CS366 Assignment 3 **Due: 2020-03-22 11:59PM**

Twitter Sentiment Analysis / Machine Learning

Twitter represents a fundamentally new instrument to make social measurements. Millions of people voluntarily express opinions across any topic imaginable, and researchers as well as business have used it for a variety of applications. For example, researchers have shown that the "mood" of communication on twitter reflects biological rhythms and can even used to predict the stock market[1], and a student the University of Washington used geocoded tweets to plot a map of locations where "thunder" was mentioned in the context of the recent storms[2]. Researchers from Northeastern University and Harvard University studying the characteristics and dynamics of Twitter have an excellent resource for learning more about this area[3].

In this assignment, you will use the Sentiment140 corpus, which was accumulated automatically using the Twitter Search API. It was also *annotated* automatically, assuming that any tweets with positive emoticons, like :), are positive, and tweets with negative emoticons, like :(, are negative. You will be working with a subset of these tweets. You will use the data in this file, which consists of 11,759 tweets drawn from the Sentiment140 corpus. The file has the following format, with the two fields separated by a "tab" character:

1 the polarity of the tweet (0 = negative emotion, 4 = positive emotion)
2 the text of the tweet (e.g., Lyx is cool)

Your tasks

1. Pre-processing, tokenizing, and tagging

For preprocessing, tokenization, and tagging, you can use the ARK tweet tokenizer and tagger. You can download and install this on your own machine from this page or use the department machines. You should train using the model that uses the Penn PoS tagset. N.B.: The tagger can make mistakes!

The input to this program is the text of the tweets, one per line. So you should first create a file of the tweets from the data file containing only the tweet text (however, be sure to retain the information about the sentiment score for each tweet, as it will need to be restored later).

To invoke ARK on the department machines, use the command

```
ark-tagger --model /usr/csapps/share/model.ritter_ptb_alldata_fixed.20130723
<input_file>
```

To invoke ARK on your own machine, you can copy the file `model.ritter_ptb_alldata_fixed.20130723` from the department machines using the path given above. The file is also in the assignment directory for your convenience.

Once you have the model file, `cd` to the directory containing the ARK .jar file, be sure the model file is in the same directory, and enter the following on a command line[4]:

[1] See http://arxiv.org/pdf/1010.3003&embedded=true
[2] http://cliffmass.blogspot.com/2012/07/thunderstorm-fest.html
[3] http://www.ccs.neu.edu/home/amislove/twittermood/
[4] Note: If you get an error about lack of permissions, enter the following at the command line: `chmod a+rx *` and try again.

```
./runTagger.sh --model model.ritter_ptb_alldata_fixed.20130723 <input_file>
```

In both instances, `<input_file>` is the file containing the tweets. (Note that this command may need some modification for platforms other than Mac OSX.)

Suggestion: create a file of, say, 100 tweets to use for program development before running ARK on the whole file.

The output of ARK looks like this:

```
I predict I won't win a single game I bet on .  Got Cliff Lee today , so if
he loses its on me RT @e _one :  Texas ( cont ) http://tl.gd/6meogh PRP VBP
PRP MD VB DT JJ NN PRP VBP IN . VBD NNP NNP NN , RB IN PRP VBZ PRP IN PRP
RT USR CD : NNP -LRB- NN -RRB- URL 0.9982 0.8812 0.9981 0.7486 0.9889 0.9905
0.9420 0.9694 0.9962 0.9690 0.9544 0.9973 0.6782 0.9824 0.9999 0.9897 0.9889
0.7855 0.9935 0.9981 0.9790 0.8629 0.9886 0.9701 0.9914 0.9978 0.4124 0.9796
0.9889 0.9306 0.7925 0.9411 0.9848 I predict I won't win a single game I bet
on .  Got Cliff Lee today , so if he loses its on me RT @e_one :  Texas (
cont ) http://tl.gd/6meogh
```

All this information is on a single line. Note that the PoS tags are given following the text, in the order of the words in the tweet. The numbers in the next field are confidence scores for the tag assignments, which can be ignored. The final field contains the text of the tweet again, this time in original format if ARK modified any characters. This last field can also be ignored. The tweet, PoS, confidence scores, and tweet(2) parts of the line are separated by tabs. For reading this in to a Python program, consider using the `split` command to separate the fields by tabs (e.g., to get the first part of the line before the tab, use `tweet.split("\t")[0]`).

