Random Forest classification of Twitter users to detect features linked to bot susceptibility

[DTU 02805 Social Graphs and Interactions]

Ulf Aslak Jensen
Technical University of Denmark
Copenhagen, Denmark
s103072@student.dtu.dk

Christopher Frederick Wiin Schenk
Technical University of Denmark
Copenhagen, Denmark
s130172@student.dtu.dk

1. INTRODUCTION

Social bots that emulate human behavior have become increasingly common on social networks such as Twitter. While this indisputably introduces a number of concerns for the people that manage and maintain social networks, it also provides a unique playground for students and researchers that wish to investigate the fundamentals of human-bot interaction.

In this study we try to detect general features present in human Twitter accounts that are more likely to connect and interact with bots than others. The data for this study can only be collected from social interactions, so we invent a female user with a particular interest in veganism and fitness. Her profile information implies that she provides fitness motivation and inspiration, that she likes "organic/raw/vegan" foods and that she is a fan of the San Francisco Giants. Accordingly she posts motivating content in the form of images from Tumblr, retweets from popular fitness and vegan Twitter accounts and upvoted vegan cooking posts from Reddit. She is scheduled to favorite about 20 tweets daily in her timeline that are not retweets or replies. Incoming private messages are responded to using a chatbot while public messages are merely favorited, such as to project an image of engagement without the risk of exposure inherently related with chatbot answers.

2. IMPLEMENTATION

In practice the bot does two things. Firstly, it acts human in order to attract human users as followers. Secondly, it sorts interacting users into groups of users following back ('*followbacks*') and users not following back ('*ignorers*'). All of this is implemented using the Twitter API wrapper for Python and is executed on a free Amazon EC2 server scheduled using the software utility *Cron*. The routines for these two activities are described in this section.

2.1 Acting human

Table 1 gives an overview of the six *cron jobs* that comprise the bots "human-like" activity. Each job has an added amount of random delay corresponding to the time until next job or end of day. Unless otherwise stated, it is hereinafter understood that all reference to code execution implies the addition of random time delay.

The Tumblr script posts an image from a 1000+ long list of pre-mined generic fitness motivation images from Tumblr, with 3 to 5 popular fitness hashtags. Posting directly from Tumblr using the Tumblr API is also an option, which was however ruled out due to difficulties controlling reposts. The script that posts from Reddit retrieves content from the subreddit *r/vegan* in JSON format and finds a post title that links to an imgur image, which is also short enough to fit the tweet format. It then posts the title as the tweet, along with the imgur link. The bot attempts to retweet from popular fitness or vegan Twitter accounts, retrieved from a manually defined list, every hour but is only successful in 1/7 of its attempts due to added randomness. As such it will appear that the bot has days where it is highly active, and days where it is not. The chatbot mechanism is implemented with the Python package *selenium*, by piping incoming messages through the headless webbrowser *PhantomJS* to *Chomsky Chatbot* powered by *Pandorabots*, and reversely piping responses back. The script works both as a regular responding mechanism, but also as a filter to detect and remove other bots from the bots friendsphere. Every 10th minute it reads in the latest message and checks if the message ID is stored in the list of previously answered messages. If not it checks whether the message is likely spam from a fellow bot (contains any of the strings: http, unfollow, validate, click, facebook, google, pintrest, blog, FB, @, .com, .ly, website, dotcom, outu, .me, email, address, send, #, RT), in which case it deletes the message and unfollows the user. The message is thus only answered if it slips through this filter. A selection of typical chat message exchanges are shown in Appendix B

Every scripts that posts to the bots timeline uses geotagging. The specific location for each tag is [37.761915, -122.444467] with a random portion of noise in the range [-0.05, 0.05], generated independently for each geographical axis.

