VaderSentiment

Release 3.3.1

CJ Hutto

Jan 03, 2021

CONTENTS:

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is *specifically attuned to sentiments expressed in social media*. It is fully open-sourced under the [MIT License] (we sincerely appreciate all attributions and readily accept most contributions, but please don't hold us liable).

CHAPTER

ONE

INSTALLATION

There are a couple of ways to install and use VADER sentiment:

1. **The simplest is to use the command line to do an installation from [PyPI] using pip, e.g.,** `> pip install vaderSentiment`
2. **Or, you might already have VADER and simply need to upgrade to the latest version, e.g.,** `> pip install --upgrade vaderSentiment`
3. You could also clone this [GitHub repository]
4. You could download and unzip the [full master branch zip file]

In addition to the VADER sentiment analysis Python module, options 3 or 4 will also download all the additional resources and datasets (described below).

CHAPTER

TWO

VADER-SENTIMENT-ANALYSIS INTRODUCTION

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is *specifically attuned to sentiments expressed in social media*. It is fully open-sourced under the [MIT License] (we sincerely appreciate all attributions and readily accept most contributions, but please don't hold us liable).

It describes the dataset of the paper:

> **VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text**
> (by C.J. Hutto and Eric Gilbert)
> Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

For questions, please contact:

C.J. Hutto
Georgia Institute of Technology, Atlanta, GA 30032
cjhutto [at] gatech [dot] edu

CHAPTER

THREE

FEATURES AND UPDATES

Many thanks to George Berry, Ewan Klein, Pierpaolo Pantone for key contributions to make VADER better. The new updates includes capabilities regarding:

1. Refactoring for Python 3 compatibility, improved modularity, and incorporation into [NLTK] . . . many thanks to Ewan & Pierpaolo.
2. Restructuring for much improved speed/performance, reducing the time complexity from something like O(N^4) to O(N). . . many thanks to George.
3. Simplified pip install and better support for vaderSentiment module and component import. (Dependency on vader_lexicon.txt file now uses automated file location discovery so you don't need to manually designate its location in the code, or copy the file into your executing code's directory.)
4. More complete demo in the `__main__` for `vaderSentiment.py`. The demo has:
   - examples of typical use cases for sentiment analysis, including proper handling of sentences with:
     - typical negations (e.g., "*not* good")
     - use of contractions as negations (e.g., "*wasn't* very good")
     - conventional use of **punctuation** to signal increased sentiment intensity (e.g., "Good!!!")
     - conventional use of **word-shape** to signal emphasis (e.g., using ALL CAPS for words/phrases)
     - using **degree modifiers** to alter sentiment intensity (e.g., intensity *boosters* such as "very" and intensity *dampeners* such as "kind of")
     - understanding many **sentiment-laden slang** words (e.g., 'sux')
     - understanding many sentiment-laden **slang words as modifiers** such as 'uber' or 'friggin' or 'kinda'
     - understanding many sentiment-laden **emoticons** such as :) and :D
     - understanding sentiment-laden **initialisms and acronyms** (for example: 'lol')
   - more examples of **tricky sentences** that confuse other sentiment analysis tools
   - example for how VADER can work in conjunction with NLTK to do **sentiment analysis on longer texts**. . . i.e., decomposing paragraphs, articles/reports/publications, or novels into sentence-level analyses
   - examples of a concept for assessing the sentiment of images, video, or other tagged **multimedia content**
   - if you have access to the Internet, the demo has an example of how VADER can work with analyzing sentiment of **texts in other languages** (non-English text sentences).

CHAPTER

FOUR

RESOURCES AND DATASET DESCRIPTIONS

The package here includes **PRIMARY RESOURCES** (items 1-3) as well as additional **DATASETS AND TESTING RESOURCES** (items 4-12):

1. **vader_icwsm2014_final.pdf** The original paper for the data set, see citation information (above).
2. **vader_lexicon.txt**

   FORMAT: the file is tab delimited with TOKEN, MEAN-SENTIMENT-RATING, STANDARD DEVIATION, and R

   NOTE: The current algorithm makes immediate use of the first two elements (token and mean valence). The final two elements (SD and raw ratings) are provided for rigor. For example, if you want to follow the same rigorous process that we used for the study, you should find 10 independent humans to evaluate/rate each new token you want to add to the lexicon, make sure the standard deviation doesn't exceed 2.5, and take the average rating for the valence. This will keep the file consistent.