Notes on the ARK output:

1. HTML character codes (i.e., &...;) are replaced with an ASCII equivalent.[5]
2. Each token is tagged with its part-of-speech; some special tags not in the Penn tagset include:
   - URLs (i.e., tokens beginning with http or www) are tagged as URL.
   - Twitter user names (beginning with @) are tagged as USR.
   - Hash tags (beginning with #) are tagged as HT.

In other words, ARK does a lot of normalization for you.

The next step is to take the ARK output and format it for later input to the program in Task 2 (below). This program should be called *twtt.py* program and take two arguments: the input .tsv filename and the output filename. For example, if you want to create the file *train.twt* from the train data, you would run:

```
python3 twtt.py training.11000.processed.noemoticon.tsv train.twt
```

The output of this program should be the tweets, one per line, in the following format:

```
<sentiment> <tab> <tagged tokens of the tweet>
```

where

[5] See this table for a list.

- `<sentiment>` is the numeric sentiment class of the tweet (0 or 4),
- `<tab>` is a tab character
- Each tagged token consists of a word, the '_' symbol, and the PoS tag (e.g., dog_NN).

For example,

```
4 I_PRP predict_VBP I_PRP won't_MD win_VB a_DT single_JJ game_NN I_PRP bet_VBP
on_IN ._.  !!!!_!  ,_,
0 I_PRP 'm_VBZ going_VBP to_IN predict_VB WINNER_NN OMG_UH 45_CD
```

See the Appendix for a list of the Penn Treebank tags. The output of *twtt.py* is used in Task 2 (see below).

Note that ARK does very badly with contracted words, such as "I'm", "it's", etc.; here is an example:

```
I'm I'll it's PRP PRP PRP
```

A nice enhancement (not required, but certainly rewarded) would be to replace ARK output like the above with the actual tag sequence:

```
I am I will it is PRP VBP PRP MD PRP VBZ
```

2. Gathering feature information

The second step is to write a Python program named *build_features.py*, in accordance with the general specifications section below, that takes tokenized and tagged tweets from Task 1 and builds a datafile containing feature vectors that will be used to classify tweets in Task 3. You can start with this working version of the code to create the file from plain text input. There is a tiny sample input file and the file generated from it in the same directory, as an example.

You should augment the file that generates the features as you see fit to add features and feature values that you think would be useful.

Feature extraction is basically the process of analyzing the preprocessed data in terms of variables that are indicative or discriminative of the classes. For example, if the use of the word "me" is indicative of a negative affect, then the number of first-person pronouns in a tweet will tell you something about the mood of the tweet. Each line in the feature file represents a single tweet encoded by a list of numbers (i.e., the features) separated by commas, and the class (affect) of the tweet, which is the last entry, e.g., in

```
0,1,3,0,0..., 1
```

the final 1 refers to the positive class.

The *build_features.py* program should take two arguments: (1) the input filename (i.e., the output of *twtt.py*), and (2) the filename of the output file.

N.B.: IDLE does not provide for passing parameters to a program. Therefore, if you are using IDLE you should develop your program with file names hard-coded (and on a small subset of the data), then modify to accept parameters and run it from the command line, as shown above.

2.1. The features to be gathered

For each tweet, you need to initially extract the 18 features given in the template file and write these, along with the class, to the output file. These features are listed in the Appendix. Many of them involve counting tokens in a tweet with certain characteristics that can be discerned from its tag. For example, counting the number of adverbs in a tweet involves counting the number of tokens that have been tagged as RB, RBR, or RBS. Be careful about capitalization; in all cases you should count both capitalized and lower case forms (e.g., both "he" and "He" count towards the number of third person pronouns).

Save your results in a file called *all_tweets.csv*.

3. Classifying tweets using sklearn

The third step is to use the feature extraction program from Task 2 to classify tweets using the sklearn machine learning module. If you are using your own machine you will need to install `sklearn` and `pandas` using `pip3 install <module>`; they are already installed on the department machines. This file provides a starting point.

We will now experiment a little.

3.1 Feature augmentation

This is an important part of the exercise! Explore alternative features to those initially included in `build_features-nominal.py`. What other kinds of features would be useful in distinguishing affect? Test your features empirically and discuss your findings. Some resources for this are available in the course `data/sentiment/lists` directory, which includes a number of word lists used in sentiment analysis. There are also lots of sentiment word lists available online, including several specifically geared to Twitter, that can be found by googling.