Table 1: Bot script jobs scheduled in PST.

min.	hour	day	Script
0	8	*	Post from Tumblr
0	8 - 23	*	Retweet popular
0	8	*/2	Post from Reddit
0	8 - 23	*	Favorite friends' posts
*/10	8 - 23	*	Respond with chatbot
0	8 - 23	*	Favorite mentions

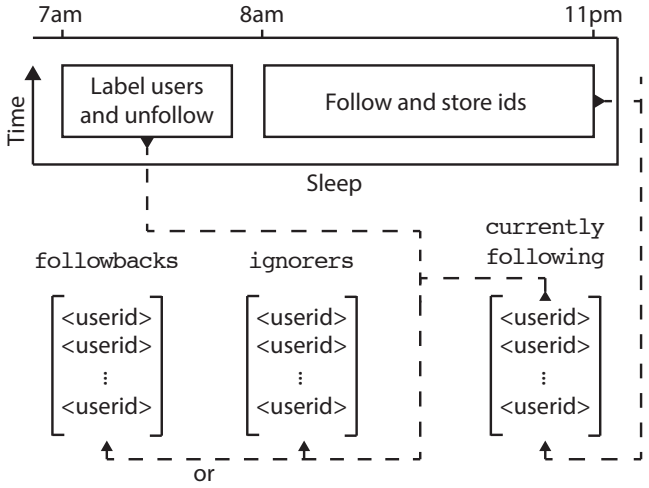

Figure 1: Routine for following and labeling users.

2.2 Collecting data

Fig. 1 illustrates how the routine for labeling interacting users as *followbacks* or *ignorers* is embedded in the daily follow/unfollow routine.

Briefly explained the bot follows on average 25 users an hour when it is active and adds them all to the bottom of a list that contains all users the bot is *currently* following. Every morning before the bot is activated, the top quarter of this list is removed from friends, labeled either as followbacks or ignorers and added to the one of the two lists they belong to.

On top of this, each of the lists containing followbacks and ignorers is continuously mined for features. This is implemented with a function that, given a user id, extracts the features presented in Tab. 2. Most of these features are generic and can be calculated just from the users profile data, while others, such as average number of hashtags in tweets and average tweet sentiment, are calculated from a subset of the 50 latest tweets produced by the user (replies and retweets included). Furthermore the features that quantify non-reciprocal friends sentiment, is calculated using the 50 latest tweets for each of 50 non-reciprocal friends selected at random.

3. SUSCEPTIBILITY ANALYSIS

This analysis is based on a dataset of 1238 observations, equally split into followbacks and ignorers. While this is not enough to make highly accurate statements about what defines susceptible users, it does provide some insight into what can be expected to actually matter in this respect. Each datapoint is represented as a 13 dimensional vector containing the features listed in Tab. 2.

3.1 Theory

The central course related theory used in this analysis involves machine learning and natural language processing (NLP). The latter theoretical subject is used in the process of feature extraction, where the tools from lecture 8 are employed to produce quantitative measures of users' and users' non-reciprocal friends' average tweet sentiment, i.e. happiness scores, a measure of Subjective Well Being (SWB)[1]. It is noted that the happiness score for reciprocal friends is disregarded due to the theoretical expectation of user assortativity based on this measure. Admittedly however, there could be some information to gain from including the difference between reciprocal friend happiness and user happiness, although the addition of this feature would increase the amount of API calls made for each feature extraction, hence greatly reducing the amount of retrievable datapoints.

Table 2: Features extracted for each selected user.

Feature name	Feature ID
#friends to #followers ratio	1
User age [days]	2
Average #hashtags in tweets	3
Average #links in tweets	4
#Reciprocal friends to #friends ratio	5
Average tweet sentiment	6
Non-reciprocal friends tweet sentiment	7
Average #tweets per day	8
Average #favorites per day	9
#Replies to total #tweets ratio	10
#RTs to total #tweets ratio	11
#Original tweets to total #tweets ratio	12
#Friends following bot to #friends ratio	13

Other NLP tools could also have been used, e.g. to compare stopwords-free lexical distributions between user and bot and use that as a feature. This would, however, be more relevant for a study that sought to understand the importance of content similarity in the context of user connection, rather than in a study, such as this, which seeks to understand, in more general terms, what defines the average bot susceptible user.

The analysis is in most part executed using machine learning and some basic statistics. While machine learning has been a returning subjects in this course, the analysis reaches slightly beyond the tools that have been introduced. To briefly comment on this choice, the introduced *NLTK* package does offers sufficient functionality to solve the task at hand, however without the same degree of flexibility that comes with a specialized machine learning package such as the one chosen for this study: *sklearn*.