   DESCRIPTION: Empirically validated by multiple independent human judges, VADER incorporates a "gold-standard" sentiment lexicon that is especially attuned to microblog-like contexts.

   The VADER sentiment lexicon is sensitive both the **polarity** and the **intensity** of sentiments expressed in social media contexts, and is also generally applicable to sentiment analysis in other domains.

   Sentiment ratings from 10 independent human raters (all pre-screened, trained, and quality checked for optimal inter-rater reliability). Over 9,000 token features were rated on a scale from "[–4] Extremely Negative" to "[4] Extremely Positive", with allowance for "[0] Neutral (or Neither, N/A)". We kept every lexical feature that had a non-zero mean rating, and whose standard deviation was less than 2.5 as determined by the aggregate of those ten independent raters. This left us with just over 7,500 lexical features with validated valence scores that indicated both the sentiment polarity (positive/negative), and the sentiment intensity on a scale from –4 to +4. For example, the word "okay" has a positive valence of 0.9, "good" is 1.9, and "great" is 3.1, whereas "horrible" is –2.5, the frowning emoticon :(is –2.2, and "sucks" and it's slang derivative "sux" are both –1.5.

   Manually creating (much less, validating) a comprehensive sentiment lexicon is a labor intensive and sometimes error prone process, so it is no wonder that many opinion mining researchers and practitioners rely so heavily on existing lexicons as primary resources. We are pleased to offer ours as a new resource. We began by constructing a list inspired by examining existing well-established sentiment word-banks (LIWC, ANEW, and GI). To this, we next incorporate numerous lexical features common to sentiment expression in microblogs, including:

   - a full list of Western-style emoticons, for example, :-) denotes a smiley face and generally indicates positive sentiment
   - sentiment-related acronyms and initialisms (e.g., LOL and WTF are both examples of sentiment-laden initialisms)
   - commonly used slang with sentiment value (e.g., nah, meh and giggly).

We empirically confirmed the general applicability of each feature candidate to sentiment expressions using a wisdom-of-the-crowd (WotC) approach (Surowiecki, 2004) to acquire a valid point estimate for the sentiment valence (polarity & intensity) of each context-free candidate feature.

3. **vaderSentiment.py** The Python code for the rule-based sentiment analysis engine. Implements the grammatical and syntactical rules described in the paper, incorporating empirically derived quantifications for the impact of each rule on the perceived intensity of sentiment in sentence-level text. Importantly, these heuristics go beyond what would normally be captured in a typical bag-of-words model. They incorporate **word-order sensitive relationships** between terms. For example, degree modifiers (also called intensifiers, booster words, or degree adverbs) impact sentiment intensity by either increasing or decreasing the intensity. Consider these examples:

   (a) "The service here is extremely good"

   (b) "The service here is good"

   (c) "The service here is marginally good"

   From Table 3 in the paper, we see that for 95% of the data, using a degree modifier increases the positive sentiment intensity of example (a) by 0.227 to 0.36, with a mean difference of 0.293 on a rating scale from 1 to 4. Likewise, example (c) reduces the perceived sentiment intensity by 0.293, on average.

4. **tweets_GroundTruth.txt** FORMAT: the file is tab delimited with ID, MEAN-SENTIMENT-RATING, and TWEET-TEXT

   DESCRIPTION: includes "tweet-like" text as inspired by 4,000 tweets pulled from Twitter's public timeline, plus 200 completely contrived tweet-like texts intended to specifically test syntactical and grammatical conventions of conveying differences in sentiment intensity. The "tweet-like" texts incorporate a fictitious username (@anonymous) in places where a username might typically appear, along with a fake URL (http://url_removed) in places where a URL might typically appear, as inspired by the original tweets. The ID and MEAN-SENTIMENT-RATING correspond to the raw sentiment rating data provided in 'tweets_anonDataRatings.txt' (described below).

