Stylometric Analysis of Raw Tweets Using Scikit-Learn

William

GA-SEA-DAT2

What is Stylometry?

- **The Problem: Identifying authorship of text through analysis of *writer invariants.***
- **Two characteristics are necessary for something to qualify as a writer invariant:**
  - **It is statistically unique to the same author**
  - **Remains largely unchanged throughout text**
- **Features that may help identify invariants:**
  - **Numeric: *Average sentence length, syllables, etc***
  - **Synonyms: *Word choice, use of contractions, etc***
  - **Punctuation: *Comma, semicolon, or hyphen use***

What is Twitter?

- Twitter is a social network that is best known for it's 140-character limit on user posts.
- It does not have any real-name policy, and anyone can easily register more than one account.
- In theory, the short character limit on posts should make it harder for writer invariants to surface in new posts.

The Question(s)

- **In the absence of usernames, what *features* can we best identify tweet authorship with?**
- **Are there features characteristic to tweets that make them easier to attribute?**

- **Hypothesis: Despite the 140-character limit, tweets may have other useful invariants:**
  - ***Frequency of link usage, which domains***
  - ***If added text is placed before/after links***
  - ***Hashtag usage, response to other users***

Some Limitations

- **Prospective user request: “I don't trust anything I can't understand myself.”**
- **Therefore: Simplicity > Accuracy**
- **Feature Combinations: A model with too many features early on makes it hard to tell what the effect of each is.**
- **Realism: A typical stylometry scenario will generally be between suspect and unattributed text passages.**

Procedural Summary

1. **Data Acquisition:** Acquire tweets from two or more feeds using Twitter's API
2. **Pre-Processing:** Extract numeric data about possible features discussed earlier
3. **Model Development:** Create a logistic regression model that can attribute tweets
4. **Analysis/Iteration:** Attempt to find the *simplest* model that can achieve stated goal
5. **Result Visualization:** Produce visuals that can convey to possible users how this works

