A new ANEW: Evaluation of a word list for sentiment analysis in microblogs

Finn Årup Nielsen

DTU Informatics, Technical University of Denmark, Lyngby, Denmark.
fn@imm.dtu.dk, http://www.imm.dtu.dk/~fn/

Abstract. Sentiment analysis of microblogs such as Twitter has recently gained a fair amount of attention. One of the simplest sentiment analysis approaches compares the words of a posting against a labeled word list, where each word has been scored for valence, — a "sentiment lexicon" or "affective word lists". There exist several affective word lists, e.g., ANEW (Affective Norms for English Words) developed before the advent of microblogging and sentiment analysis. I wanted to examine how well ANEW and other word lists performs for the detection of sentiment strength in microblog posts in comparison with a new word list specifically constructed for microblogs. I used manually labeled postings from Twitter scored for sentiment. Using a simple word matching I show that the new word list may perform better than ANEW, though not as good as the more elaborate approach found in SentiStrength.

1 Introduction

Sentiment analysis has become popular in recent years. Web services, such as socialmention.com, may even score microblog postings on Identi.ca and Twitter for sentiment in real-time. One approach to sentiment analysis starts with labeled texts and uses supervised machine learning trained on the labeled text data to classify the polarity of new texts [1]. Another approach creates a sentiment lexicon and scores the text based on some function that describes how the words and phrases of the text matches the lexicon. This approach is, e.g., at the core of the *SentiStrength* algorithm [2].

It is unclear how the best way is to build a sentiment lexicon. There exist several word lists labeled with emotional valence, e.g., ANEW [3], General Inquirer, OpinionFinder [4], SentiWordNet and WordNet-Affect as well as the word list included in the SentiStrength software [2]. These word lists differ by the words they include, e.g., some do not include strong obscene words and Internet slang acronyms, such as "WTF" and "LOL". The inclusion of such terms could be important for reaching good performance when working with short informal text found in Internet fora and microblogs. Word lists may also differ in whether the words are scored with sentiment strength or just positive/negative polarity.

I have begun to construct a new word list with sentiment strength and the inclusion of Internet slang and obscene words. Although we have used it for sentiment analysis on Twitter data [5] we have not yet validated it. Data sets with manually labeled texts can evaluate the performance of the different sentiment analysis methods. Researchers increasingly use Amazon Mechanical Turk (AMT) for creating labeled language data, see, e.g., [6]. Here I take advantage of this approach.

2 Construction of word list

My new word list was initially set up in 2009 for tweets downloaded for on-line sentiment analysis in relation to the United Nation Climate Conference (COP15). Since then it has been extended. The version termed AFINN-96 distributed on the Internet[1] has 1468 different words, including a few phrases. The newest version has 2477 unique words, including 15 phrases that were not used for this study. As SentiStrength[2] it uses a scoring range from −5 (very negative) to +5 (very positive). For ease of labeling I only scored for valence, leaving out, e.g., subjectivity/objectivity, arousal and dominance. The words were scored manually by the author.

The word list initiated from a set of obscene words [7, 8] as well as a few positive words. It was gradually extended by examining Twitter postings collected for COP15 particularly the postings which scored high on sentiment using the list as it grew. I included words from the public domain *Original Balanced Affective Word List*[3] by Greg Siegle. Later I added Internet slang by browsing the Urban Dictionary[4] including acronyms such as WTF, LOL and ROFL. The most recent additions come from the large word list by Steven J. DeRose, *The Compass DeRose Guide to Emotion Words*.[5] The words of DeRose are categorized but not scored for valence with numerical values. Together with the DeRose words I browsed Wiktionary and the synonyms it provided to further enhance the list. In some cases I used Twitter to determine in which contexts the word appeared. I also used the Microsoft Web n-gram similarity Web service ("Clustering words based on context similarity"[6]) to discover relevant words. I do not distinguish between word categories so to avoid ambiguities I excluded words such as patient, firm, mean, power and frank. Words such as "surprise"—with high arousal but with variable sentiment—were not included in the word list.

Most of the positive words were labeled with +2 and most of the negative words with –2, see the histogram in Figure 1. I typically rated strong obscene words, e.g., as listed in [7], with either –4 or –5. The word list have a bias towards negative words (1598, corresponding to 65%) compared to positive words (878). A single phrase was labeled with valence 0. The bias corresponds closely to the bias found in the OpinionFinder sentiment lexicon (4911 (64%) negative and 2718 positive words).

[1] http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=59819
[2] http://sentistrength.wlv.ac.uk/
[3] http://www.sci.sdsu.edu/CAL/wordlist/origwordlist.html
[4] http://www.urbandictionary.com
[5] http://www.derose.net/steve/resources/emotionwords/ewords.html
[6] http://web-ngram.research.microsoft.com/similarity/

I compared the score of each word with mean valence of ANEW. Figure 2 shows a scatter plot for this comparison yielding a Spearman's rank correlation on 0.81 when words are directly matched and including words only in the intersection of the two word lists. I also tried to match entries in ANEW and my word list by applying Porter word stemming (on both word lists) and WordNet lemmatization (on my word list) as implemented in NLTK [9]. The results did not change significantly.

When splitting the ANEW at valence 5 and my list at valence 0 I find a few discrepancies: aggressive, mischief, ennui, hard, silly, alert, mischiefs, noisy. Word stemming generates a few further discrepancies, e.g., alien/alienation, affection/affected, profit/profiteer.

Apart from ANEW I also examined General Inquirer and the OpinionFinder word lists. As these word lists report polarity I associated words with positive sentiment with the valence +1 and negative with −1. I furthermore obtained the sentiment strength from SentiStrength via its Web service[7] and converted its positive and negative sentiments to one single value by selecting the one with the numerical largest value and zeroing the sentiment if the positive and negative sentiment magnitudes were equal.

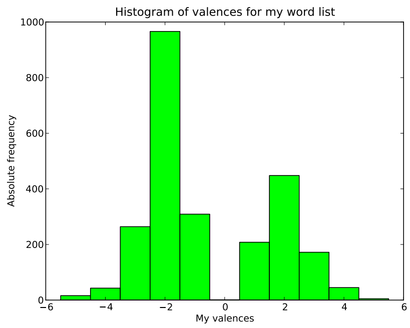

Fig. 1. Histogram of my valences.

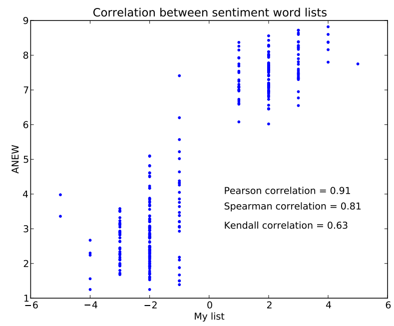

Fig. 2. Correlation between ANEW and my new word list.

3 Twitter data

For evaluating and comparing the word list with ANEW, General Inquirer, OpinionFinder and SentiStrength a data set of 1,000 tweets labeled with AMT was applied. These labeled tweets were collected by Alan Mislove for the *Twittermood*/"Pulse of a Nation"[8] study [10]. Each tweet was rated ten times to get a more reliable estimate of the human-perceived mood, and each rating was a sentiment strength with an integer between 1 (negative) and 9 (positive). The average over the ten values represented the canonical "ground truth" for this study. The tweets were not used during the construction of the word list.

To compute a sentiment score of a tweet I identified words and found the va-

[7] http://sentistrength.wlv.ac.uk/
[8] http://www.ccs.neu.edu/home/amislove/twittermood/

Table 1. Example tweet scoring. –5 has been subtracted from the original ANEW score. SentiStrength reported “positive strength 1 and negative strength –2”.

