified. This is why our classifier has such low accuracy and makes that many mistakes.

5. CONCLUSIONS

What have you learned from the course? Would you do anything differently? Did your strategy work? Which advice would you offer to prospective bot designers.

Throughout the course of this project, we have created a Twitter bot, using the Twitter REST API. With this bot we were able to automatically follow people living in San Francisco, and create original tweets fitting our bot's personality. We also retweeted tweets, that had specific keywords in them. Because Twitter doesn't allow you to follow an infinite amount of people a day, we created a classifier to specify if the people we found would have a high probability to follow us back. Due to the low accuracy of the classifier, we

would still follow people that doesn't follow us back, and we would ignore some that had the potential to follow us. But if we said that the people we tried to classify only needed to have a 60% probability to follow us, we would get a higher chance of following people, that would follow us back.

A thought we had was when we followed people, that we sent them a private message, that might have given them more encouragement to follow back.

To other bot designers we would say that a great idea is to create a profile with interests that are general and to be a girl.