The Data

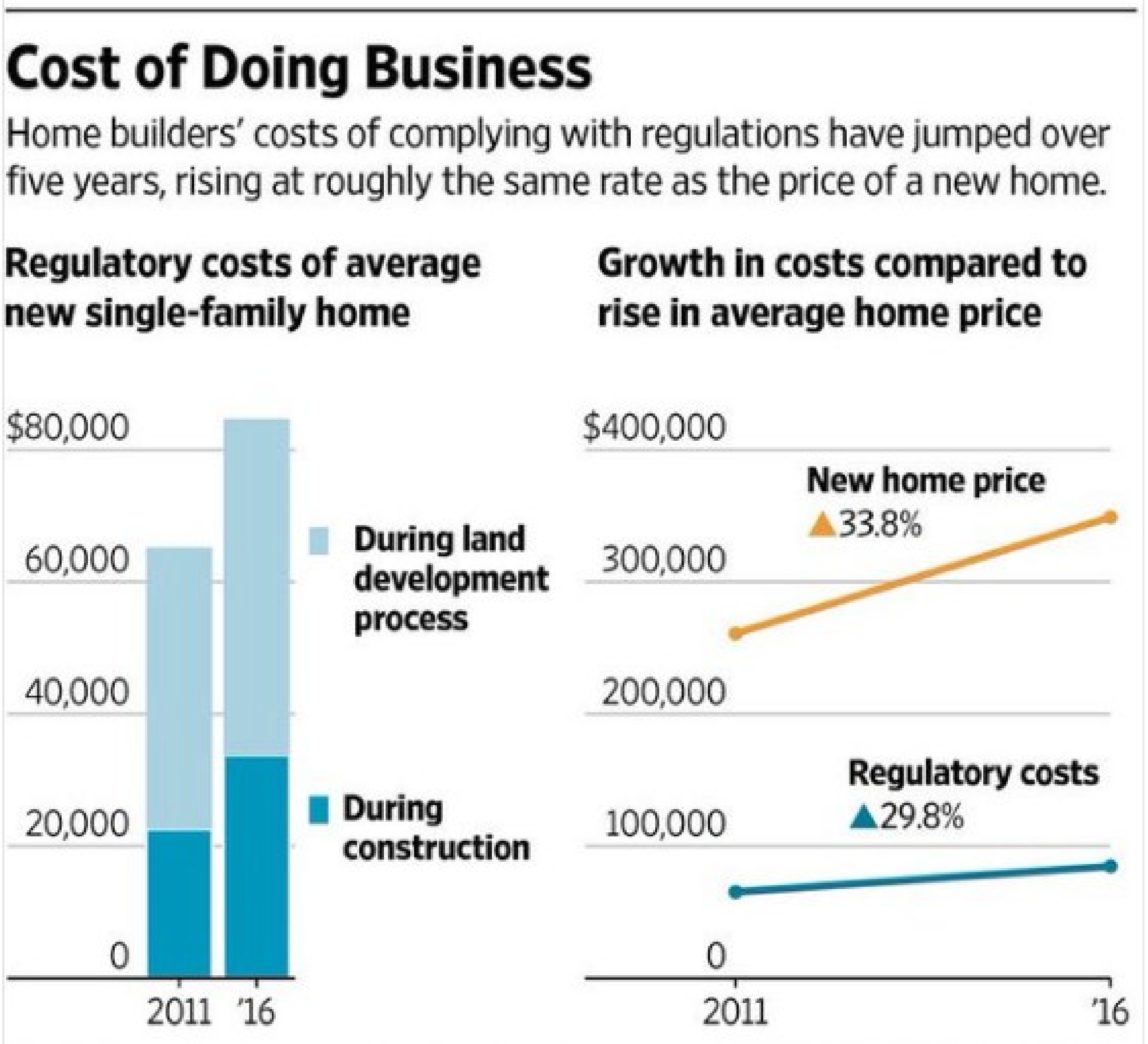
- **Source:** Via Twitter's API I downloaded two feeds that discuss the same subject matter. This eliminates the tempation to use topic modeling.

- **What Code:** To acquire the tweets in CSV form, I used a python library called “Tweepy” which requires that you have a Twitter account to work with it's API.

Ninja Economics
@NinjaEconomics

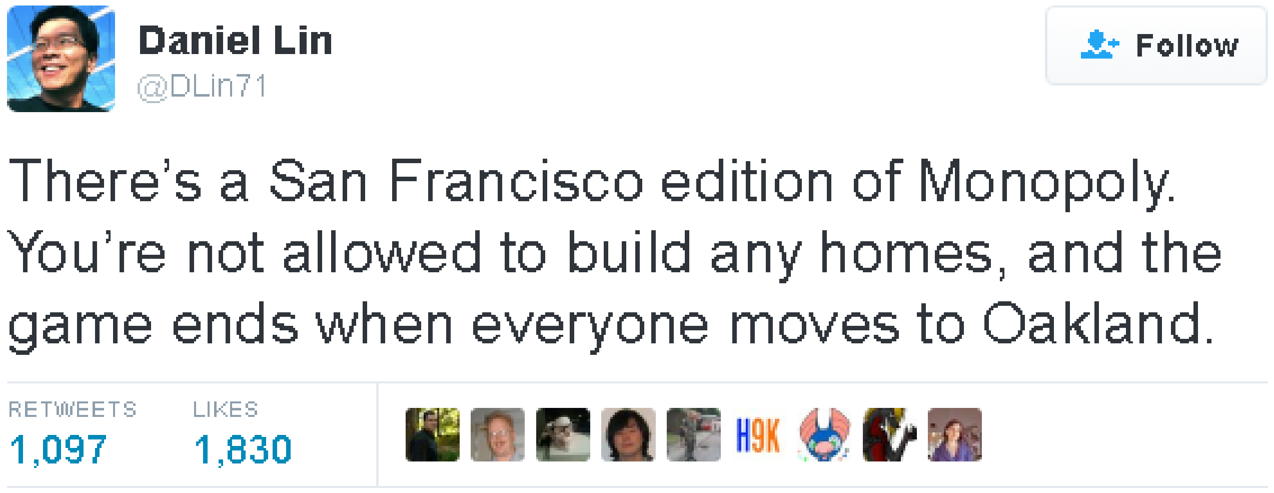
Follow

The average cost for new-home builders to comply with regulations has increased by nearly 30% over the last 5 years.