Graph tools such as *NetworkX* has not been used in this work, mainly due to scaling problems inherent to the API request rate limit enforced by Twitter. With workaround methods future work could include network characteristic features for each user, such as assortativity, clustering coefficient, etc.

3.2 Outlier detection

In an effort the address the most probable flaw in this dataset, that is the likely presence of bots, a simple K Nearest Neighbor (KNN) density estimator is used to detect outlying datapoints, i.e. datapoints with very low KNN density. Fig. 2 shows KNN densities in the dataset for K = 5, where ob-

[1]When social bots attack: Modeling susceptibility of users in online social networks, Wagner et. al., 2012.

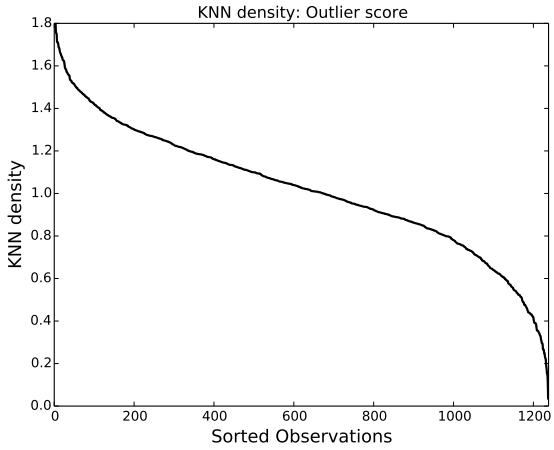

Figure 2: KNN densities in descending order, indicating the presence of outliers below the 10th percentile.

servations are sorted in order of descending density. The figure clearly indicates the presence of outliers, arguably below the 10th percentile. By comparing IDs of the outlying observations with IDs of observations in either class of the dataset, it can be shown that outliers are 60 - 80% more present among followbacks. Assuming that outliers can be regarded as bot-type users, this observation agrees with the notion that bots are typically automated to follow back and that they should therefore be over represented in this class.

3.3 Principle component Analysis

In order to detect for clustering in the dataset, a principle component analysis (PCA) is performed. Fig. 3 shows the how the datapoints project onto the first two principle components. The *left* figure displays the full dataset, and the *right* displays it without outliers (< 10th percentile KNN density removed). The plots make no clear indications about clustering in the data, thus motivating the use of more advanced methods.

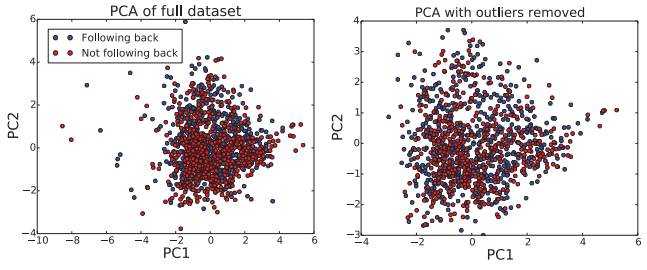

Figure 3: The full dataset and the dataset without outliers plotted on the first two principle components.

3.4 Choice of classifier

In order to determine what classifier may be used to best predict observations in the data, a number of popular classifiers, implemented with the Python module `sklearn`, are tested against the dataset using 10-fold cross-validation. Tab. 3 lists the accuracies of different classifiers in descending order. As the list indicates, the highest accuracy is obtained using a Random Forest (RF) classifier, which is therefore chosen for use in the further analysis.

Table 3: Different classifiers and their accuracies when tested against the dataset.

Classifier	Accuracy
Random Forest	0.732
AdaBoost	0.708
Decision Tree	0.671
LDA	0.623
Linear SVM	0.607
RBF SVM	0.579
Nearest Neighbors	0.570
Naive Bayes	0.562
QDA	0.564

3.5 Feature analysis

Calling the method `.feature_importances_` on the random forest classifier object with `sklearn` returns a vector indicating the importances of the different features. These are listed in order of importance in Tab. 4.

Table 4: Feature importance scores and observation means for each class, followbacks (FB) and ignorers (IG), in order of importance. See Tab. 2 for feature ID reference.