Words:	ear	infection	making	it	impossible	2	sleep	headed	2	the	doctors	2	get	new	prescription	so	fucking	early	
My	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-4	0	-4
ANEW	0	-3.34	0	0	0	0	2.2	0	0	0	0	0	0	0	0	0	0	0	-1.14
GI	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
OF	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
SS																			-2

lence for each word by lookup in the sentiment lexicons. The sum of the valences of the words divided by the number of words represented the combined sentiment strength for a tweet. I also tried a few other weighting schemes: The sum of valence without normalization of words, normalizing the sum with the number of words with non-zero valence, choosing the most extreme valence among the words and quantisizing the tweet valences to +1, 0 and −1. For ANEW I also applied a version with match using the NLTK WordNet lemmatizer.

4 Results

My word tokenization identified 15,768 words in total among the 1,000 tweets with 4,095 unique words. 422 of these 4,095 words hit my 2,477 word sized list, while the corresponding number for ANEW was 398 of its 1034 words. Of the 3392 words in General Inquirer I labeled with non-zero sentiment 358 were found in our Twitter corpus and for OpinionFinder this number was 562 from a total of 6442, see Table 1 for a scored example tweet.

I found my list to have a higher correlation (Pearson correlation: 0.564, Spearman's rank correlation: 0.596, see the scatter plot in Figure 3) with the labeling from the AMT than ANEW had (Pearson: 0.525, Spearman: 0.544). In my application of the General Inquirer word list it did not perform well having a considerable lower AMT correlation than my list and ANEW (Pearson: 0.374, Spearman: 0.422). OpinionFinder with its 90% larger lexicon performed better than General Inquirer but not as good as my list and

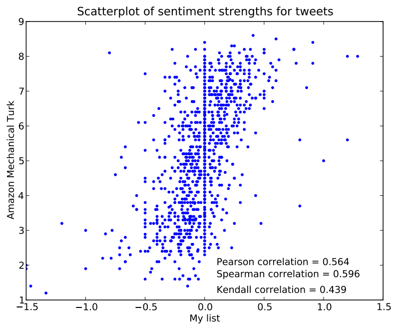

Fig. 3. Scatter plot of sentiment strengths for 1,000 tweets with AMT sentiment plotted against sentiment found by application or my word list.

	My	ANEW	GI	OF	SS
AMT	.564	.525	.374	.458	.610
My		.696	.525	.675	.604
ANEW			.592	.624	.546
GI				.705	.474
OF					.512

Table 2. Pearson correlations between sentiment strength detections methods on 1,000 tweets. AMT: Amazon Mechanical Turk, GI: General Inquirer, OF: OpinionFinder, SS: SentiStrength.

ANEW (Pearson: 0.458, Spearman: 0.491). The SentiStrength analyzer showed superior performance with a Pearson correlation on 0.610 and Spearman on 0.616, see Table 2.

I saw little effect of the different tweet sentiment scoring approaches: For ANEW 4 different Pearson correlations were in the range 0.522–0.526. For my list I observed correlations in the range 0.543–0.581 with the extreme scoring as the lowest and sum scoring without normalization the highest. With quantization of the tweet scores to +1, 0 and –1 the correlation only dropped to 0.548. For the Spearman correlation the sum scoring with normalization for the number of words appeared as the one with the highest value (0.596).

To examine whether the difference in performance between the application of ANEW and my list is due to a different lexicon or a different scoring I looked on the intersection between the two word lists. With a direct match this intersection consisted of 299 words. Building two new sentiment lexicons with these 299 words, one with the valences from my list, the other with valences from ANEW, and applying them on the Twitter data I found that the Pearson correlations were 0.49 and 0.52 to ANEW's advantage.

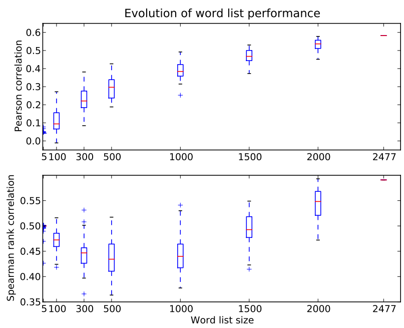

Fig. 4. Performance growth with word list extension from 5 words 2477 words. Upper panel: Pearson, lower: Spearman rank correlation, generated from 50 resamples among the 2477 words.

5 Discussion

On the simple word list approach for sentiment analysis I found my list performing slightly ahead of ANEW. However the more elaborate sentiment analysis in SentiStrength showed the overall best performance with a correlation to AMT labels on 0.610. This figure is close to the correlations reported in the evaluation of the SentiStrength algorithm on 1,041 MySpace comments (0.60 and 0.56) [2].

Even though General Inquirer and OpinionFinder have the largest word lists I found I could not make them perform as good as SentiStrength, my list and ANEW for sentiment strength detection in microblog posting. The two former lists both score words on polarity rather than strength and it could explain the difference in performance.

Is the difference between my list and ANEW due to better scoring or more words? The analysis of the intersection between the two word list indicated that the ANEW scoring is better. The slightly better performance of my list with the entire lexicon may be due to its inclusion of Internet slang and obscene words.

Newer methods, e.g., as implemented in SentiStrength, use a range of techniques: detection of negation, handling of emoticons and spelling variations [2]. The present application of my list used none of these approaches and might have benefited. However, the SentiStrength evaluation showed that valence switching at negation and emoticon detection might not necessarily increase the performance of sentiment analyzers (Tables 4 and 5 in [2]).

The evolution of the performance (Figure 4) suggests that the addition of words to my list might still improve its performance slightly.

Although my list comes slightly ahead of ANEW in Twitter sentiment analysis, ANEW is still preferable for scientific psycholinguistic studies as the scoring has been validated across several persons. Also note that ANEW's standard deviation was not used in the scoring. It might have improved its performance.

Acknowledgment I am grateful to Alan Mislove and Sune Lehmann for providing the 1,000 tweets with the Amazon Mechanical Turk labels and to Steven J. DeRose and Greg Siegle for providing their word lists. Mislove, Lehmann and Daniela Balslev also provided input to the article. I thank the Danish Strategic Research Councils for generous support to the 'Responsible Business in the Blogosphere' project.

References

1. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval **2**(1-2) (2008) 1–135
2. Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., Kappas, A.: Sentiment strength detection in short informal text. Journal of the American Society for Information Science and Technology **61**(12) (2010) 2544–2558
3. Bradley, M.M., Lang, P.J.: Affective norms for English words (ANEW): Instruction manual and affective ratings. Technical Report C-1, The Center for Research in Psychophysiology, University of Florida (1999)
4. Wilson, T., Wiebe, J., Hoffmann, P.: Recognizing contextual polarity in phrase-level sentiment analysis. In: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, Stroudsburg, PA, USA, Association for Computational Linguistics (2005)
5. Hansen, L.K., Arvidsson, A., Nielsen, F.Å., Colleoni, E., Etter, M.: Good friends, bad news — affect and virality in Twitter. Accepted for The 2011 International Workshop on Social Computing, Network, and Services (SocialComNet 2011) (2011)
6. Akkaya, C., Conrad, A., Wiebe, J., Mihalcea, R.: Amazon Mechanical Turk for subjectivity word sense disambiguation. In: Proceedings of the NAACL HLT 2010 Workshop on Creating, Speech and Language Data with Amazon's Mechanical Turk, Association for Computational Linguistics (2010) 195–203
7. Baudhuin, E.S.: Obscene language and evaluative response: an empirical study. Psychological Reports **32** (1973)
8. Sapolsky, B.S., Shafer, D.M., Kaye, B.K.: Rating offensive words in three television program contexts. BEA 2008, Research Division (2008)
9. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python. O'Reilly, Sebastopol, California (June 2009)
10. Biever, C.: Twitter mood maps reveal emotional states of America. The New Scientist **207**(2771) (July 2010) 14