Cost of Doing Business

Home builders' costs of complying with regulations have jumped over five years, rising at roughly the same rate as the price of a new home.

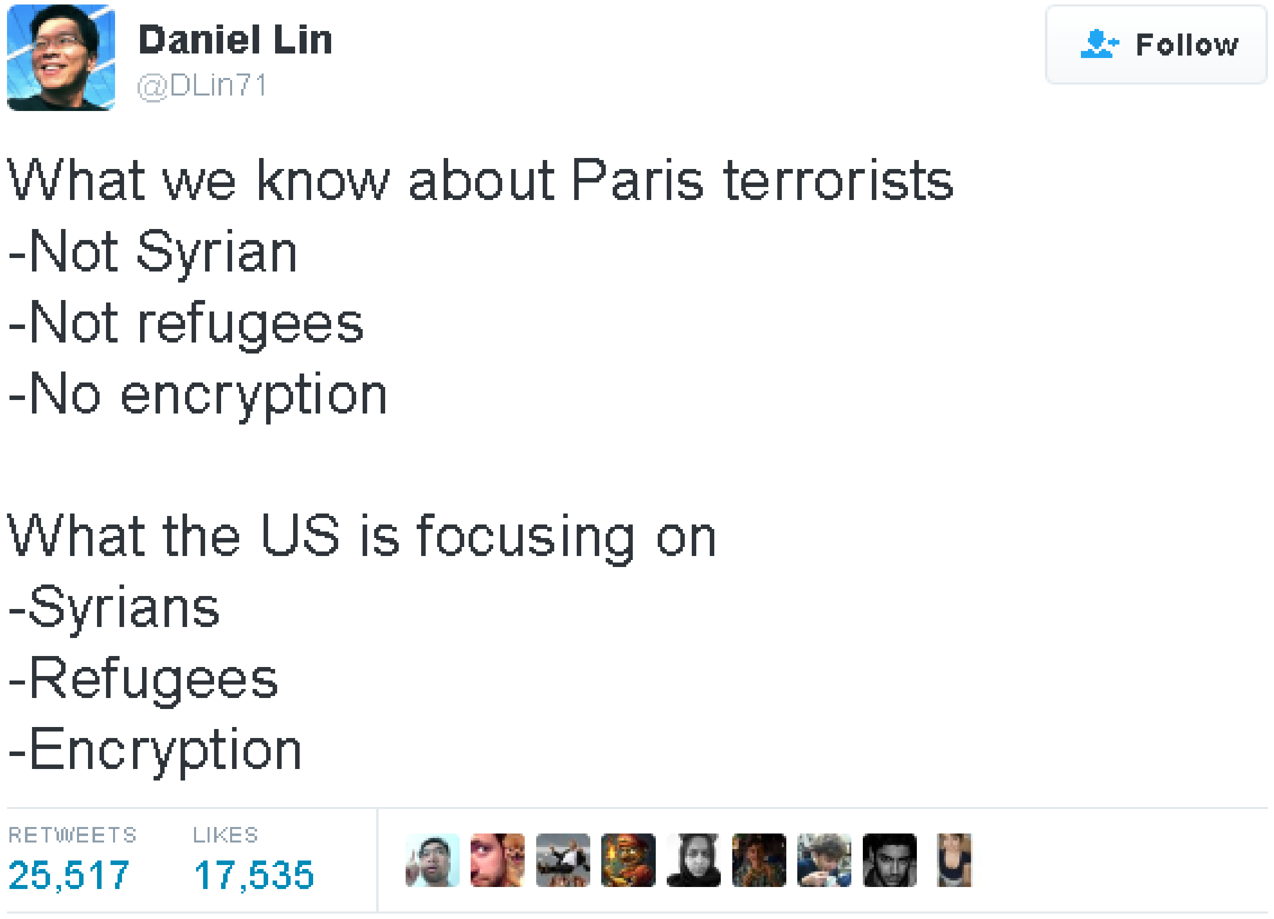


Daniel Lin
@DLin71

Follow

There's a San Francisco edition of Monopoly. You're not allowed to build any homes, and the game ends when everyone moves to Oakland.