5. **tweets_anonDataRatings.txt** FORMAT: the file is tab delimited with ID, MEAN-SENTIMENT-RATING, STANDARD DEVIATION, and RAW-SENTIMENT-RATINGS

   DESCRIPTION: Sentiment ratings from a minimum of 20 independent human raters (all pre-screened, trained, and quality checked for optimal inter-rater reliability).

6. **nytEditorialSnippets_GroundTruth.txt** FORMAT: the file is tab delimited with ID, MEAN-SENTIMENT-RATING, and TEXT-SNIPPET

   DESCRIPTION: includes 5,190 sentence-level snippets from 500 New York Times opinion news editorials/articles; we used the NLTK tokenizer to segment the articles into sentence phrases, and added sentiment intensity ratings. The ID and MEAN-SENTIMENT-RATING correspond to the raw sentiment rating data provided in 'nytEditorialSnippets_anonDataRatings.txt' (described below).

7. **nytEditorialSnippets_anonDataRatings.txt** FORMAT: the file is tab delimited with ID, MEAN-SENTIMENT-RATING, STANDARD DEVIATION, and RAW-SENTIMENT-RATINGS

   DESCRIPTION: Sentiment ratings from a minimum of 20 independent human raters (all pre-screened, trained, and quality checked for optimal inter-rater reliability).

8. **movieReviewSnippets_GroundTruth.txt** FORMAT: the file is tab delimited with ID, MEAN-SENTIMENT-RATING, and TEXT-SNIPPET

   DESCRIPTION: includes 10,605 sentence-level snippets from rotten.tomatoes.com. The snippets were derived from an original set of 2000 movie reviews (1000 positive and 1000 negative) in Pang & Lee (2004); we used the NLTK tokenizer to segment the reviews into sentence phrases, and added sentiment intensity ratings. The ID and MEAN-SENTIMENT-RATING correspond to the raw sentiment rating data provided in 'movieReviewSnippets_anonDataRatings.txt' (described below).

9. **movieReviewSnippets_anonDataRatings.txt** FORMAT: the file is tab delimited with ID, MEAN-SENTIMENT-RATING, STANDARD DEVIATION, and RAW-SENTIMENT-RATINGS

   DESCRIPTION: Sentiment ratings from a minimum of 20 independent human raters (all pre-screened, trained, and quality checked for optimal inter-rater reliability).

10. **amazonReviewSnippets_GroundTruth.txt** FORMAT: the file is tab delimited with ID, MEAN-SENTIMENT-RATING, and TEXT-SNIPPET

    DESCRIPTION: includes 3,708 sentence-level snippets from 309 customer reviews on 5 different products. The reviews were originally used in Hu & Liu (2004); we added sentiment intensity ratings. The ID and MEAN-SENTIMENT-RATING correspond to the raw sentiment rating data provided in 'amazonReviewSnippets_anonDataRatings.txt' (described below).

11. **amazonReviewSnippets_anonDataRatings.txt** FORMAT: the file is tab delimited with ID, MEAN-SENTIMENT-RATING, STANDARD DEVIATION, and RAW-SENTIMENT-RATINGS

    DESCRIPTION: Sentiment ratings from a minimum of 20 independent human raters (all pre-screened, trained, and quality checked for optimal inter-rater reliability).

12. **Comp.Social website with more papers/research:** [Comp.Social](http://comp.social.gatech.edu/papers/)

CHAPTER

FIVE

PYTHON CODE EXAMPLE

For a **more complete demo**, point your terminal to vader's install directory (e.g., if you installed using pip, it might be `\Python3x\lib\site-packages\vaderSentiment`), and then run `python vaderSentiment.py`.

The demo has more examples of tricky sentences that confuse other sentiment analysis tools. It also demonstrates how VADER can work in conjunction with NLTK to do sentiment analysis on longer texts. . . i.e., decomposing paragraphs, articles/reports/publications, or novels into sentence-level analysis. It also demonstrates a concept for assessing the sentiment of images, video, or other tagged multimedia content.