Feature ID	Importance score	$\bar{x}_{\mathrm{FB}}$	$\bar{x}_{\mathrm{IG}}$
13	0.158	0.0226	0.0009
5	0.112	0.1222	0.1624
2	0.090	1039.9	1275.3
7	0.074	6.2768	6.1852
1	0.074	1.9880	1.9038
9	0.071	5.3073	3.2925
6	0.068	6.2833	6.2186
8	0.064	11.942	9.8949
3	0.063	0.5223	0.5094
11	0.061	0.3293	0.3234
12	0.056	0.4891	0.5018
4	0.056	0.2778	0.2742
10	0.053	0.1850	0.1776

The importance score only provides information about *how much* features matter, not *how* they matter. To give an understanding of this the mean values of each feature in each class is listed in Tab. 4 aswell. E.g. it is observed that there is a relatively large difference between the means for feature 13, making it an important feature, and that this mean is greater for followbacks indicating, not surprisingly, that this group of users have more friends that follow the bot (see Tab. 2 for feature reference).

To visualize *predicted feature distributions*, each column of features in the dataset can be plotted against the probabilities of observations being classified as followbacks, estimated using the function `predict_proba`. In Fig. 4 this is done for the top three features: 13, 5 and 2, respectively. Coloring, corresponding to estimated Gaussian Kernel Den-

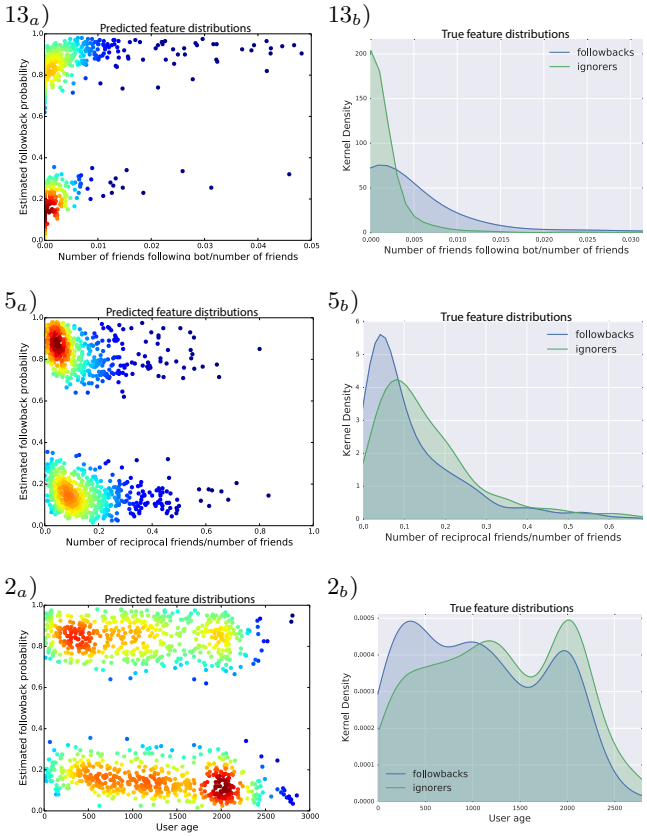

Figure 4: Predicted and true feature distributions for the three most informative features in the dataset. The predicted distributions are colored in accordance to a KDE where red represents high density and blue, the opposite.

sity (KDE) - where red represents high values of KDE and blue, the opposite - reveals how features in each class cluster around different values. For the sake of rigor, distributions of the features in their *true* classes, referred to as *true feature distributions*, are displayed as well. Note, however, that "predicted" feature distributions are also "true" in the sense that they are predicted by a model trained on themselves. Hence we can only interpret a datapoint's position on the "Estimated followback probability"-axis as a measure of how typical it is to its class.

While the differences in how the observations cluster are subtle, Fig. 4 definitely points to the fact that there are measurable differences between followbacks and ignorers. From the predicted feature distributions in Fig. 4 we observe that the model tends to estimate higher followback probabilties in both classes as the feature value increases. In 4.5 it can be seen that followbacks tend to have fewer reciprocated friendships, while 4.2 shows that followbacks are statistically newer to Twitter. Similar plots for the remaining features can be found in Appendix A. The most notable observations from the appendix are in features 7 and 12, corresponding to "Average non-reciprocal friends tweets happiness" and "Number of original tweets/total number of tweets", respectively. In feature 7 it is seen how the distribution is shifted slightly towards higher values of non-reciprocal friend happiness for followbacks. In feature 12 it is observed that the amount of original tweets produced by a user is shifted towards a lower value for followbacks.