Supplementary material

A Extra results

AMT	Mislove	ANEW	AFINN-96	AFINN	AFINN(q)	AFINN(s)	AFINN(a)	ANEW(r)	ANEW(l)	ANEW(rs)	ANEW(a)
		0.506	0.546	0.564	0.548	0.581	0.564	0.526	0.527	0.526	0.523
			0.564	0.578	0.588	0.583	0.616	0.780	0.760	0.855	0.950
				0.970	0.734	0.846	0.802	0.660	0.643	0.580	0.582
					0.754	0.870	0.824	0.696	0.682	0.599	0.598
						0.836	0.889	0.531	0.528	0.592	0.606
							0.884	0.547	0.536	0.628	0.594
								0.572	0.563	0.606	0.620
									0.973	0.853	0.819
										0.829	0.796
											0.893

Table 3. Pearson correlation matrix. The first data row is the correlation for Amazon Mechanical Turk (AMT) labeling. AFINN is my list. AFINN: sum of word valences with weighting over number of words in text, AFINN(q): $(+1, 0, -1)$-quantization, AFINN(s): sum of word valences, AFINN(a): average of non-zero word valences, AFINN(x): extreme scoring, ANEW: ANEW with "raw" (direct match) and sum of word valences weighted by number of words, ANEW: with NLTK WordNet lemmatizer word match, ANEW(rs): raw match with sum of word valences, ANEW(a): average of non-zero word valences, GI: General Inquirer, OF: OpinionFinder, SS: SentiStrength.

AMT	Mislove	ANEW	AFINN-96	AFINN	AFINN(q)	AFINN(s)	AFINN(a)	ANEW(r)	ANEW(l)	ANEW(rs)	ANEW(a)
		0.507	0.592	0.596	0.580	0.591	0.581	0.545	0.548	0.537	0.526
			0.630	0.635	0.615	0.625	0.635	0.884	0.857	0.895	0.950
				0.970	0.912	0.941	0.925	0.656	0.646	0.650	0.647
					0.941	0.969	0.954	0.668	0.657	0.661	0.656
						0.944	0.933	0.622	0.620	0.644	0.634
							0.959	0.630	0.622	0.662	0.644
								0.641	0.635	0.651	0.644
									0.972	0.946	0.936
										0.918	0.905
											0.943

Table 4. Spearman rank correlation matrix. For explanation of columns see Table 3.