3.2 Classifiers

Report classification accuracy on test data among support vector machines (SVMs), Gaussian Naïve Bayes, Logistic Regression, and Random Forest[6]. Save the output of the best classifier to the file *3.1output.txt*. Use the best classification algorithm for the following questions.

3.3 Amount of training data

Many researchers attribute the success of modern machine learning to the sheer volume of data that is now available. Modify the amount of data that is used to train your system in increments of 500, starting at 500 (i.e., n = 500, 1000, ..., 5500) and report the resulting accuracies in a table in *3.2output.txt*. In that file, also comment on the changes to accuracy as the number of training samples increases, including at least two sentences on a possible explanation.

3.4 Feature analysis

For each of $n = 500$ and $n = 5500$ in Section 3.3, run sklearn's feature selector `SelectKBest` and copy the output into the file *3.3output.txt*. The template file includes a method to do this; it uses

[6] You can run all of these classifiers with default parameters, but Random Forest will do better with `n_estimators=403`.

$k = 5$ and `chi2`, but feel free to experiment with other values of k and other scoring algorithms (see this web page for more information). What features, if any, retain their importance at both low and high(er) amounts of input data? Provide a possible explanation as to why this might be.

Bonus Points

You can access current tweets using Twitter's Application Programming Interface (API), which is easily accessed from Python. To set up the API for use within Python, you have to do the following:

1. Install `oauth2` on your machine. To do this, run the following command to get the latest package:

   ```
   pip3 install -U oauth2
   ```

2. Set up a Twitter Developer account, as follows:

   a. Create a twitter account if you do not already have one.

   b. Go to https://dev.twitter.com/apps and log in with your twitter credentials.

   c. Click "Create new application".

   d. Fill out the form and agree to the terms. Put in a dummy website (http://localhost.com, for example) if you don't have one you want to use.

   e. On tab "API Keys", Click "Create my access token."

   You will be shown a list of OAuth Settings, including *consumer key, consumer secret, access token, and access secret.*

3. Install the TwitterSearch module for accessing twitter from the command line, follows:

   ```
   sudo pip3 install TwitterSearch
   ```

A file with a sample of code for fetching tweets on a particular topic using TwitterSearch is here. There is also ample documentation for the TwitterSearch module.

You are now ready to use TwitterSearch to access tweets from Python. To do this, start IDLE or whatever Python IDE you are using, and load up the provided TwitterSearch template. You will have to fill in the information obtained earlier in this part of the code:

```
>>> ts = TwitterSearch(
            consumer_key='<Enter consumer key>',
            consumer_secret='<Enter consumer secret>',
            access_token_key='<Enter your access token key here>',
            access_token_secret='<Enter your access token key here>')
```

Now, you can get tweets on a specified topic by setting the keywords, as in the line

```
tso.set_keywords(['Trump', 'Rubio']).
```

(Sorry, my examples are from an old experiment!)

Result:

```
 @TeachESL tweeted:  RT @jonathanchait:  Hey, Rubio &amp; Cruz, here's a good story
to use against Trump.  Black students evicted from Trump rally for no reason.  http...
@Abby669314511 tweeted:  RT @AlecMacGillis:  While @reihan urges GOP to respond to
Trump by de-prioritizing big donors, Rubio names as finance chair hedge fund zilli...
@B_Green159 tweeted:  RT @irritatedwoman:  Donald Trump calls on Marco Rubio to drop
out - POLITICO https://t.co/hSxeUmd9ry
@TheMoonPoolLife tweeted:  Trump calls on Rubio to drop out https://t.co/R4rYEh1OoS
@NathanWurtzel tweeted:  Can agree/disagree on imm.  reform; can't deny Rubio brave
to try that and to take Trump on while Cruz cringes.  https://t.co/IaBQq8DuFX
@MJGuasto9498 tweeted:  RT @politico:  Trump calls on Rubio to drop out
https://t.co/bOvfFmco3q | AP photo https://t.co/OkHleLGXco
@iko1iko1 tweeted:  RT @ChristopherJM: You can vote for Cruz, Trump or Rubio in a south
London pub's '#pisspoll' https://t.co/ycWTVflBtr via @mashpolitics http...
@dianamee tweeted:  RT @20committee:  This cuts to the heart of why Trump is rising...and
why Rubio is in deep trouble.  https://t.co/mZ846i04jw
@RestoreHistory tweeted:  RT @dcexaminer:  POLL: Clinton beats Donald Trump but loses
to Marco Rubio and Ted Cruz in a general election https://t.co/kGPzN18td5 https:...
@JajsaArthur tweeted:  @RickCanton @marcorubio instead just bubble Trump instead of
Rubio.  He is not winning Florida.  I live there Trump right now is beating Rubio
@Lestatbp tweeted:  RT
@GodlessUtopia:  Super Tuesday Poll
```

