Breaking Down Social Media Filter Bubbles via Reinforcement Learning

Tal Stramer
Department of Computer Science
Stanford University
Stanford, CA 94305
tstramer@stanford.edu

Abstract

This paper tackles the problem of social media filter bubbles, which is a phenomenon in which a user's social media feed become tailored (either via an algorithm or manually or both) over time to agree with their preconceived notions. There has been very little work to address this problem by social media companies themselves, so we build a Twitter bot to target and engage users of specific political viewpoints with content that they may disagree with. Using Q-learning, the bot learns the right mix of complimentary, mime, conversational, and political content to engage the user with to maximize engagement. The model is able to produce a positive response from over 25% of users engaged.

1 Introduction

Since the advent of social media and the Internet generally, a host of new challenges around fact-checking have emerged. As people consume more and more of their news via platforms like Facebook and Twitter, the number of news sources has proliferated, and there is less and less centralized agreement on facts anymore. As a result of this, it has become much easier for malicious actors to spread outright falsehoods. One issue that is contributing to this problem is social media filter bubbles, which result in user's being exposed to only content that reinforces their existing beliefs, rather than forcing them to fact-check or question content.

While there is a good amount of literature that analyzes the effect of social media filter bubbles, especially on news proliferation, there has been very little work to come up with approaches that can actually address the problem in the immediate future. Our approach provides a novel, practical way to begin to break down social media filter bubbles.

In this paper, we focus on building a Twitter bot to engage user's having a polarized political perspective with content they might disagree with. One of the biggest challenges of building this type of system is overcoming a user's hesitancy or resistance to content, especially political, that disagrees with their viewpoint. This is even more of a problem when the content comes from an apparent stranger. To address this problem, we use reinforcement learning to learn strategies for how to best engage user's with politically disagreeable content that maximizes positive engagement. The bot is equipped with various strategies to attempt to elicit positive engagement before presenting content that may disagree with the user, such as complimenting the user or linking them to a funny picture or mime.

Our main contributions are as follows:

- Finding that user's on social media are willing to listen to opposing political viewpoints if presented in a specific context.

Fall Quarter CS-238 Final Project 2016 Stanford, CA

- Building an algorithm that can automatically identify user's having a polarized political perspective.
- Using reinforcement learning to learn how to present politically disagreeable content to user's on social media while maximizing positive engagement.

2 Related Work

There have been several previous papers that study the effect of blogs and social media platforms on the consumption and proliferation of news. Bakshy et al. [1] looked at how much cross-cutting content individuals encounter on Facebook and found that 24% of the hard content shared by liberal's friends are cross-cutting, compared to 35% for conservatives. After ranking, the amount of cross-cutting news actually viewed is between 5-10% less than shared.

Pentina et al. [2] looked at the evolution of news consumption in the digital age, and found that information overload is a major contributor to filter bubbles. They note that while people do not necessarily trust what they read on Facebook or Twitter, it is still used as a filtering mechanism to determine what news may be important to look into more based on social signals.

There has also been previous work to explore ways to build social media bots to automatically interact with user's towards some goal. Levine et al. [3] used active learning to build a Twitter bot to attract followers. They model the problem as a contextual bandit problem, which is a generalization of the multi-arm bandit problem. They restrict the set of actions to retweets, extracting a set of features from each tweet to be retweeted and observing changes in follower count after retweeting. They use the ϵ-greedy algorithm to explore new features. A major drawback of their approach is that the bot is very limited in its possible actions, for example it cannot engage users in conversation.

Teljsted [4] uses supervised learning to detect automated Twitter accounts related to the Ukraine conflict. They look at a variety of different features, and found that features such as the number of mentions per tweet, the urls per tweet ratio, and tweet length variance have the highest information gain for distinguishing a bot from a real user. Our bot takes into account these characteristics when deciding which action to take, so as not to appear to be a bot to users.

3 Methodology

At a high level, we build a Twitter bot that engages users based on an optimal policy computed via Q-learning. In the sections below, we describe each part of the system.