RETWEETS 1,097 LIKES 1,830



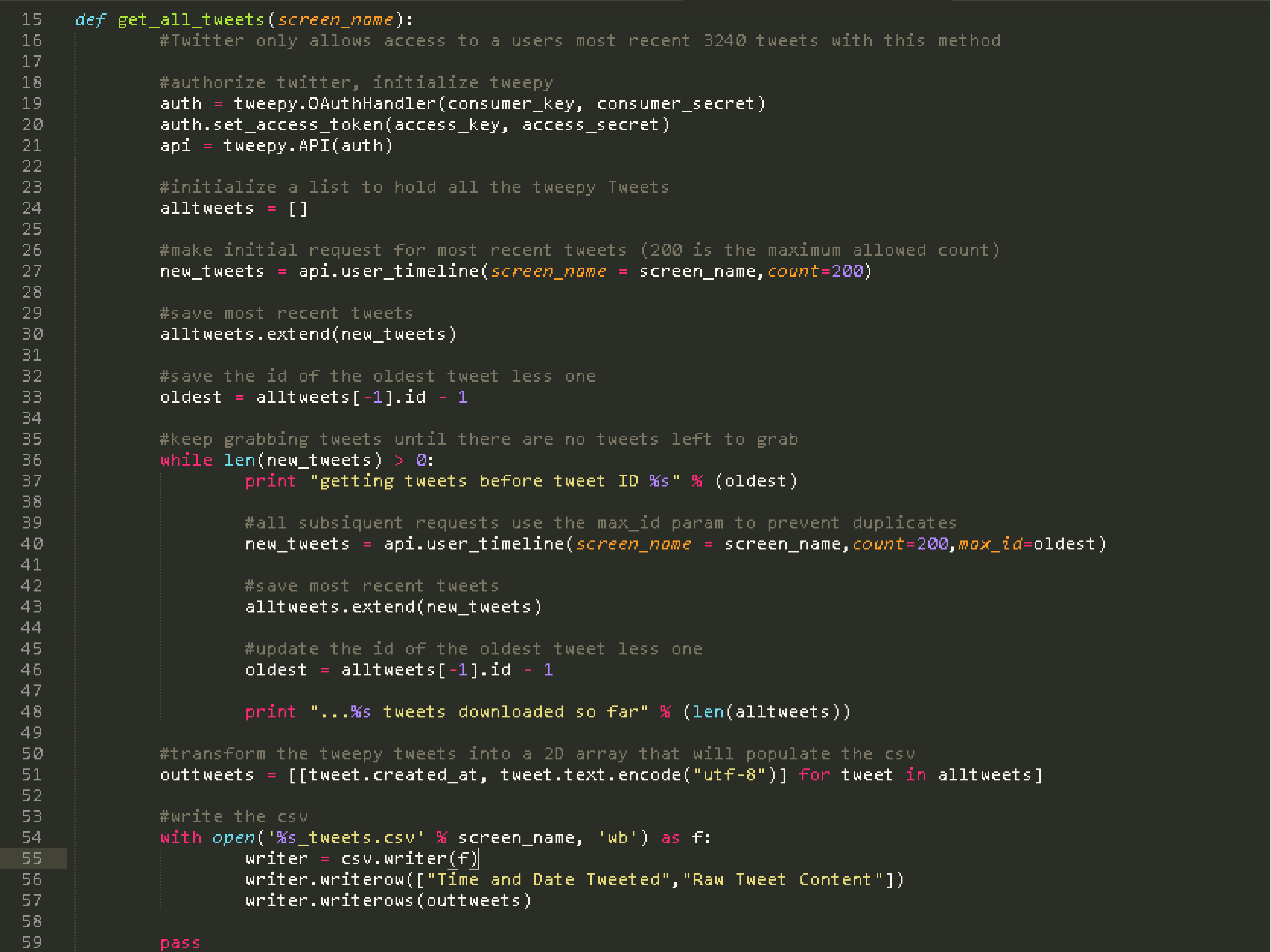
Daniel Lin
@DLin71

Follow

What we know about Paris terrorists
-Not Syrian
-Not refugees
-No encryption

What the US is focusing on
-Syrians
-Refugees
-Encryption

RETWEETS 25,517 LIKES 17,535

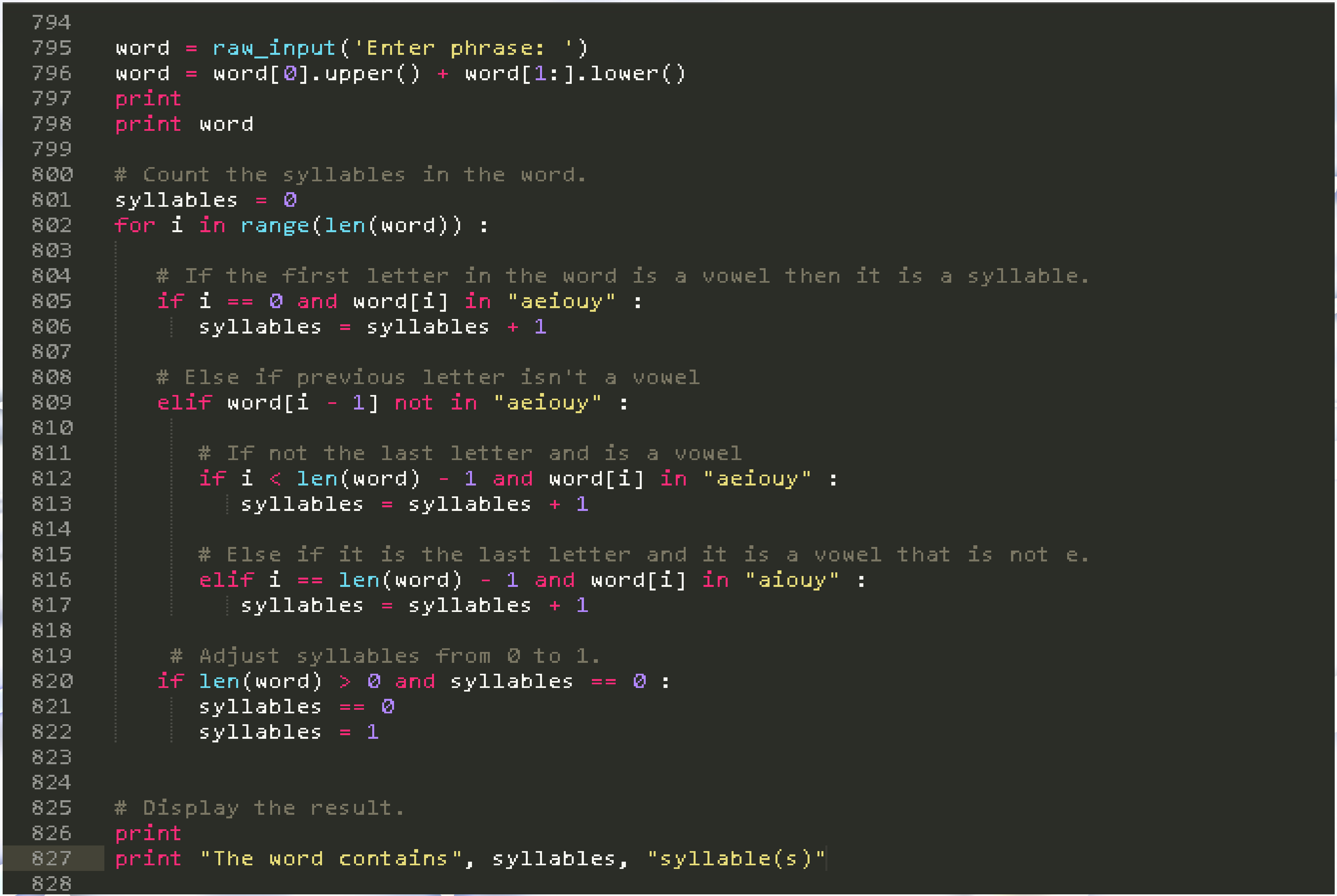


```python
def get_all_tweets(screen_name):
        #Twitter only allows access to a users most recent 3240 tweets with this method

        #authorize twitter, initialize tweepy
        auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
        auth.set_access_token(access_key, access_secret)
        api = tweepy.API(auth)

        #initialize a list to hold all the tweepy Tweets
        alltweets = []

        #make initial request for most recent tweets (200 is the maximum allowed count)
        new_tweets = api.user_timeline(screen_name = screen_name,count=200)

        #save most recent tweets
        alltweets.extend(new_tweets)

        #save the id of the oldest tweet less one
        oldest = alltweets[-1].id - 1

        #keep grabbing tweets until there are no tweets left to grab
        while len(new_tweets) > 0:
                print "getting tweets before tweet ID %s" % (oldest)

                #all subsiquent requests use the max_id param to prevent duplicates
                new_tweets = api.user_timeline(screen_name = screen_name,count=200,max_id=oldest)

                #save most recent tweets
                alltweets.extend(new_tweets)

                #update the id of the oldest tweet less one
                oldest = alltweets[-1].id - 1

                print "...%s tweets downloaded so far" % (len(alltweets))

        #transform the tweepy tweets into a 2D array that will populate the csv
        outtweets = [[tweet.created_at, tweet.text.encode("utf-8")] for tweet in alltweets]

        #write the csv
        with open('%s_tweets.csv' % screen_name, 'wb') as f:
                writer = csv.writer(f)
                writer.writerow(["Time and Date Tweeted","Raw Tweet Content"])
                writer.writerows(outtweets)

        pass
```