Listings

```python
#!/usr/bin/env python
#
# This program go through the list a captures double entries
# and ordering problems
#
# $Id: Nielsen2011New_check.py,v 1.4 2011/03/16 11:34:24 fn Exp $


filebase = '/home/fnielsen/'
filename_afinn = filebase + 'fnielsen/data/Nielsen2009Responsible_emotion.csv'

lines = [ line.split('\t') for line in open(filename_afinn) ]
for line in lines:
    if len(line) != 2:
        print(line)

afinn = map(lambda (k,v): (k,int(v)),
                    [ line.split('\t') for line in open(filename_afinn) ])
words = [ word for word, valence in afinn ]


swords = sorted(list(set(words)))

for n in range(len(words)):
    if words[n] != swords[n]:
        print(words[n] + " " + swords[n])
        break
```

```python
#!/usr/bin/env python
#
#    Construct histogram of AFINN valences.
#
# $Id: Nielsen2011New_hist.py,v 1.2 2011/03/14 20:53:17 fn Exp $

import numpy as np
import pylab
import sys
reload(sys)
sys.setdefaultencoding('utf-8')

filebase = '/home/fnielsen/'
filename = filebase + 'fnielsen/data/Nielsen2009Responsible_emotion.csv'
afinn = dict(map(lambda (k,v): (k,int(v)),
                     [ line.split('\t') for line in open(filename) ]))

pylab.hist(afinn.values(), bins=11, range=(-5.5, 5.5), facecolor=(0,1,0))
pylab.xlabel('My valences')
pylab.ylabel('Absolute frequency')
pylab.title('Histogram of valences for my word list')

pylab.show()

# pylab.savefig(filebase + 'fnielsen/eps/' + 'Nielsen2011New_hist.eps')
```

```python
#!/usr/bin/env python
#
# $Id: Nielsen2011New_example.py,v 1.2 2011/04/11 17:31:22 fn Exp $

import csv
```

```
import re
import numpy as np
from nltk.stem.wordnet import WordNetLemmatizer
from nltk import sent_tokenize
import pylab
from scipy.stats.stats import  kendalltau, spearmanr
import simplejson

# This variable determines the data set
mislove = 2

# Filenames
filebase = '/home/fn/'
filename_afinn = filebase + 'fnielsen/data/Nielsen2009Responsible_emotion.csv'
filename_afinn96 = 'AFINN-96.txt'
filename_mislove1 = "results.txt"
filename_mislove2a = "turk.txt"
filename_mislove2b = "turk-results.txt"
filename_anew = filebase + 'data/ANEW.TXT'
filename_gi = filebase + "data/inqtabs.txt"
filename_of = filebase + "data/subjclueslen1-HLTEMNLP05.tff"

# Word splitter pattern
pattern_split = re.compile(r"\W+")


afinn = dict(map(lambda (k,v): (k,int(v)),
                     [ line.split('\t') for line in open(filename_afinn) ]))


afinn96 = dict(map(lambda (k,v): (k,int(v)),
                     [ line.split('\t') for line in open(filename_afinn96) ]))


# ANEW
anew = dict(map(lambda l: (l[0], float(l[2]) - 5) ,
                    [ re.split('\s+', line) for line in open(filename_anew).readlines()[41:1075] ]))


# OpinionFinder
of = {}
for line in open(filename_of).readlines():
    elements = re.split('(\s|\=)', line)
    if elements[22] == 'positive':
        of[elements[10]] = +1
    elif elements[22] == 'negative':
        of[elements[10]] = -1


# General inquirer
csv_reader = csv.reader(open(filename_gi), delimiter='\t')
header = []
gi = {}
previousword = []
previousvalence = []
for row in csv_reader:
    if not header:
        header = row
    elif len(row) > 2:
        word = re.search("\w+", row[0].lower()).group()
        if row[2] == "Positiv":
            valence = 1
        elif row[3] == "Negativ":
            valence = -1
        else:
            valence = 0
        if re.search("#", row[0].lower()):
            if previousword == word:
```

```
                if previousvalence == []:
                    previousvalence = valence
                elif previousvalence != valence:
                    previousvalence = 0
            else:
                if previousvalence:
                    gi[previousword] = previousvalence
                previousword = word
                previousvalence = []
        elif valence:
            gi[word] = valence



# Lemmatizer for WordNet
lemmatizer = WordNetLemmatizer()



def words2anewsentiment(words):
    """
    Convert words to sentiment based on ANEW via WordNet Lemmatizer
    """
    sentiment = 0
    for word in words:
        if word in anew:
            sentiment += anew[word]
            continue
        lword = lemmatizer.lemmatize(word)
        if lword in anew:
            sentiment += anew[lword]
            continue
        lword = lemmatizer.lemmatize(word, pos='v')
        if lword in anew:
            sentiment += anew[lword]
    return sentiment/len(words)


def extremevalence(valences):
    """
    Return the most extreme valence. If extremes have different sign then
    zero is returned
    """
    imax = np.argsort(np.abs(valences))[-1]
    extremes = filter(lambda v: abs(v) == abs(valences[imax]), valences)
    extremes_samesign = filter(lambda v: v == valences[imax], valences)
    if extremes == extremes_samesign:
        return valences[imax]
    else:
        return 0

def corrmatrix2latex(C, columns=None):
    """
    s = corrmatrix2latex(C)
    print(s)
    """
    s = '\n\\begin{tabular}{' + ('r'*(C.shape[0]-1)) + '}\n'
    if columns:
        s += " & ".join(columns[1:]) + " \\\\\n\\hline\n"
    for n in range(C.shape[0]):
        row = []
        for m in range(1, C.shape[1]):
            if m > n:
                row.append("%.3f" % C[n,m])
            else:
                row.append("     ")
        s += " & ".join(row) + ' \\\\\n'
    s += '\\end{tabular}\n'
```

```python
    return s


def spearmanmatrix(data):
    """
    Spearman r rank correlation matrix
    """
    C = np.zeros((data.shape[1], data.shape[1]))
    for n in range(data.shape[1]):
        for m in range(data.shape[1]):
            C[n,m] = spearmanr(data[:,n], data[:,m])[0]
    return C


def kendallmatrix(data):
    """
    Kendall tau rank correlation matrix
    """
    C = np.zeros((data.shape[1], data.shape[1]))
    for n in range(data.shape[1]):
        for m in range(data.shape[1]):
            C[n,m] = kendalltau(data[:,n], data[:,m])[0]
    return C



# Read Mislove CSV Twitter data: 'tweets' an array of dictionaries
if mislove == 1:
    csv_reader = csv.reader(open(filename_mislove1), delimiter='\t')
    header = []
    tweets = []
    for row in csv_reader:
        if not header:
            header = row
        else:
            tweets.append({'id': row[0],
                           'quant': int(row[1]),
                           'score_our': float(row[2]),
                           'score_mean': float(row[3]),
                           'score_std': float(row[4]),
                           'text': row[5],
                           'scores': map(int, row[6:])})
elif mislove == 2:
    if False:
        tweets = simplejson.load(open('tweets_with_sentistrength.json'))
    else:
        csv_reader = csv.reader(open(filename_mislove2a), delimiter='\t')
        tweets = []
        for row in csv_reader:
            tweets.append({'id': row[2],
                           'score_mislove2': float(row[1]),
                           'text': row[3]})
        csv_reader = csv.reader(open(filename_mislove2b), delimiter='\t')
        tweets_dict = {}
        header = []
        for row in csv_reader:
            if not header:
                header = row
            else:
                tweets_dict[row[0]] = {'id': row[0],
                                       'score_mislove': float(row[1]),
                                       'score_amt_wrong': float(row[2]),
                                       'score_amt': np.mean(map(int, re.split("\s+", " ".join(row
                                       }
        for n in range(len(tweets)):
            tweets[n]['score_mislove'] = tweets_dict[tweets[n]['id']]['score_mislove']
            tweets[n]['score_amt_wrong'] = tweets_dict[tweets[n]['id']]['score_amt_wrong']
            tweets[n]['score_amt'] = tweets_dict[tweets[n]['id']]['score_amt']
```

```


# Add sentiments to 'tweets'
for n in range(len(tweets)):
    words = pattern_split.split(tweets[n]['text'].lower())
    tweets[n]['words'] = words
    afinn_sentiments = map(lambda word: afinn.get(word, 0), words)
    afinn_sentiment = float(sum(afinn_sentiments))/len(afinn_sentiments)
    afinn96_sentiments = map(lambda word: afinn96.get(word, 0), words)
    afinn96_sentiment = float(sum(afinn96_sentiments))/len(afinn96_sentiments)
    anew_sentiments = map(lambda word: anew.get(word, 0), words)
    anew_sentiment = float(sum(anew_sentiments))/len(anew_sentiments)
    gi_sentiments = map(lambda word: gi.get(word, 0), words)
    gi_sentiment = float(sum(gi_sentiments))/len(gi_sentiments)
    of_sentiments = map(lambda word: of.get(word, 0), words)
    of_sentiment = float(sum(of_sentiments))/len(of_sentiments)
    tweets[n]['sentiment_afinn96'] = afinn96_sentiment
    tweets[n]['sentiment_afinn'] = afinn_sentiment
    tweets[n]['sentiment_afinn_quant'] = np.sign(afinn_sentiment)
    tweets[n]['sentiment_afinn_sum'] = sum(afinn_sentiments)
    nonzeros = len(filter(lambda nonzero: nonzero, afinn_sentiments))
    if not nonzeros: nonzeros = 1
    tweets[n]['sentiment_afinn_nonzero'] = sum(afinn_sentiments)/nonzeros
    tweets[n]['sentiment_afinn_extreme'] = extremevalence(afinn_sentiments)
    tweets[n]['sentiment_anew_raw'] = anew_sentiment
    tweets[n]['sentiment_anew_lemmatize'] = words2anewsentiment(words)
    tweets[n]['sentiment_anew_raw_sum'] = sum(anew_sentiments)
    nonzeros = len(filter(lambda nonzero: nonzero, anew_sentiments))
    if not nonzeros: nonzeros = 1
    tweets[n]['sentiment_anew_raw_nonzeros'] = sum(anew_sentiments)/nonzeros
    tweets[n]['sentiment_gi_raw'] = gi_sentiment
    tweets[n]['sentiment_of_raw'] = of_sentiment


# Index for example tweet
words = tweets[index]['words'][:-1]
index = 10
s = ""
s += "Text: & \\multicolumn{%d}{c}{" % len(words) + tweets[index]['text'] + "} \\\\ [1pt] \\hline
s += "Words: & " + " & ".join(words) + " \\\\ [1pt] \n"
s += "My & " + " & ".join([ str(afinn.get(w,0)) for w in words ]) + " & "+ str(sum([ afinn.get(w,
])) + " \\\\ [1pt] \n"
s += "ANEW & " + " & ".join([ str(anew.get(w,0)) for w in words ]) + " & "+ str(sum([ anew.get(w,
]))+ " \\\\ [1pt] \n"
s += "GI & " + " & ".join([ str(gi.get(w,0)) for w in words ]) + " & "+ str(sum([ gi.get(w,0) for
]))+ " \\\\ [1pt] \n"
s += "OF & " + " & ".join([ str(of.get(w,0)) for w in words ]) + " & "+ str(sum([ of.get(w,0) for
]))+ " \\\\ [1pt] \n"
s += "SS " + " & " * (len(words)+1) + " -1 " + "\\\\ [1pt] \hline \n"