3.1 Targeting

In this paper, we focus only on English tweets sent from users in the United States. Furthermore, we restrict our analysis to users with at least 500 followers and 500 tweets, who are more likely to be active users of Twitter. To detect users with a polarized political perspective, we hand-craft a list of keywords that are commonly used by people in the political left or right. For example, for the political right we use keywords such as "#MakeAmericaGreatAgain" and "#TrumpTrain" while on the left we use keywords such as "#AuditTheVote". We then only consider tweets containing one of these predefined keywords. We further filter users by the sentiment of their tweets, considering only users that have recent tweets that are moderately to very negative, as determined by a sentiment analysis library. These users are more likely to have a politically polarizing viewpoints.

3.2 Actions

Our bot can take one of the following actions: Reply to a tweet (A_1), favorite a tweet (A_2), retweet a tweet (A_3), or follow a user (A_4). Additionally, the action of replying to a tweet breaks down into the following sub-actions:

- Reply with one of a list of pre-defined compliments, such as 'You're wonderful' ($A_{1,1}$)
- Reply with one of a list of pre-defined perfunctory questions, such as 'How are you doing?' ($A_{1,2}$)

- Reply with one of a list of pointed questions, such as 'Proof?' ($A_{1,3}$)
- Reply by quoting a mime-style tweet, such as inspirational quotes or cute pictures. These lists were generated by first hand-crafting a list of seed accounts (such as @cuteemergency for cute pictures) and then automatically extracting the highest engaging tweets from these accounts. ($A_{1,4}$)
- Reply by sharing an article from a news organization considered to be 'cross-cutting' from the political perspective of the user. These tweets were generated by first hand-crafting a list of seed accounts (such as @theyoungturks for the political left and @foxnews for the political right) and then automatically extracting recent high engaging tweets from these accounts.($A_{1,5}$)

3.3 State Space

Each state in our state space consists of a set of features extracted from the conversation between a bot and user. These features include:

- User's follower count, discretized into three categories: low (< 1000), medium (< 5000), or high (>= 5000) (S_1)
- User's tweet count, discretized into three categories: low (< 1000), medium (< 5000), or high (>= 5000) (S_2)
- Sentiment of user's last tweet in conversation, discretized into three categories: negative, neutral, or positive (S_3)
- Number of tweets in the conversation, discretized into three categories: 0 user replies, 1 user reply, or multiple user replies (S_4)
- Whether or not a bot tweet in the conversation has been favorited or retweeted by the user in the conversation (S_5)
- Whether or not a user tweet in the conversation has been favorited or retweeted by the bot in the conversation (S_6)
- Whether or not the bot is following the user (S_7)
- Whether or not the user is following the bot (S_8)

While there is other potentially useful information we could include in the state representation, we avoid doing so due to the already high dimensionality of the state space (1296) coupled with the limited data we are able to obtain due to Twitter's aggressive API rate limits. In the 'Learning' sub-section below, we describe how we address the high-dimensionality problem using local approximation.

3.4 Rewards

The reward for a (state, action) is additively de-composed into the reward for the state and action, i.e. R(s, a) = R(s) + R(a). R(s) is a weighted sum of different state variables. The table below lists the variables, values, and weights for each state variable (if a variable does not appear it has a weight of 0):

variable	values	weight
S_3	negative=-1, neutral=0, positive=1	5
S_4	no_replies=0, one_reply=1, multiple_replies=2	5
S_5	false=0, true=1	5
S_8	false=0, true=1	10

This definition of R(S) rewards states in which the user has engaged the bot positively, either via positive sentiment replies, retweets, favorites, or follows. R(A) is defined by a score for each action type. The table below lists these scores (if an action type does not appear it has a score of 0):

action	score
$A_{1,5}$	5
A_3	-1
A_4	-2

This definition of R(A) rewards sharing cross-cutting political content and punishes the bot for retweeting or following a user. The reason for the negative rewards for retweeting and following a user is to attempt to avoid the bot giving away the fact that it is a bot via a timeline full of only retweets or a high ratio of (accounts the bot is following) to (accounts following the bot).

