User Manual

Erin Oerton, Richard Everett, Philipp Boeing

April 26, 2014

1 Prerequisites

To use our method, the following tools need to be installed:

- Python 2.7
- SentiWordNet for Python
- Scikit Learn for Python

2 Feature Extraction

2.1 Parsing the raw dataset

Run `process_data.py` to parse the raw tweet file downloaded from UCL Twitter Collector into tab-separated files of tweets for each company, which can be opened by spreadsheet programs.

2.2 Filtering of Tweets

The provided dataset needs some pre-processing before feature extraction. The data must first be filtered to remove duplicate tweets which would distort the results (we filtered on the tweet body only in order to leave retweets in the data). Next, we want to remove tweets which appear to be auto-generated or 'spam' tweets. For this particular data, we found the most effective way to do this is to remove tweets containing the word 'free' (the Microsoft tweets contained many 'free XBox' type tweets).

The data is supplied in .csv format, so we carried out filtering in MS Excel rather than writing a dedicated script - this was more efficient on the large dataset, but this filtering could easily be acheived using Python for smaller datasets.

2.3 Generating sentiment values per tweet

Next, use the Python script `preprocessing_sentiment_classifier.py` to read the tweet text files. The list of files to be read can be specified at the top of the script:

```
company = 'microsoft'

if company == 'jpmorgan':
  files_to_process = ['jpmorgan.txt']
elif company == 'microsoft':
  files_to_process = ['microsoft1.txt','microsoft2.txt']
elif company == 'mcdonalds':
  files_to_process = ['mcdonalds1.txt','mcdonalds2.txt',
  'mcdonalds3.txt']
```

2.4 Generating daily sentiment values per company

Next, the Python script `preprocessing_group_time.py` reads the previously generated file containing the raw sentiment values for each tweet and aggregates them into a daily vector. It also calculates the daily volume and normalizes it to account for missing data.

The input files are set at the beginning of the script in the same way as the previous file.

3 Financial Data

Stock market opening and closing prices were downloaded as a CSV file from `finance.yahoo.com`. The file was then converted using Excel into `model_stock.py` which stores the values as a Python datastructure. To add new data, it must be encoded in a similar structure.

4 Model creation

Models are trained and evaluated by `model.py`. The script can be modified to create new models.

Model training and evaluation is kicked off in the main block:

```
if __name__ == '__main__':
    mcdonalds = read_data('./../output/mcdonalds_daily.txt')
    experiment_zero(mcdonalds,'mcdonalds')
```

The parameters to be used are specified at the top of `experiment_zero()`:

```
finance_datatype = 0
finance_n = 2
sentiment_datatype = 1
sentiment_n = 1
day = 0
target = 0
volume = 0
```

- **finance_datatype** : Integer
  2 = Stock price change,
  1 = Percentage stock price change
  0 = Only direction

- **finance_n** : Integer ≥ 0 Number of days of finance data to include
- **sentiment_datatype** : Boolean
  1 = all sentiment features
  0 = Total
- **sentiment_n** : Integer ≥ 0 Number of days of sentiment data to include
- **day** : Boolean 1 = Include day of the week, 0 = do not
- **target** : Boolean 1 = Amount, 0 = Direction
- **volume** : Boolean 1 = Yes, 0 = No