Assignment 4

Due **Sun Nov 10** by 11:59PM to Canvas

For this assignment, you’ll create a “chatbot” (really a dialog system) to detect satire, and respond with a sassy response of your own. You’ll build your bot on Twitter -- people will write to your bot by starting their messages with @<yourbot> and your bot will figure out what it should say back.  
  
Learning objectives:

* Describe why a language model is a useful tool for many applications in NLP
* Create a classifier for Natural Language text and understand its limitations (or error modes)
* Determine socially appropriate ways for bots to respond to humans
* Recommend whether models can be used in applications for which they were not originally created

This assignment has the following main parts:

[Detecting Satire (40 points)](#_sejczo5s0qat)

[Set Up Twitter Bot (20 points)](#_30j0zll)

[Integrate the bot with the satire classifier (30 points)](#_1fob9te)

[Reflect: Would you recommend using our satire-classifier as a good starting point to build a fake-news classifier? (10 points)](#_kh03wm2f459c)

[Extra credit: Test with Users and Iterate (5 points)](#_tyjcwt)

[Extra credit: Deploy bot (10 points)](#_begtfdv3khxn)

[Grading Overview and Rubric](#_3dy6vkm)

Before you begin, download the Jupyter notebook for this assignment, and associated data: [Notebook+Data](https://drive.google.com/open?id=1Z9d20QI_eJydq5wT4pJ_uTepzD_rOIa4) [186MB]

## Detecting Satire (40 points)

Follow the Jupyter Notebook we created for this part. If you get stuck, ask for help on Piazza. This is supposed to take you about 2 fun-filled hours (not counting the time your computer’s training the model!) If you are taking much much longer, ask for help on Piazza!

For the stuff below, we hope you ran through the Jupyter notebook we created. If it didn’t work for you, you can still go on below. Remember to use the exported model file, like so:

load\_learner(path=data\_path, file='satire\_awd.pkl') and

load\_learner(path=data\_path, file='headlines-lm.pkl')

## Set Up Twitter Bot (20 points)

*Do this this week! Sometimes Twitter takes a long time to approve you!*

**Create a Jupyter notebook (or add to the satire one.) Below we call this the “bot notebook”.** As you complete the instructions below, you’ll notice there are some questions (in red/bold). Please answer these in the bot notebook.

Your chatbot will work as follows -- you’ll run a python script, which will check to see if there are any messages addressed to your bot that have not yet been answered, and, if so, it will figure out how to respond. This script will be run periodically to check for new messages. For the purposes of this assignment, you can just run your script manually.

You could just type to your chatbot from the commandline, but that would be pretty boring. So, we’re going to make a Twitter bot. This introduces some interesting challenges, and is something that real people actually write chatbots for! In fact, networks of Twitter bots can apparently throw elections, ruin whole world orders, etc. Be careful with your new powers.

The first step is to sign up on Twitter for API access. This has all gotten a bit more complicated as Twitter has over the years attempted to deal with various threats. Nevertheless, despite all the additional questions, Jeff did this in < 5 minutes.

Apply for a developer account here:  
<https://apps.twitter.com>

**As you go through the process of signing up for a developer account, reflect on the questions you are asked, why you are being asked them, and how you think they will serve the intended purpose (or not). (5 points)**

If you get stuck somewhere, or are otherwise unable to sign up for a developer account. Please on Piazza, so we both know and can get back to you asap. You can still do the rest of the assignment (don’t get stuck!), but you’ll of course need to modify it a bit so you’re not using your own account. For instance, have your bot just print its replies to the commandline instead of tweeting them back.

We’ll be using a Python library called “[Tweepy](http://www.tweepy.org/)” to interact with the Twitter API. You can install that in your python environment using pip:  
pip install tweepy (use !pip in jupyter)

To authenticate Tweepy to use your Twitter account, you’ll need to provide it with your Twitter developer credentials.

consumer\_key = 'XXX'

consumer\_secret = 'XXX'

access\_token = 'XXX'

access\_token\_secret = 'XXX'

You can obtain these from your Twitter developer page. Create a new application, go to the Keys section, and copy/paste and/or generate them from there.

This all assumes that you don’t mind signing up and doing all of this under your main Twitter account. You are more than welcome to sign up for another Twitter account, and following these instructions there. You can even sign up as a developer on one account, and then access another account’s feed (that’s why there are two sets of tokens). But, for these instructions, we’re assuming you’re doing everything on one account.