3.5 Learning

To learn the optimal action for each state, we use Q-learning with local approximation. To explore the state space, we use the ϵ-greedy algorithm, which executes a random action some fixed percentage of the time. In our case, we start with ϵ=.5 and reduce the value over time as we accumulate more data.

The optimal Q function is re-learned based on all available data on a fixed interval. Based on manual tuning, we use a discount factor of .9 and a learning rate of .1. In order to generate a list of (state, action, next state, reward) pairs for training, we iterate over each finished conversation between a bot and user, and break it up into a set of (state, action, next state) pairs based on all the bot actions taken in the conversation. A conversation is considered finished when there has been no activity on the conversation for at least an hour. The reward for each state and action is calculated based on the reward function described in the previous section.

Due to the high-dimensionality of our state space, many of the states will not have any examples to learn from. However, we assume that states close to each other will have similar values, and use bilinear interpolation to assign values to unseen states after learning. The neighbor function is based on representing each state as a vector and computing the closest neighbors based on the 2-norm distance.

3.6 Anti-Bot Behavior

To avoid appearing to be a bot, we employ various tactics to appear to be a normal user. To that end, our bot will periodically fetch its home timeline via the Twitter API and retweet a popular tweet within it. In addition, it will periodically execute one of actions $A_{1,4}$ or $A_{1,5}$ randomly in non-reply form.

3.7 Architecture

Our system is built entirely in Python and is broken up into several different processes that can all run independently, providing the ability to easily scale the number of bots. These processes include:

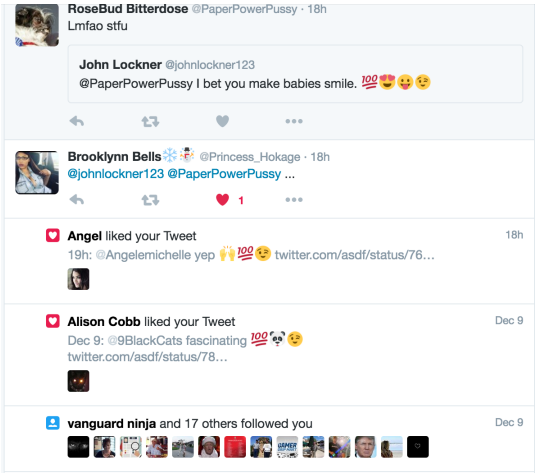
- Targetor: This process streams tweets from the Twitter API's GetStream endpoint, which allows us to stream tweets matching a set of filter predicates (in our case a list of left or right-wing keywords). We use our targeting algorithm to determine which streamed tweets to engage with. For each of these, a new 'State' object is pushed onto an action queue for further processing. If the total number of active states is above a specific threshold (20 in our case), we stop processing new tweets until we are below the threshold (which will happen when conversations expire due to inactivity of at least an hour). This avoids exceeding Twitter API rate limits.
- Action executor: This process listens for 'State' objects pushed onto the action queue. For each state, the action executor determines the optimal action to execute based on an optimal Q table (computed by the 'Learner' process), and then executes this action. A random delay between 1-3 minutes is added before executing an action so as not to appear to be a bot to the user.
- Learner: This process runs periodically (every 5 minutes in our case), and retrieves all available data and updates a centralized, in-memory definition of Q which is then read by the action executor.
- Anti-bot behavior executor: This process runs periodically (every 5 minutes in our case), and executes our anti-bot behavior algorithm.
- User event recorder: This process streams user events from the Twitter API's GetUser endpoint, which returns event specific to the bot user, such as follows, favorites, retweets, replies, etc. For each event, the appropriate State object is updated and pushed back onto the action queue.