# 'Ear infection making it impossible 2 sleep. headed 2 the doctors 2
# get new prescription. so fucking ' has positive strength 1 and negative strength -2

print(s)



score_amt = np.asarray([ t['score_amt'] for t in tweets ])
score_afinn = np.asarray([ t['sentiment_afinn'] for t in tweets ])

t = """
                    AMT
              positive neural negative
AFINN positive  %d       %d      %d
      neutral   %d       %d      %d
      negative  %d       %d      %d
""" % (sum((1*(score_amt>5)) * (1*(score_afinn>0))),
       sum((1*(score_amt==5)) * (1*(score_afinn>0))),
```

```
      sum((1*(score_amt<5)) * (1*(score_afinn>0))),
      sum((1*(score_amt>5)) * (1*(score_afinn==0))),
      sum((1*(score_amt==5)) * (1*(score_afinn==0))),
      sum((1*(score_amt<5)) * (1*(score_afinn==0))),
      sum((1*(score_amt>5)) * (1*(score_afinn<0))),
      sum((1*(score_amt==5)) * (1*(score_afinn<0))),
      sum((1*(score_amt<5)) * (1*(score_afinn<0))))

print(t)

# 0.1*(277+299+5)
```

```
#!/usr/bin/env python
#
#    The program compares AFINN and ANEW word lists.
#
#    (Copied from Hansen2010Diffusion and extended.)
#
# $Id: Hansen2010Diffusion_anew.py,v 1.3 2010/12/15 15:50:39 fn Exp $

from nltk.stem.wordnet import WordNetLemmatizer
import nltk
import numpy as np
import pylab
import re
from scipy.stats.stats import  kendalltau, spearmanr
import sys
reload(sys)
sys.setdefaultencoding('utf-8')


filebase = '/home/fn/'
filename = filebase + 'fnielsen/data/Nielsen2009Responsible_emotion.csv'
afinn = dict(map(lambda (k,v): (k,int(v)),
                  [ line.split('\t') for line in open(filename) ]))

filename = filebase + 'data/ANEW.TXT'
anew = dict(map(lambda l: (l[0], float(l[2])) ,
                [ re.split('\s+', line) for line in open(filename).readlines()[41:1075] ]))

lemmatizer = WordNetLemmatizer()
stemmer = nltk.PorterStemmer()

anew_stem = dict([ (stemmer.stem(word), valence) for word, valence in anew.items() ])



def word2anewsentiment_raw(word):
    return anew.get(word, None)


def word2anewsentiment_wordnet(word):
    sentiment = None
    if word in anew:
        sentiment = anew[word]
    else:
        lword = lemmatizer.lemmatize(word)
        if lword in anew:
            sentiment = anew[lword]
        else:
            lword = lemmatizer.lemmatize(word, pos='v')
            if lword in anew:
                sentiment = anew[lword]
    return sentiment


def word2anewsentiment_stem(word):
```

```python
    return anew_stem.get(stemmer.stem(word), None)



sentiments_raw = []
for word in afinn.keys():
    sentiment_anew = word2anewsentiment_raw(word)
    if sentiment_anew:
        sentiments_raw.append((afinn[word], sentiment_anew))


sentiments_wordnet = []
for word in afinn.keys():
    sentiment_anew = word2anewsentiment_wordnet(word)
    if sentiment_anew:
        sentiments_wordnet.append((afinn[word], sentiment_anew))
        if (afinn[word] > 0 and sentiment_anew < 5) or \
                (afinn[word] < 0 and sentiment_anew > 5):
            print(word)


sentiments_stem = []
for word in afinn.keys():
    sentiment_stem_anew = word2anewsentiment_stem(word)
    if sentiment_stem_anew:
        sentiments_stem.append((afinn[word], sentiment_stem_anew))
        if (afinn[word] > 0 and sentiment_stem_anew < 5) or \
                (afinn[word] < 0 and sentiment_stem_anew > 5):
            print(word)


sentiments_raw = np.asarray(sentiments_raw)
pylab.figure(1)
pylab.plot(sentiments_raw[:,0], sentiments_raw[:,1], '.')
pylab.xlabel('Our list')
pylab.ylabel('ANEW')
pylab.title('Correlation between sentiment word lists (Direct match)')
pylab.text(1, 3, "Pearson correlation = %.2f" % np.corrcoef(sentiments_raw.T)[1,0])
pylab.text(1, 2.5, "Spearman correlation = %.2f" % spearmanr(sentiments_raw[:,0], sentiments_raw[
pylab.text(1, 2, "Kendall correlation = %.2f" % kendalltau(sentiments_raw[:,0], sentiments_raw[:,
# pylab.savefig(filebase + 'fnielsen/eps/' + 'Nielsen2011New_anew_raw.eps')

# pylab.show()



sentiments = np.asarray(sentiments_wordnet)
pylab.figure(2)
pylab.plot(sentiments[:,0], sentiments[:,1], '.')
pylab.xlabel('Our list')
pylab.ylabel('ANEW')
pylab.title('Correlation between sentiment word lists (WordNet lemmatizer)')
pylab.text(1, 3, "Pearson correlation = %.2f" % np.corrcoef(sentiments.T)[1,0])
pylab.text(1, 2.5, "Spearman correlation = %.2f" % spearmanr(sentiments[:,0], sentiments[:,1])[0]
pylab.text(1, 2, "Kendall correlation = %.2f" % kendalltau(sentiments[:,0], sentiments[:,1])[0])
# pylab.savefig(filebase + 'fnielsen/eps/' + 'Nielsen2011New_anew_wordnet.eps')

# pylab.show()



sentiments_stem = np.asarray(sentiments_stem)
pylab.figure(3)
pylab.plot(sentiments_stem[:,0], sentiments_stem[:,1], '.')
pylab.xlabel('My list')
pylab.ylabel('ANEW')
pylab.title('Correlation between sentiment word lists (Porter stemmer)')
pylab.text(1, 3, "Correlation = %.2f" % np.corrcoef(sentiments_stem.T)[1,0])
```

```
pylab.text(1, 2.5, "Spearman correlation = %.2f" % spearmanr(sentiments_stem[:,0], sentiments_ste
pylab.text(1, 2, "Kendall correlation = %.2f" % kendalltau(sentiments_stem[:,0], sentiments_stem[
# pylab.savefig(filebase + 'fnielsen/eps/' + 'Nielsen2011New_anew_stem.eps')

# pylab.show()
```

```python
#!/usr/bin/env python
#
# $Id: Nielsen2011New.py,v 1.10 2011/03/16 13:41:36 fn Exp $

import csv
import re
import numpy as np
from nltk.stem.wordnet import WordNetLemmatizer
from nltk import sent_tokenize
import pylab
from scipy.stats.stats import  kendalltau, spearmanr
import simplejson

# This variable determines the data set
mislove = 2

# Filenames
filebase = '/home/fnielsen/'
filename_afinn = filebase + 'fnielsen/data/Nielsen2009Responsible_emotion.csv'
filename_afinn96 = 'AFINN-96.txt'
filename_mislove1 = "results.txt"
filename_mislove2a = "turk.txt"
filename_mislove2b = "turk-results.txt"
filename_anew = filebase + 'data/ANEW.TXT'
filename_gi = filebase + "data/inqtabs.txt"
filename_of = filebase + "data/subjclueslen1-HLTEMNLP05.tff"

# Word splitter pattern
pattern_split = re.compile(r"\W+")


afinn = dict(map(lambda (k,v): (k,int(v)),
                 [ line.split('\t') for line in open(filename_afinn) ]))


afinn96 = dict(map(lambda (k,v): (k,int(v)),
                 [ line.split('\t') for line in open(filename_afinn96) ]))


# ANEW
anew = dict(map(lambda l: (l[0], float(l[2]) - 5) ,
                [ re.split('\s+', line) for line in open(filename_anew).readlines()[41:1075] ]))


# OpinionFinder
of = {}
for line in open(filename_of).readlines():
    elements = re.split('(\s|\=)', line)
    if elements[22] == 'positive':
        of[elements[10]] = +1
    elif elements[22] == 'negative':
        of[elements[10]] = -1


# General inquirer
csv_reader = csv.reader(open(filename_gi), delimiter='\t')
header = []
gi = {}
previousword = []
previousvalence = []
for row in csv_reader:
```

```python
    if not header:
        header = row
    elif len(row) > 2:
        word = re.search("\w+", row[0].lower()).group()
        if row[2] == "Positiv":
            valence = 1
        elif row[3] == "Negativ":
            valence = -1
        else:
            valence = 0
        if re.search("#", row[0].lower()):
            if previousword == word:
                if previousvalence == []:
                    previousvalence = valence
                elif previousvalence != valence:
                    previousvalence = 0
            else:
                if previousvalence:
                    gi[previousword] = previousvalence
                previousword = word
                previousvalence = []
        elif valence:
            gi[word] = valence



# Lemmatizer for WordNet
lemmatizer = WordNetLemmatizer()



def words2anewsentiment(words):
    """
    Convert words to sentiment based on ANEW via WordNet Lemmatizer
    """
    sentiment = 0
    for word in words:
        if word in anew:
            sentiment += anew[word]
            continue
        lword = lemmatizer.lemmatize(word)
        if lword in anew:
            sentiment += anew[lword]
            continue
        lword = lemmatizer.lemmatize(word, pos='v')
        if lword in anew:
            sentiment += anew[lword]
    return sentiment/len(words)


def extremevalence(valences):
    """
    Return the most extreme valence. If extremes have different sign then
    zero is returned
    """
    imax = np.argsort(np.abs(valences))[-1]
    extremes = filter(lambda v: abs(v) == abs(valences[imax]), valences)
    extremes_samesign = filter(lambda v: v == valences[imax], valences)
    if extremes == extremes_samesign:
        return valences[imax]
    else:
        return 0

def corrmatrix2latex(C, columns=None):
    """
    s = corrmatrix2latex(C)
    print(s)
    """
```

```
    s = '\n\\begin{tabular}{' + ('r'*(C.shape[0]-1)) + '}\n'
    if columns:
        s += " & ".join(columns[1:]) + " \\\\\n\\hline\n"
    for n in range(C.shape[0]):
        row = []
        for m in range(1, C.shape[1]):
            if m > n:
                row.append("%.3f" % C[n,m])
            else:
                row.append("     ")
        s += " & ".join(row) + ' \\\\\n'
    s += '\\end{tabular}\n'
    return s


def spearmanmatrix(data):
    """
    Spearman r rank correlation matrix
    """
    C = np.zeros((data.shape[1], data.shape[1]))
    for n in range(data.shape[1]):
        for m in range(data.shape[1]):
            C[n,m] = spearmanr(data[:,n], data[:,m])[0]
    return C


def kendallmatrix(data):
    """
    Kendall tau rank correlation matrix
    """
    C = np.zeros((data.shape[1], data.shape[1]))
    for n in range(data.shape[1]):
        for m in range(data.shape[1]):
            C[n,m] = kendalltau(data[:,n], data[:,m])[0]
    return C



# Read Mislove CSV Twitter data: 'tweets' an array of dictionaries
if mislove == 1:
    csv_reader = csv.reader(open(filename_mislove1), delimiter='\t')
    header = []
    tweets = []
    for row in csv_reader:
        if not header:
            header = row
        else:
            tweets.append({'id': row[0],
                           'quant': int(row[1]),
                           'score_our': float(row[2]),
                           'score_mean': float(row[3]),
                           'score_std': float(row[4]),
                           'text': row[5],
                           'scores': map(int, row[6:])})
elif mislove == 2:
    if True:
