MoodMinder - setup and execution

In this section we describe how to set up MoodMinder and run it locally. MoodMinder runs in two forms, one that classifies and recommends actions for past tweets and one that classifies and recommends actions for streaming tweets. Both of the them have the same setup procedure described below.

Steps to setup and run MoodMinder

MoodMinder uses the Django framework as described in Section 3c in the final report. In order to set up and run the app, the user should install the following tools.

1. `python` == 2.7
2. `django` == 1.9
3. `django-nose` == 1.4.1
4. `nose` == 1.3.7
5. `requests` == 2.7.0
6. `wsgiref` == 0.1.2
7. `nltk` along with its data set
8. `sklearn`
9. `numpy`
10. `scipy`
11. `pandas`
12. `re` (regular expression)
13. `emoji`

Once the tools listed above are installed, switch to the directory moodminder_test and start the server in the local machine by entering the following command

```
python manage.py runserver
```

This starts the server in localhost at port 8000. Once the server is up and running, the user can use MoodMinder in two ways.

1. Classifying and recommending actions for past tweets

Open `127.0.0.1:8000/app/profile_past`. The webpage should look similar to Screenshot 1 in Section 3d in the final project report. The user can enter any twitter handle, usually his/her own handle. After selecting the 'Get latest tweet' option, MoodMinder classifies and recommends positive action for the most recent tweet. The user should enter a twitter handle every time he/she wants to see the most recent tweet of the corresponding twitter user. At present, MoodMinder aggregates all results and displays them along with the twitter handle,

sentiment and recommendation. Sample outputs are shown in Section 3d in the final project report.

2. Classifying and recommending actions for streaming tweets

Open `127.0.0.1:8000/app/profile_stream`. The webpage should look similar to Screenshot 3 in Section 3d in the final project report. The user can enter any twitter handle, usually his/her own handle. After selecting the 'Get latest tweet' option, MoodMinder starts listening for new tweets from the corresponding twitter user. Every time a new tweet is received, MoonMinder stores it in memory and they are displayed whenever the 'Refresh' button is selected. Like the previous scenario, MoodMinder aggregates all results and displays them along with the twitter handle, sentiment and recommendation. In the current version, the user has to manually click 'Refresh' in order to see the new tweets. One possible future work involves auto refreshing the webpage when new tweets are received. Sample outputs are shown in Section 3d in the final project report.

We also briefly describe the source code of MoodMinder. The complete details can be found in the source files moodminder_test/app/.

Description of source code

All the file names in this section are relative to `moodminder_test/app/`.

1. `views.py` - part of django framework that sets up the web ages for viewing past (profile_past) and streaming tweets (profile_stream).
2. `TweetProcess.py` - acts as part of the controller and the model in the MVC architecture. `TweetProcess` invokes the underlying model to classify the tweet and recommend positive actions, and updates the view when the results are obtained.
3. `TweetClassify.py` - Implements the classifier to classify incoming tweets into positive or negative.
4. `TweetRecommend.py` - Implements the recommender system that extracts keywords from the tweets and recommends positive actions from a pre-populated database based on the category generated.
5. `TweetView.py` - Implements the view; the current implementation is a simple view that simply print the output to stdout.
6. `StdOutListener.py` - Implemented the stdout listener that listens for incoming tweets when MoodMinder runs in streaming mode.
7. `Feat_extractor.py` - Extracts unigram and bigram features used by the classifier
8. `Util_*py` - Simple utility functions.
9. `Tests.py` - unit tests for functions in `Tweet*.py`. This is explained in Section 4a in the final project report.