The remaining features exhibit no pronounced differences in their distribution across classes and will not be commented on in further detail, though a reader with an particular interest in subtlety may find value in them.

4. INTERVENTIONS

A series of "interventions" were carried out throughout the final weeks of the course, involving hashtag specific tweeting from all working bots associated with the course. The aim of this was to measure the impact that it could have on a local Twitter environment. The interventions were implemented by having each bot produce original tweets with a predefined original hashtag, and continuously amplify this hashtag by retweeting and favoriting tweets that used it. To allow for continuous access to tweets the Twitter streaming API was used. The streaming API requires a continuous HTTP connection to return tweet data in realtime, allowing for instant reacting to events. The stream can be filtered to only return tweets containing a given hashtag or keyword.

4.1 Favoring and retweeting

Favoring and retweeting was implemented with a script that kept a running connection to the streaming API and instantly favorited and retweeted hashtag matching content. After each retweet the script was put to sleep for, on average, 15 minutes (not closing the connection), a simple mechanism to avoid spamming. The first four matches were retweeted independent of origin, and the next 15 were then restricted to not originate from course bots. After a total of 19 retweets, the script stopped retweeting and kept on only favoriting.

4.2 Timing strategy

Tweet scheduling was based on an initial/follow-up scheme. The idea was to tweet at the beginning of the intervention interval to help get the ball rolling. Then, after some hours had passed and the hashtag had likely built momentum, a second tweet was produced. The aim was to post the second status between 2 p.m. to 8 p.m., which is when users are shown to retweet the most[2].

4.3 Content strategy

The overall content strategy was to include content shown to be retweetable, such as statuses containing references to popular media and celebrities, and content of positive nature. E.g. for the #turkeyface intervention this strategy was employed by photoshopping a cropped image of the popular movie phenomenon George Clooney's face to a turkey holding an espresso. For the #GoldenGateAliens intervention the approach was to speculate about whether the presence of aliens was due to the making of a movie.

[2]The Science of ReTweets, Dan Zarells, 2009.

4.4 Evaluation

The discussed content-strategy was not implemented until half way through the interventions. This created an opportunity to compare tweet performance before and after strategy implementation. Without using sophisticated statistical tools and simply looking at the amount of attraction that the tweets have received, it is observed that strategic content has in fact been retweeted more. Regarding the timing strategy no effect of optimal interval was registered, likely due to the tiny size of the dataset.

5. DISCUSSION AND CONCLUSION

The susceptibility analysis points to three distinct features that are more present in users susceptible to bots: they have a relatively high amount of friends following the bot, they don't have so many friendships that are reciprocated and they are rather new to Twitter. These findings are very much in agreement with any prejudice one could have about people that don't think twice and just click the follow button. Looking at the appended figures, however, there are subtle hints of more unapparent features, such as a tendency to follow happier people, and a lower degree of original tweet output.

For a study such as this, however, the results should be taken with a grain of salt. It can hardly be disputed that there will always be a high degree of context dependency, in this case the most pronounced issue likely being that the bot was created as an attractive woman that attempted to only interact within specific communities on Twitter. As such the results could just as well reflect a much too specific type of user, e.g. lonely men in the bay area.

In future studies this notion should be taken into account when designing an avatar. For the scientific purpose of profiling the *average* bot-susceptible Twitter user, the optimal Twitter bot should be someone very *average*, that is by any measure equally appealing to all people. This natural principle of designing the "bait" to attract the desired "prey" also applies for any other group that a study may want to profile.

APPENDIX

A. FEATURE DISTRIBUTIONS

In this appendix the remaining feature distributions are presented.