The Modeling

- **Logistic Regression:** Data is not numerically continuous, quantities and yes/no for features.
- **First:** Analyze twitter feed by chosen features.
- **Next:** Choose which features best distinguish one feed from another, much iteration...
- **Final:** Gather a subset of new tweets from previously selected feeds (ten or so), remove the user handles, and use model to attribute.
- **Iris Dataset:** Very similar problem, in the sense that the aim is to categorize by chosen features.

Example: Syllable Count



```
word = raw_input('Enter phrase: ')
word = word[0].upper() + word[1:].lower()
print
print word

# Count the syllables in the word.
syllables = 0
for i in range(len(word)) :

   # If the first letter in the word is a vowel then it is a syllable.
   if i == 0 and word[i] in "aeiouy" :
      syllables = syllables + 1

   # Else if previous letter isn't a vowel
   elif word[i - 1] not in "aeiouy" :

      # If not the last letter and is a vowel
      if i < len(word) - 1 and word[i] in "aeiouy" :
         syllables = syllables + 1

      # Else if it is the last letter and it is a vowel that is not e.
      elif i == len(word) - 1 and word[i] in "aiouy" :
         syllables = syllables + 1

    # Adjust syllables from 0 to 1.
   if len(word) > 0 and syllables == 0 :
      syllables == 0
      syllables = 1


# Display the result.
print
print "The word contains", syllables, "syllable(s)"
```