        tweets = simplejson.load(open('tweets_with_sentistrength.json'))
    else:
        csv_reader = csv.reader(open(filename_mislove2a), delimiter='\t')
        tweets = []
        for row in csv_reader:
            tweets.append({'id': row[2],
                           'score_mislove2': float(row[1]),
                           'text': row[3]})
        csv_reader = csv.reader(open(filename_mislove2b), delimiter='\t')
        tweets_dict = {}
        header = []
        for row in csv_reader:
```

```
            if not header:
                header = row
            else:
                tweets_dict[row[0]] = {'id': row[0],
                                       'score_mislove': float(row[1]),
                                       'score_amt_wrong': float(row[2]),
                                       'score_amt': np.mean(map(int, re.split("\s+", " ".join(row
                                       }
        for n in range(len(tweets)):
            tweets[n]['score_mislove'] = tweets_dict[tweets[n]['id']]['score_mislove']
            tweets[n]['score_amt_wrong'] = tweets_dict[tweets[n]['id']]['score_amt_wrong']
            tweets[n]['score_amt'] = tweets_dict[tweets[n]['id']]['score_amt']


# Add sentiments to 'tweets'
for n in range(len(tweets)):
    words = pattern_split.split(tweets[n]['text'].lower())
    afinn_sentiments = map(lambda word: afinn.get(word, 0), words)
    afinn_sentiment = float(sum(afinn_sentiments))/len(afinn_sentiments)
    afinn96_sentiments = map(lambda word: afinn96.get(word, 0), words)
    afinn96_sentiment = float(sum(afinn96_sentiments))/len(afinn96_sentiments)
    anew_sentiments = map(lambda word: anew.get(word, 0), words)
    anew_sentiment = float(sum(anew_sentiments))/len(anew_sentiments)
    gi_sentiments = map(lambda word: gi.get(word, 0), words)
    gi_sentiment = float(sum(gi_sentiments))/len(gi_sentiments)
    of_sentiments = map(lambda word: of.get(word, 0), words)
    of_sentiment = float(sum(of_sentiments))/len(of_sentiments)
    tweets[n]['sentiment_afinn96'] = afinn96_sentiment
    tweets[n]['sentiment_afinn'] = afinn_sentiment
    tweets[n]['sentiment_afinn_quant'] = np.sign(afinn_sentiment)
    tweets[n]['sentiment_afinn_sum'] = sum(afinn_sentiments)
    nonzeros = len(filter(lambda nonzero: nonzero, afinn_sentiments))
    if not nonzeros: nonzeros = 1
    tweets[n]['sentiment_afinn_nonzero'] = sum(afinn_sentiments)/nonzeros
    tweets[n]['sentiment_afinn_extreme'] = extremevalence(afinn_sentiments)
    tweets[n]['sentiment_anew_raw'] = anew_sentiment
    tweets[n]['sentiment_anew_lemmatize'] = words2anewsentiment(words)
    tweets[n]['sentiment_anew_raw_sum'] = sum(anew_sentiments)
    nonzeros = len(filter(lambda nonzero: nonzero, anew_sentiments))
    if not nonzeros: nonzeros = 1
    tweets[n]['sentiment_anew_raw_nonzeros'] = sum(anew_sentiments)/nonzeros
    tweets[n]['sentiment_gi_raw'] = gi_sentiment
    tweets[n]['sentiment_of_raw'] = of_sentiment


# Numpy matrix
if mislove == 1:
    columns = ["AMT", 'AFINN', 'AFINN(q)', 'AFINN(s)', 'AFINN(a)',
               'ANEW(r)', 'ANEW(l)', 'ANEW(rs)', 'ANEW(a)']
    sentiments = np.matrix([ [t['score_mean'],
                              t['sentiment_afinn'],
                              t['sentiment_afinn_quant'],
                              t['sentiment_afinn_sum'],
                              t['sentiment_afinn_nonzero'],
                              t['sentiment_anew_raw'],
                              t['sentiment_anew_lemmatize'],
                              t['sentiment_anew_raw_sum'],
                              t['sentiment_anew_raw_nonzeros']] for t in tweets ])
elif mislove == 2:
    columns = ["AMT",
               'AFINN', 'AFINN(q)', 'AFINN(s)', 'AFINN(a)', 'AFINN(x)',
               'ANEW(r)', 'ANEW(l)', 'ANEW(rs)', 'ANEW(a)',
               "GI", "OF", "SS"]
    sentiments = np.matrix([ [t['score_amt'],
                              t['sentiment_afinn'],
                              t['sentiment_afinn_quant'],
                              t['sentiment_afinn_sum'],
                              t['sentiment_afinn_nonzero'],
```

```python
                                t['sentiment_afinn_extreme'],
                                t['sentiment_anew_raw'],
                                t['sentiment_anew_lemmatize'],
                                t['sentiment_anew_raw_sum'],
                                t['sentiment_anew_raw_nonzeros'],
                                t['sentiment_gi_raw'],
                                t['sentiment_of_raw'],
                                t['sentistrength']] for t in tweets ])


x = np.asarray(sentiments[:,1]).flatten()
y = np.asarray(sentiments[:,0]).flatten()
pylab.plot(x, y, '.')
pylab.xlabel('My list')
pylab.ylabel('Amazon Mechanical Turk')
pylab.text(0.1, 2, "Pearson correlation = %.3f" % np.corrcoef(x, y)[1,0])
pylab.text(0.1, 1.6, "Spearman correlation = %.3f" % spearmanr(x, y)[0])
pylab.text(0.1, 1.2, "Kendall correlation = %.3f" % kendalltau(x, y)[0])
pylab.title('Scatterplot of sentiment strengths for tweets')
pylab.show()
# pylab.savefig(filebase + 'fnielsen/eps/Nielsen2011New_tweetscatter.eps')

# Ordinary correlation coefficient
C = np.corrcoef(sentiments.transpose())
s1 = corrmatrix2latex(C, columns)
print(s1)

f = open(filebase + '/fnielsen/tex/Nielsen2011New_corrmatrix.tex', 'w')
f.write(s1)
f.close()


# Spearman Rank correlation
C2 = spearmanmatrix(sentiments)
s2 = corrmatrix2latex(C2, columns)
print(s2)

f = open(filebase + '/fnielsen/tex/Nielsen2011New_spearmanmatrix.tex', 'w')
f.write(s2)
f.close()


# Kendall rank correlation
C3 = kendallmatrix(sentiments)
s3 = corrmatrix2latex(C3, columns)
print(s3)

f = open(filebase + '/fnielsen/tex/Nielsen2011New_kendallmatrix.tex', 'w')
f.write(s3)
f.close()
```

```python
#!/usr/bin/env python
#
#   This script will call the SentiStrength Web service with the text from
#   the 1000 tweets and write a JSON file with tweets and the SentiStrength.
#
# $Id: Nielsen2011New_sentistrength.py,v 1.1 2011/03/13 19:12:46 fn Exp $

import csv
import re
import numpy as np
import pylab
import random
from scipy import sparse
from scipy.stats.stats import kendalltau, spearmanr
import simplejson
import sys
```

```
reload(sys)
sys.setdefaultencoding('utf-8')

from time              import sleep, time
from urllib            import FancyURLopener, urlencode, urlretrieve
from xml.sax.saxutils import unescape as unescape_saxutils


class MyOpener(FancyURLopener):
    version = 'RBBBot, Finn Aarup Nielsen (http://www.imm.dtu.dk/~fn/, fn@imm.dtu.dk)'



# This variable determines the data set
mislove = 2

# Filenames
filebase = '/home/fn/'
filename_afinn = filebase + 'fnielsen/data/Nielsen2009Responsible_emotion.csv'
filename_mislove1 = "results.txt"
filename_mislove2a = "turk.txt"
filename_mislove2b = "turk-results.txt"


urlbase = "http://sentistrength.wlv.ac.uk/results.php?"

pattern_positive = re.compile("positive strength <b>(\d)</b>")
pattern_negative = re.compile("negative strength <b>(\-\d)</b>")


# Word splitter pattern
pattern_split = re.compile(r"\W+")

myopener = MyOpener()

# Read Mislove CSV Twitter data: 'tweets' an array of dictionaries
if mislove == 1:
    csv_reader = csv.reader(open(filename_mislove1), delimiter='\t')
    header = []
    tweets = []
    for row in csv_reader:
        if not header:
            header = row
        else:
            tweets.append({'id': row[0],
                           'quant': int(row[1]),
                           'score_our': float(row[2]),
                           'score_mean': float(row[3]),
                           'score_std': float(row[4]),
                           'text': row[5],
                           'scores': map(int, row[6:])})
elif mislove == 2:
    csv_reader = csv.reader(open(filename_mislove2a), delimiter='\t')
    tweets = []
    for row in csv_reader:
        tweets.append({'id': row[2],
                       'score_mislove2': float(row[1]),
                       'text': row[3]})
    csv_reader = csv.reader(open(filename_mislove2b), delimiter='\t')
    tweets_dict = {}
    header = []
    for row in csv_reader:
        if not header:
            header = row
        else:
            tweets_dict[row[0]] = {'id': row[0],
                                   'score_mislove': float(row[1]),
                                   'score_amt_wrong': float(row[2]),
```

```python
                                    'score_amt': np.mean(map(int, re.split("\s+", " ".join(row[4:
                                    }
    for n in range(len(tweets)):
        tweets[n]['score_mislove'] = tweets_dict[tweets[n]['id']]['score_mislove']
        tweets[n]['score_amt_wrong'] = tweets_dict[tweets[n]['id']]['score_amt_wrong']
        tweets[n]['score_amt'] = tweets_dict[tweets[n]['id']]['score_amt']




for n in range(len(tweets)):
    url = urlbase + urlencode({'text': tweets[n]['text']})
    try:
        html = myopener.open(url).read()
        positive = int(pattern_positive.findall(html)[0])
        negative = int(pattern_negative.findall(html)[0])
        tweets[n]['sentistrength_positive'] = positive
        tweets[n]['sentistrength_negative'] = negative
        if positive > abs(negative):
            tweets[n]['sentistrength'] = positive
        elif abs(negative) > positive:
            tweets[n]['sentistrength'] = negative
        else:
            tweets[n]['sentistrength'] = 0
    except Exception, e:
        error = str(e)
        tweets[n]['sentistrength_error'] = error
    print(n)


simplejson.dump(tweets, open("tweets_with_sentistrength.json", "w"))
```

```python
#!/usr/bin/env python
#
#     Generates a plot of the evolution of the performance as
#     the word list is extended.
#
# $Id: Nielsen2011New_evolution.py,v 1.2 2011/03/13 23:48:38 fn Exp $


import csv
import re
import numpy as np
import pylab
import random
from scipy import sparse
from scipy.stats.stats import  kendalltau, spearmanr

# This variable determines the data set
mislove = 2

# Filenames
filebase = '/home/fnielsen/'
filename_afinn = filebase + 'fnielsen/data/Nielsen2009Responsible_emotion.csv'
filename_mislove1 = "results.txt"
filename_mislove2a = "turk.txt"
filename_mislove2b = "turk-results.txt"


# Word splitter pattern
pattern_split = re.compile(r"\W+")


afinn = dict(map(lambda (k,v): (k,int(v)),
                 [ line.split('\t') for line in open(filename_afinn) ]))