I recommend putting these in a separate python file, e.g., credentials.py, and then importing that file into your main script. That helps separate out your secret information, and will make it easier for you to, for instance, turn in your bot code without also giving us access to your Twitter account. (Note: We will create a file in the folder called credentials.py with our own credentials)

Once you do this, you can test out whether or not your script has access to Twitter with the following:

import credentials

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

api = tweepy.API(auth)

user = api.me()

print (user.name)

This snippet of code should access your Twitter account, fetch your name stored there, and then print it out. If it doesn’t, go back and debug!

The next step is to find a tweet that mentioned your bot, which are the messages that your bot will respond to. Here’s some template code for searching for tweets with Tweepy:

####

# Define the search

#####

query = '@satirebot'

max\_tweets = 100

####

# Do the search

#####

searched\_tweets = []

last\_id = -1

while len(searched\_tweets) < max\_tweets:

count = max\_tweets - len(searched\_tweets)

try:

new\_tweets = api.search(q=query, count=count, max\_id=str(last\_id - 1))

if not new\_tweets:

break

searched\_tweets.extend(new\_tweets)

last\_id = new\_tweets[-1].id

except tweepy.TweepError as e:

# depending on TweepError.code, one may want to retry or wait

# to keep things simple, we will give up on an error

break

####

# Iterate over the search

#####

for status in searched\_tweets:

# do something with all these tweets

print (status)

If you run this snippet, it will print out the messy JSON version of all the statuses returned. You can see what sorts of things each of the status objects contain. Some important ones are:

text -- the text of the returned tweet

author.screen\_name - the Twitter username of the user who sent the tweet

id - id of the tweet, which you can use to reply to it, or search only for tweets that were posted after it

And, of course, you want to be able to send new tweets to, which is done like this:  
api.update\_status('I\’m on a boat!')

You will likely want to not just send a status, but actually reply to the original tweet. You can combine some of what you’ve learned here to do that:  
api.update\_status(

'this is a reply! @' + status.author.screen\_name,

status.id\_str

)

You have to include the @-mention in the tweet, or it won’t show up as a reply.

**Add code to the bot notebook to respond to a query, as above. (15 points)**

Of course, this is just the tip of the iceberg with what you can do with Tweepy. If you’re interested, there is a lot of documentation online. You might start with the [official Tweepy docs](http://docs.tweepy.org/en/v3.6.0/getting_started.html).

## Integrate the bot with the satire classifier (30 points)

Now that you can do basic replies with your bot, it’s time to make it do something useful! Specifically, our bot should do two things:

1. When someone tweets a headline @ the bot, it replies with whether the headline is satire.
2. It also makes up a headline that plays off the original headline, and tweets it back.

Here’s an example (assuming the bot’s called @satirebot):

User: @satirebot Rising Seas Will Erase More Cities by 2050, New Research Shows

satirebot: Yeah, looks real, not satire. But here’s what I say: “Rising Seas will sound great on national security”

*Hint: you want to call api.update\_status using the outputs of our models (classifier and language models) instead of responding 'this is a reply!'*

Grading rubric for this part:

Bot responds with satire or not: 10 points

Bot responds with made up headline: 10 points

Bot interactions are designed to be friendly, non-misleading, and should let users to discover what is going on. 10 points

**Add code to the bot notebook to respond to a query, as above. (30 points)**

## Reflect: Would you recommend using our satire-classifier as a good starting point to build a fake-news classifier? (10 points)

If so, what changes would we need to make to make it useful for this purpose? If not, why not?

**Add reflections to the bot notebook (10 points)**

## Extra credit: Test with Users and Iterate (5 points)

In this part, you’ll ask three participants to interact with your bot. You’ll give the user high-level information about what the domain of the bot is, and then see how they interact with it. Ask each of the participants to ask your chatbot at least three different things. Record how they interact with your bot. After this participant input, update your bot to attempt to address how that participant interacted with your chatbot.

**Add to bot notebook: How did what your participants input compare to the ones you tested so far? How did participants react when the chatbot didn’t respond correctly, or responded with nonsense? (2.5 points)**

**Add to bot notebook: what change could you make in response to this feedback? (2.5 points)**

## Extra credit: Deploy bot (10 points)

Take your Jupyter notebook (the bot notebook) and deploy it such that it runs once an hour, and responds to all messages sent to it.

**Add to bot notebook: Twitter handle (of bot) to test for. (10 points)**

## Grading Overview and Rubric

Please turn your assignment in on Canvas -- Please turn in a .zip file of the two notebooks (and any other files you used.)

For the grading rubric, see each section.