You can access additional information as described in the documentation, or print just the text of each tweet:

```
>>> print (tweets['text']
```

The "user" key references an embedded dictionary that includes a lot of other information; you can for example print the location associated with a tweet as follows:

```
>>> print( '@%s tweeted:  %s' % ( tweet['user']['location'], tweet['text']
) )
@District of Columbia, USA tweeted:  RT @TimothyNoah1:  CNN national poll:
Trump wins 42 percent of college-educated Republicans, more than twice Rubio's
19 and Cruz's 15.  https...  @Edinboro, PA tweeted:  RT @DRUDGE_REPORT: CAN
RUBIO WIN ANYTHING? https://t.co/QHCJuO1xYd
```

Suggestions:

1. Access a (sufficiently large) set of tweets on a given topic or topics, and use the models you trained in the assignment to evaluate the predominant sentiment.
2. Access tweets from two or more locations (e.g., New York and Nebraska) and analyze sentiment for a given topic in each location to see how they compare.

Submission

This assignment will be submitted by including it in your github classroom repository. You should include the following:

1. All your code for *twtt.py*, *build_features.py*, and *sklearn_template.py*.
2. The first 100 lines of the tagged and tokenized file from Task 1.
3. The first 100 lines *all_tweets.csv* file from Task 2.
4. *3.1output.txt*: Report on classifiers.
5. *3.2output.txt*: Report on the amount of training data.
6. *3.3output.txt*: Report on feature analysis.
7. Any lists of words that you modified from the original version.
8. README describing the files and their contexts.

Appendix

1. Conversion from raw tweets to tagged tweets

Raw tweet:

```
Meet me today at the FEC in DC at 4. Wear a carnation so I know
it's you. <a href="Http://bit.ly/PACattack" target="_blank"
class="tweet-url web" rel="nofollow">Http://bit.ly/PACattack</a>.
```

Output from *twtt.py*:

```
Meet_VB me_PRP today_NN at_IN the_DT FEC_NN in_IN DC_NN at_IN 4_NN ._.
Wear_VB a_DT carnation_NN so_RB I_PRP know_VB it_PRP 's_POS you_PRP ._.
```

2. Base set of features to be computed for each text

Counts:

- First person pronouns
- Second person pronouns
- Third person pronouns
- Coordinating conjunctions
- Past-tense verbs
- Future-tense verbs

- Commas
- Colons, semi-colons
- Dashes
- Parentheses
- Ellipses
- Common nouns
- Proper nouns
- Adverbs
- wh-words
- Modern slang acroynms
- Words all in upper case (at least 2 letters long)
- Average length of tokens, excluding punctuation tokens (in characters)

3. Definitions of feature categories

First person:

I, me, my, mine, we, us, our, ours

Second person:

you, your, yours, u, ur, urs

Third person:

he, him, his, she, her, hers, it, its, they, them, their, theirs

Coordinating conjunctions (CC)

and, but, for, nor, or, so, yet

Past tense:

VBD, VBN

Future Tense:

'll, will, gonna, going+to+VB

Common Nouns:

NN, NNS

Proper Nouns:

NNP, NNPS

Adverbs:

RB, RBR, RBS

wh-words :

WDT, WP, WP$, WRB

Modern slang acronyms:

smh, fwb, lmfao, lmao, lms, tbh, rofl, wtf, bff, wyd, lylc, brb, atm, imao, sml, btw, bw, imho, fyi, ppl, sob, ttyl, imo, ltr, thx, kk, omg, ttys, afn, bbs, cya, ez, f2f, gtr, ic, jk, k, ly, ya, nm, np, plz, ru, so, tc, tmi, ym, sol