```

```

# Read Mislove CSV Twitter data: 'tweets' an array of dictionaries
if mislove == 1:
    csv_reader = csv.reader(open(filename_mislove1), delimiter='\t')
    header = []
    tweets = []
    for row in csv_reader:
        if not header:
            header = row
        else:
            tweets.append({'id': row[0],
                           'quant': int(row[1]),
                           'score_our': float(row[2]),
                           'score_mean': float(row[3]),
                           'score_std': float(row[4]),
                           'text': row[5],
                           'scores': map(int, row[6:])})
elif mislove == 2:
    csv_reader = csv.reader(open(filename_mislove2a), delimiter='\t')
    tweets = []
    for row in csv_reader:
        tweets.append({'id': row[2],
                       'score_mislove2': float(row[1]),
                       'text': row[3]})
    csv_reader = csv.reader(open(filename_mislove2b), delimiter='\t')
    tweets_dict = {}
    header = []
    for row in csv_reader:
        if not header:
            header = row
        else:
            tweets_dict[row[0]] = {'id': row[0],
                                   'score_mislove': float(row[1]),
                                   'score_amt_wrong': float(row[2]),
                                   'score_amt': np.mean(map(int, re.split("\s+", " ".join(row[4:]
                                   }
    for n in range(len(tweets)):
        tweets[n]['score_mislove'] = tweets_dict[tweets[n]['id']]['score_mislove']
        tweets[n]['score_amt_wrong'] = tweets_dict[tweets[n]['id']]['score_amt_wrong']
        tweets[n]['score_amt'] = tweets_dict[tweets[n]['id']]['score_amt']


allwords = []
for n in range(len(tweets)):
    words = pattern_split.split(tweets[n]['text'].lower())
    tweets[n]['words'] = words
    allwords.extend(words)

print("All words: %d" % len(allwords))
print("Number of unique words in Twitter corpus: %d" % len(set(allwords)))


terms = afinn.keys()
# terms = list(set(allwords).intersection(afinn.keys()))
print("Number of unique words matched to word list: %d" % len(terms))

term2index = dict(zip(terms, range(len(terms))))

M = sparse.lil_matrix((len(tweets), len(terms)))
M = np.zeros((len(tweets), len(terms)))   # Sparse is not necessary
for n in range(len(tweets)):
    for word in tweets[n]['words']:
        if term2index.has_key(word):
            M[n, term2index[word]] = afinn[word]

score_amt = [ t['score_amt'] for t in tweets ]

# Fix resampling seed for reproducibility
```

```
random.seed(a=1729)

K = [1, 10, 30, 50, 70, 100, 150, 200, 250, 300, 350, 400, len(terms)]
K = [5, 100, 300, 500, 1000, 1500, 2000, len(terms)]
I = range(len(terms))
resamples = 50
R = np.zeros((resamples, len(K)))
S = np.zeros((resamples, len(K)))
for n in range(len(K)):
    for m in range(resamples):
        J = random.sample(I, K[n])
        score_afinn = M[:,J].sum(axis=1)
        R[m,n] = np.corrcoef(score_amt, score_afinn)[0,1]
        S[m,n] = spearmanr(score_amt, score_afinn)[0]


# pylab.figure(1)
pylab.subplot(2, 1, 1)
pylab.boxplot(R, positions=K, widths=40)
pylab.ylabel('Pearson correlation')
# pylab.xlabel('Word list size')
pylab.title('Evolution of word list performance')
pylab.axis((0, 2600, -0.05, 0.65))
pylab.show()

# pylab.figure(2)
pylab.subplot(2, 1, 2)
pylab.boxplot(S, positions=K, widths=40)
pylab.ylabel('Spearman rank correlation')
pylab.xlabel('Word list size')
# pylab.title('Evolution of word list performance')
pylab.axis((0, 2600, 0.35, 0.6))
pylab.show()

# pylab.savefig(filebase + 'fnielsen/eps/Nielsen2011New_evolution.eps')
```

```
#!/usr/bin/env python
#
# $Id: Nielsen2011New_anewafinn.py,v 1.1 2011/03/13 17:18:10 fn Exp $



import csv
import re
import numpy as np
from nltk.stem.wordnet import WordNetLemmatizer
from nltk import sent_tokenize
import pylab
from scipy.stats.stats import kendalltau, spearmanr

# This variable determines the data set
mislove = 2

# Filenames
filebase = '/home/fnielsen/'
filename_afinn = filebase + 'fnielsen/data/Nielsen2009Responsible_emotion.csv'
filename_afinn96 = 'AFINN-96.txt'
filename_mislove1 = "results.txt"
filename_mislove2a = "turk.txt"
filename_mislove2b = "turk-results.txt"
filename_anew = filebase + 'data/ANEW.TXT'
filename_gi = filebase + "data/inqtabs.txt"
filename_of = filebase + "data/subjclueslen1-HLTEMNLP05.tff"

# Word splitter pattern
pattern_split = re.compile(r"\W+")