4 Experiment & Results

To test our algorithm, we ran 10 different bots for 48 hours, maintaining a central definition of our optimal Q policy among all bots. Five of the bots targeted left-wing tweets and five targeted right-wing tweets.

In total, our bots sent ≈1300 replies, followed ≈250 users, were followed by ≈100 users, retweeted ≈400 tweets, and favorited ≈800 tweets. The percentage of users that engaged in some way with the bot (retweet, favorite, reply, or follow) was ≈43%, while the percentage that engaged positively (i.e. a non-negative sentiment reply) was ≈26%. One bot account was flagged for automated behavior by Twitter, which prompted the implementation of the anti-bot behavior algorithm and reducing the number of active conversations for each bot, after which no account was flagged.

Below is an example of an activity timeline of one of the bots:



The table below summarizes the 6 highest value (state, action) pairs discovered after learning in ranked order.

State	Action
(low,medium,negative,0,false,true,true,false)	$A_{1,4}$
(low,low,negative,0,false,true,true,false)	$A_{1,4}$
(low,medium,negative,0,false,true,true,false)	$A_{1,1}$
(low,low,negative,0,false,true,false,false)	A_2
(low,low,negative,0,false,true,false,false)	A_4
(low,low,negative,1,false,true,true,false)	$A_{1,5}$

Notably, the highest-reward actions all occur for users with low follower count (<1000) and low or medium statuses count (<= 5000). These users appear more likely to engage with an apparent stranger. In addition, the top 3 most valuable actions are to either send a user some sort of mime content or compliment them, and occur when we are in a state in which we have not yet tweeted at the user but have favorited their tweet and followed them. After that, the next best actions are to follow a user or favorite their tweet before sending the first reply to the user. Notably, retweeting is not as high-value, which is likely due to the reward function definition. Another thing to note is that the action of tweeting a cross-cutting article is only high-value after sending an initial reply, which intuitively makes sense as the user is more likely to feel comfortable engaging.

Among user's who in engage in some way, ≈75% of these engagements are a single favorite with no future engagements. Cross-cutting political content shared as the first reply is met with no reply ≈80% of the time, a negative sentiment reply ≈15% of the time, and a retweet or favorite ≈0% of the time. Among conversations with at least one reply by the user, cross-cutting political content is met with a reply around ≈70% of the time, a negative sentiment reply around ≈60% of the time, and a retweet or favorite ≈4% of the time. The conversation usually ends after a 3rd reply from the bot due to incoherence.

5 Discussion & Conclusion

In this paper we examined the problem of social media filter bubbles. We attempt to break down these bubbles by building a Twitter bot to engage users of specific political viewpoints with content that they may disagree with while also attempting to maximize positive engagement.

Our results show an improvement in positive engagement after sharing cross-cutting political content if it is preceded by a tweet complimenting the user or sharing mime content.

One drawback with our system is the lack of coherence of bot replies which is primarily due to the bot not taking into account the text of the tweet being replied to in the model. This is especially a problem after the first reply by a user, in which there are higher expectations of coherence. A future improvement of the system to address this problem would be to build a topic model for tweet text, and then categorize actions according to this topic model and incorporate the topic into the state space.

Another future improvement would be to train an initial model offline based on a training set of conversations and engagements between real users. This would help the bot with overall coherence and could eliminate the need for hand-crafting actions.

References

[1] Bakshy, Eytan, Solomon Messing, and Lada A. Adamic. "Exposure to ideologically diverse news and opinion on Facebook." Science 348.6239 (2015): 1130-1132.

[2] Pentina, Iryna, and Monideepa Tarafdar. "From “information” to “knowing”: Exploring the role of social media in contemporary news consumption." Computers in Human Behavior 35 (2014): 211-223.

[3] Levine, Nir, Timothy A. Mann, and Shie Mannor. "Actively Learning to Attract Followers on Twitter." arXiv preprint arXiv:1504.04114 (2015).

[4] TELJSTEDT, CHRISTOPHER. "Separating Tweets from Croaks."