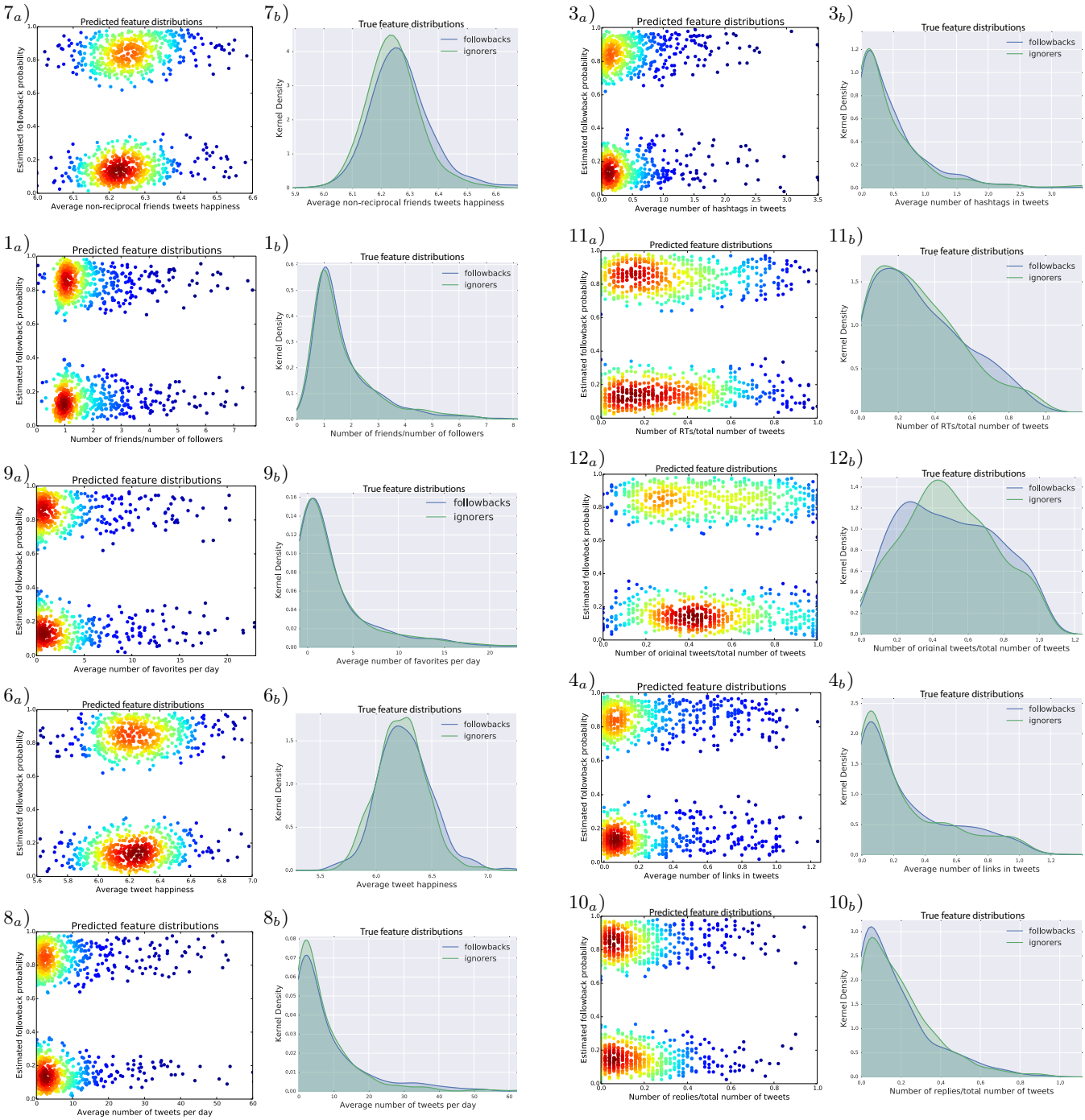

Figure 5: Predicted and true feature distributions for the remaining features in the dataset. The predicted distributions are colored in accordance to a KDE where red represents high density and blue, the opposite. Plots are listed in order of feature importance.

B. CHATBOT EXCHANGES

Example of typical chatbot exchanges.

