User-Level Twitter Polarity Classification with a Hybrid Approach

Documentation

# Data Collection:

Tweets are collected using Test.py and stored in csv file. Tweets we used for performance evaluation can be found in 1412778302.xls and 2569997099.xls.

# Data Cleansing:

Data cleansing can be done by calling the method tweet.cleanse() which can be found in Tweet.py. This method takes a tweet string as input and transform the tweet into a 4-element tuple. Please see appendix for a detailed documentation of this method.

# Data Labeling:

Data labeling with lexicon-based method can be done by calling the method tweet.get\_polarity(). This method takes a cleansed tweet (4-element tuple), POS tagger and lexicon lists as input and output a polarity score and a subjectivity score.

Lexicon augmentation is done by running the following code:

*with* open('sub\_chi\_square.csv', 'w') *as* csvfile:  
 fieldnames = ['Word', 'Chi-square', 'In Sub', 'In Obj']  
 writer = csv.DictWriter(csvfile, *fieldnames*=fieldnames)  
 writer.writeheader()  
 *for* word *in* all\_words:  
 f11, f12, f21, f22 = 0, 0, 0, 0  
 chi\_square = 0  
 *for* line *in* sub\_tweet:  
 *if* word *in* line: f11 += 1  
 *else*: f12 += 1  
 *for* line *in* obj\_tweet:  
 *if* word *in* line: f21 += 1  
 *else*: f22 += 1  
 f11e = (f11 + f21) \* (f11 + f12) / (f11 + f12 + f21 + f22)  
 f12e = (f12 + f22) \* (f11 + f12) / (f11 + f12 + f21 + f22)  
 f21e = (f11 + f21) \* (f21 + f22) / (f11 + f12 + f21 + f22)  
 f22e = (f12 + f22) \* (f21 + f22) / (f11 + f12 + f21 + f22)  
 n = f11 + f12 + f21 + f22  
 *if* f11e != 0 *and* f12e != 0 *and* f21e != 0 *and* f22e != 0:  
 chi\_square = (abs(f11 - f11e) - 0.5) \*\* 2 / f11e + (abs(f12 - f12e) - 0.5) \*\* 2 / f12e + \  
 (abs(f21 - f21e) - 0.5) \*\* 2 / f21e + (abs(f22 - f22e) - 0.5) \*\* 2 / f22e  
 *if* chi\_square > 1:  
 writer.writerow({'Word': word, 'Chi-square': chi\_square, 'In Sub': f11, 'In Obj': f21})

Word with chi-square score greater than 1 will be stored into a csv file. The result of the code can also be found in 1412778302.xls and 2569997099.xls.

# Classification:

Subjectivity classification can be done by running Subjectivity.py and polarity classification can be done by running Polarity.py.

Appendix:

Python files:

1. File name: Polarity.py

This file contains codes for polarity classification.

1. File name: Subjectivity.py

This file contains codes for generating subjectivity indicators and subjectivity classification.

Use the code below to identify subjectivity indicator. When performing subjectivity classification, comment out this part of code.

*with* open('sub\_chi\_square.csv', 'w') *as* csvfile:  
 fieldnames = ['Word', 'Chi-square', 'In Sub', 'In Obj']  
 writer = csv.DictWriter(csvfile, *fieldnames*=fieldnames)  
 writer.writeheader()  
 *for* word *in* all\_words:  
 f11, f12, f21, f22 = 0, 0, 0, 0  
 chi\_square = 0  
 *for* line *in* sub\_tweet:  
 *if* word *in* line: f11 += 1  
 *else*: f12 += 1  
 *for* line *in* obj\_tweet:  
 *if* word *in* line: f21 += 1  
 *else*: f22 += 1  
 f11e = (f11 + f21) \* (f11 + f12) / (f11 + f12 + f21 + f22)  
 f12e = (f12 + f22) \* (f11 + f12) / (f11 + f12 + f21 + f22)  
 f21e = (f11 + f21) \* (f21 + f22) / (f11 + f12 + f21 + f22)  
 f22e = (f12 + f22) \* (f21 + f22) / (f11 + f12 + f21 + f22)  
 n = f11 + f12 + f21 + f22  
 *if* f11e != 0 *and* f12e != 0 *and* f21e != 0 *and* f22e != 0:  
 chi\_square = (abs(f11 - f11e) - 0.5) \*\* 2 / f11e + (abs(f12 - f12e) - 0.5) \*\* 2 / f12e + \  
 (abs(f21 - f21e) - 0.5) \*\* 2 / f21e + (abs(f22 - f22e) - 0.5) \*\* 2 / f22e  
 *if* chi\_square > 1:  
 writer.writerow({'Word': word, 'Chi-square': chi\_square, 'In Sub': f11, 'In Obj': f21})

1. File name: Test.py

This file contains codes for tweet collection.

To collect certain number of tweet from a specific Twitter user by indicating user ID, this method returns the most recent 200 tweets sent by this user.

*for* status *in* tweepy.Cursor(api.user\_timeline, *user\_id*='2569997099',  
 *exclude\_replies*='false', *include\_rts*='false').items(200):

1. File name: Tweet.py

This file contains methods for tweet cleansing, micro-phrase splitting and polarity score calculation.

**Cleanse method:** tweet.cleanse(“text”)

Parameter: tweet as string

Return type: 4-element tuple: first element: tokenized word in list; second element: extracted emojis as string; third element: 0 if no all-cap-words, 1 if exists all-cap-words; fourth element: 0 if no lengthening words, 1 if exist lengthening words

**Match emoji method:** tweet.match-emj(“text”)

Parameter: string

Return type: separated emoji string

**Split micro-phrase method:** tweet.micro([tagged\_tokenized\_list])

Parameter: tokenized and tagged word list

Return type: list of micro-phrases

**Calculate overall polarity:** tweet.get\_polarity(cleansed\_tweet, tagger, [strong\_pos\_list], [strong\_neg\_list] , [weak\_pos\_list], [weak\_neg\_list] , [emj\_pos\_list], [emj\_neg\_list])

Parameter: tuple of cleansed tweet, POS tagger, strong positive word list, strong negative word list, weak positive word list, weak negative word list, strong positive emoji list, strong negative emoji list

Return type: polarity score and subjectivity score as integer