Twist:

User Timeline Tweets Classifier

By

Sonali Joshi

Priya Iyer

Vaidyanath Venkat

Code Documentation

Date: 12/06/2012.

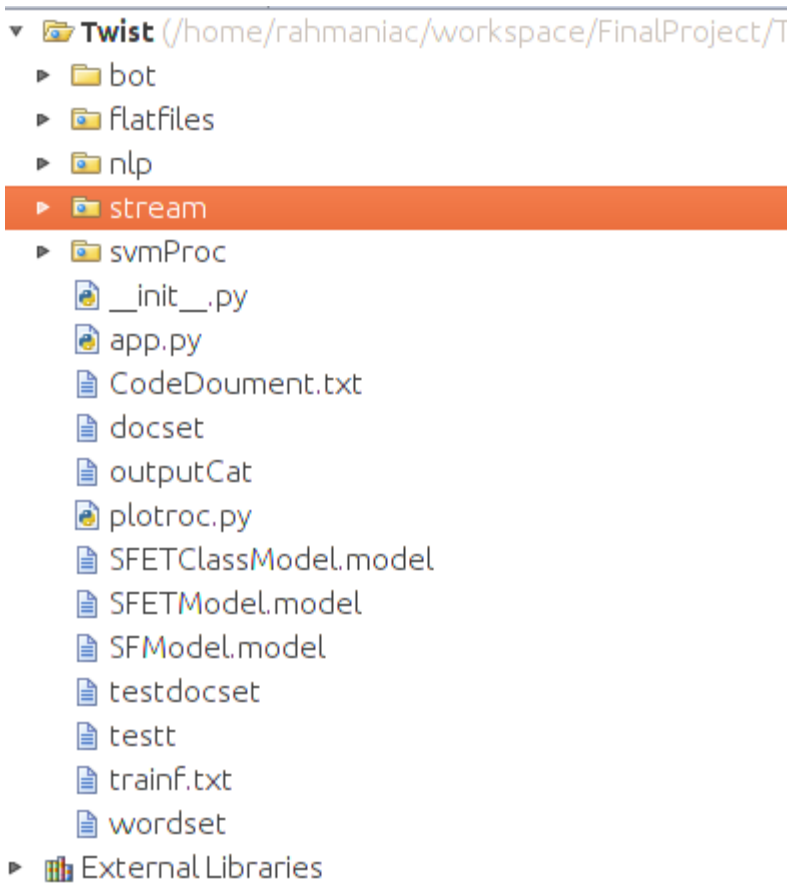
Mentor:

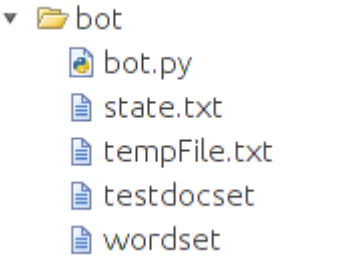
Andy Schlaikjer

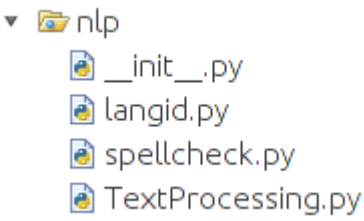
Instructor

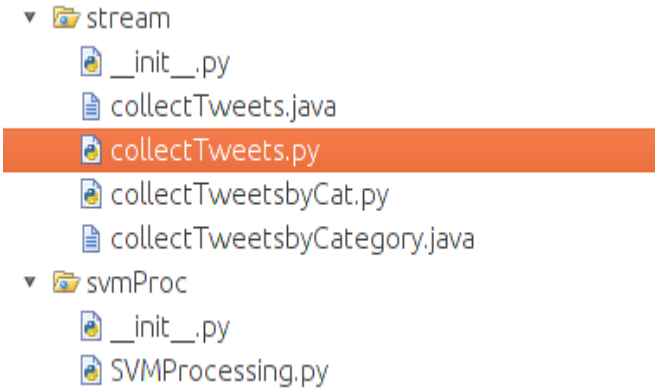
Marti Heart

1. **Project Structure**









2. **Modules :**

BOT

This module contains the codea nd logic for the twitter bot. The functioanlities include the following

1. Authenticate the app and connect to twitter API
2. Fetch the bot's home timeline
3. Get a list of tweets for clasfication
4. Process tweets and classify them
5. Save the last tweeted id to a file
6. Retweet each tweet along with their category

NLP

All the text processing code is contained in this module.

TextProcessing.py

This is the interface that is called during the trainign and classification. From here the calls to the spell check module is made

1. Generate Unigrams
2. Filter punctuations.
3. Reduce strings. spell correct, stem
4. Generate bigrams
5. Stop words removal
6. Language Identification

Spell check.py

This contains the logic needed to implement the Norvig's spell check.

1. Determine Part of speech and return all proper nouns without any check
2. Detemrine if any word contains the same letter more than three times consecutively (ex : cuuuuttteee)
3. Reduce such strings to their normal spelling.

4. For all words, compare with NLTK's dictionary and run spell check if needed.

Spell check:

1. For each word, split the word into 2 at all possible positions. - Ex -> hell is split as (h,ell), (he,ll), (hel,l), (hell,) etc
2. This becomes the input for the remaining steps
3. For each pair in the list we do the following
    a. Add a new letter from a though z at every possible position. So 'hell' yields ahell,bhell...zhell, haell,hbell..hzell etc.
    b. Edit each letter and substitute that with a through z. Ex, 'hell' yields aell,bell,...zell, hall,hbll..hzll etc..
    c.Delete each letter. so hell becomes, ell, hll, and hel etc.
    d. swap every pair of letters. For ex -> hell yields ehll, hlel, hell etc.
4. Create a set of distinct suggestions after these 4 operations.
5. Fitler out all invalid words like 'aell'.
6. retruns the final list.

LangId.py

This contains the logic as suggested by Misja Hoebe to use nltks trigram sets to detect language

1. Nltk has trigram sets for many languages with their wieghts in their corpus
2. From the corpus load the weights for english, french spanish
3. From the tweets, genrate trigrams and their frequencies.
4. This is then compared with the nltk corpus for a language match.

SVM

This directory contains the ML code requried to classify the tweets.

SVMProcessing.py

Contains the core logic that trains and creates the model for classification, classifies the tweets and finally calculates metrics for the classification.

1. Loads the tweets into a global cache. this contains the tweet and word ids for all tweets in the set.

2. Creates different models by varying the cost and other parameters and trains a set of tweets.
3. Calculates the metrics for each model by computing the precision and recall values and decides on the apt model to be used.
4. Once the model is decided, the tweets can be classifed using the model.
5. Liblinear package is used to train the machine.