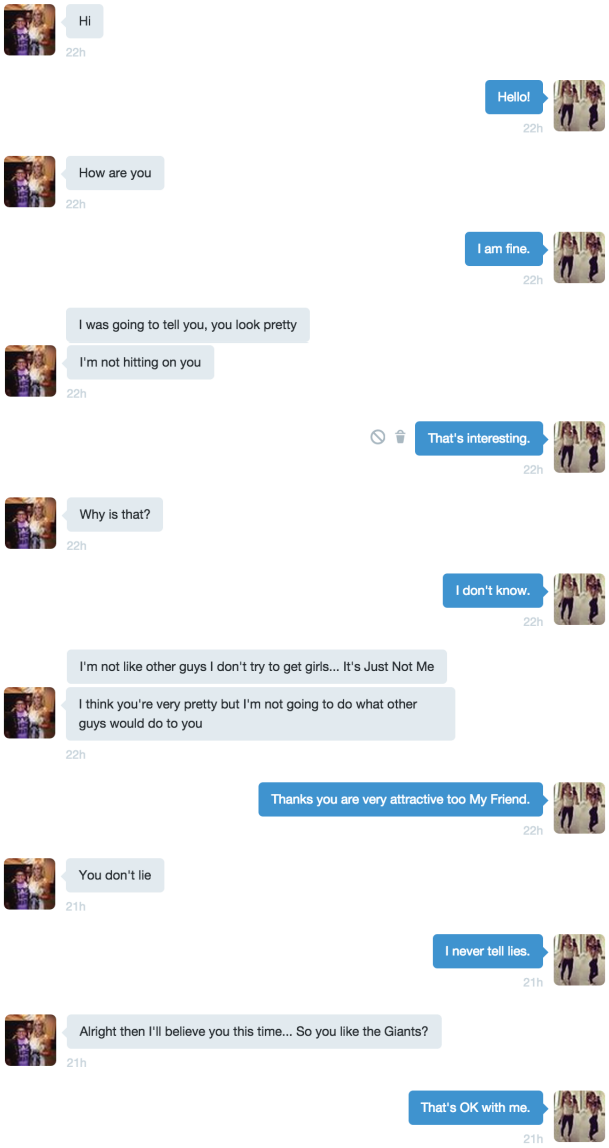

Figure 6: In order to keep conversions brief to minimize the risk of being exposed, the bot only responds with short sentences.

Figure 7: This example shows how most exchanges go, and also illustrates the main purpose of the bot, namely to say "Thank you too" or "No problem" to the countless messages like this that are received. Ideally all conversations whould look somewhat like this.

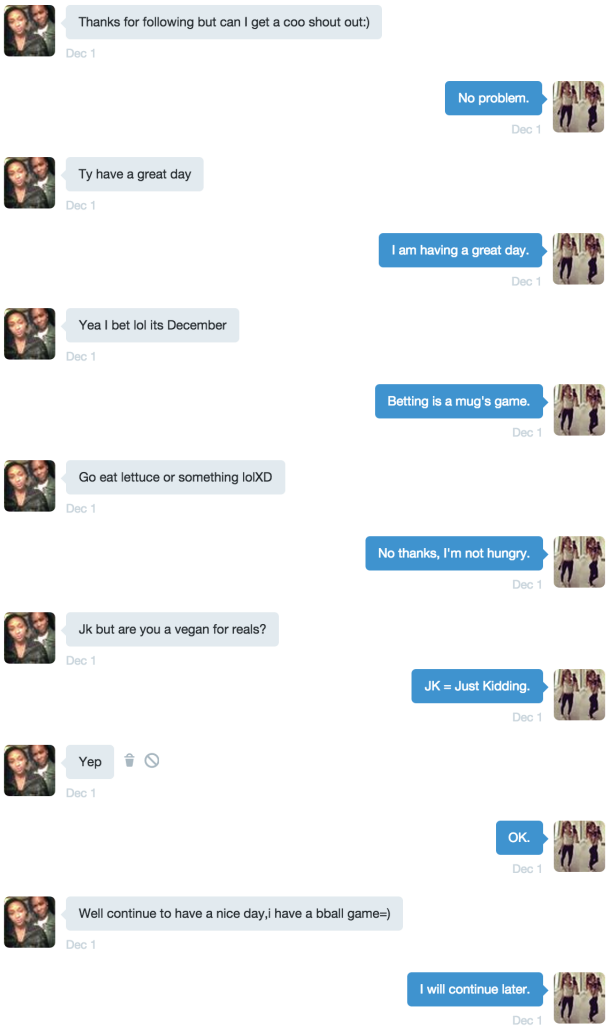

Figure 8: A fine example of how the bot stops making sense at some point where the user usually breaks off the conversation.

C. THE TRUE REWARD OF ALL THIS WORK