```

```
afinn = dict(map(lambda (k,v): (k,int(v)),
                 [ line.split('\t') for line in open(filename_afinn) ]))


# ANEW
anew = dict(map(lambda l: (l[0], float(l[2]) - 5) ,
                [ re.split('\s+', line) for line in open(filename_anew).readlines()[41:1075] ]))

anewafinn = set(anew.keys()).intersection(afinn.keys())


afinn_intersect = dict([ (k,afinn[k]) for k in anewafinn ])
anew_intersect = dict([ (k, anew[k]) for k in anewafinn ])


# Read Mislove CSV Twitter data: 'tweets' an array of dictionaries
if mislove == 1:
    csv_reader = csv.reader(open(filename_mislove1), delimiter='\t')
    header = []
    tweets = []
    for row in csv_reader:
        if not header:
            header = row
        else:
            tweets.append({'id': row[0],
                           'quant': int(row[1]),
                           'score_our': float(row[2]),
                           'score_mean': float(row[3]),
                           'score_std': float(row[4]),
                           'text': row[5],
                           'scores': map(int, row[6:])})
elif mislove == 2:
    csv_reader = csv.reader(open(filename_mislove2a), delimiter='\t')
    tweets = []
    for row in csv_reader:
        tweets.append({'id': row[2],
                       'score_mislove2': float(row[1]),
                       'text': row[3]})
    csv_reader = csv.reader(open(filename_mislove2b), delimiter='\t')
    tweets_dict = {}
    header = []
    for row in csv_reader:
        if not header:
            header = row
        else:
            tweets_dict[row[0]] = {'id': row[0],
                                   'score_mislove': float(row[1]),
                                   'score_amt_wrong': float(row[2]),
                                   'score_amt': np.mean(map(int, re.split("\s+", " ".join(row[4:
                                   }
    for n in range(len(tweets)):
        tweets[n]['score_mislove'] = tweets_dict[tweets[n]['id']]['score_mislove']
        tweets[n]['score_amt_wrong'] = tweets_dict[tweets[n]['id']]['score_amt_wrong']
        tweets[n]['score_amt'] = tweets_dict[tweets[n]['id']]['score_amt']



# Computer sentiment for each tweet
amt = []
afinn_intersect_sentiment = []
anew_intersect_sentiment = []
for n in range(len(tweets)):
    amt.append(tweets[n]['score_amt'])
    words = pattern_split.split(tweets[n]['text'].lower())
    afinn_sentiments = map(lambda word: afinn_intersect.get(word, 0), words)
    afinn_intersect_sentiment.append(float(sum(afinn_sentiments))/len(afinn_sentiments))
    anew_sentiments = map(lambda word: anew_intersect.get(word, 0), words)
```

```
    anew_intersect_sentiment.append(float(sum(anew_sentiments))/len(anew_sentiments))




np.corrcoef(amt, afinn_intersect_sentiment)[0,1]
np.corrcoef(amt, anew_intersect_sentiment)[0,1]

spearmanr(amt, afinn_intersect_sentiment)[0]
spearmanr(amt, anew_intersect_sentiment)[0]
```