If you have access to the Internet, the demo will also show how VADER can work with analyzing sentiment of non-English text sentences.

```
    from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
    #note: depending on how you installed (e.g., using source code download versus␣
↪pip install), you may need to import like this:
    #from vaderSentiment import SentimentIntensityAnalyzer

# --- examples -------
sentences = ["VADER is smart, handsome, and funny.",       # positive sentence example
             "VADER is not smart, handsome, nor funny.",   # negation sentence example
             "VADER is smart, handsome, and funny!",       # punctuation emphasis␣
↪handled correctly (sentiment intensity adjusted)
             "VADER is very smart, handsome, and funny.",  # booster words handled␣
↪correctly (sentiment intensity adjusted)
             "VADER is VERY SMART, handsome, and FUNNY.",  # emphasis for ALLCAPS␣
↪handled
             "VADER is VERY SMART, handsome, and FUNNY!!!",# combination of signals -␣
↪VADER appropriately adjusts intensity
             "VADER is VERY SMART, uber handsome, and FRIGGIN FUNNY!!!",# booster␣
↪words & punctuation make this close to ceiling for score
             "The book was good.",                                    # positive␣
↪sentence
             "The book was kind of good.",                 # qualified positive␣
↪sentence is handled correctly (intensity adjusted)
             "The plot was good, but the characters are uncompelling and the dialog is␣
↪not great.", # mixed negation sentence
             "At least it isn't a horrible book.",         # negated negative sentence␣
↪with contraction
             "Make sure you :) or :D today!",              # emoticons handled
             "Today SUX!",                                 # negative slang with␣
↪capitalization emphasis
             "Today only kinda sux! But I'll get by, lol"  # mixed sentiment example␣
↪with slang and constrastive conjunction "but"
              ]
```

(continues on next page)

(continued from previous page)

```
analyzer = SentimentIntensityAnalyzer()
for sentence in sentences:
    vs = analyzer.polarity_scores(sentence)
    print("{:-<65} {}".format(sentence, str(vs)))
```

For a **more complete demo**, go to the install directory and run `python vaderSentiment.py`. (Be sure you are set to handle UTF-8 encoding in your terminal or IDE.)

CHAPTER

SIX

OUTPUT FOR THE ABOVE EXAMPLE CODE

```
VADER is smart, handsome, and funny.---------------------------- {'neg': 0.0, 'neu':␣
↪0.254, 'pos': 0.746, 'compound': 0.8316}
VADER is not smart, handsome, nor funny.------------------------ {'neg': 0.646, 'neu
↪': 0.354, 'pos': 0.0, 'compound': -0.7424}
VADER is smart, handsome, and funny!---------------------------- {'neg': 0.0, 'neu':␣
↪0.248, 'pos': 0.752, 'compound': 0.8439}
VADER is very smart, handsome, and funny.----------------------- {'neg': 0.0, 'neu':␣
↪0.299, 'pos': 0.701, 'compound': 0.8545}
VADER is VERY SMART, handsome, and FUNNY.----------------------- {'neg': 0.0, 'neu':␣
↪0.246, 'pos': 0.754, 'compound': 0.9227}
VADER is VERY SMART, handsome, and FUNNY!!!---------------------- {'neg': 0.0, 'neu':␣
↪0.233, 'pos': 0.767, 'compound': 0.9342}
VADER is VERY SMART, uber handsome, and FRIGGIN FUNNY!!!--------- {'neg': 0.0, 'neu':␣
↪0.294, 'pos': 0.706, 'compound': 0.9469}
The book was good.---------------------------------------------- {'neg': 0.0, 'neu':␣
↪0.508, 'pos': 0.492, 'compound': 0.4404}
The book was kind of good.-------------------------------------- {'neg': 0.0, 'neu':␣
↪0.657, 'pos': 0.343, 'compound': 0.3832}
The plot was good, but the characters are uncompelling and the dialog is not great. {
↪'neg': 0.327, 'neu': 0.579, 'pos': 0.094, 'compound': -0.7042}
At least it isn't a horrible book.------------------------------ {'neg': 0.0, 'neu':␣
↪0.637, 'pos': 0.363, 'compound': 0.431}
Make sure you :) or :D today!----------------------------------- {'neg': 0.0, 'neu':␣
↪0.294, 'pos': 0.706, 'compound': 0.8633}
Today SUX!------------------------------------------------------ {'neg': 0.779, 'neu
↪': 0.221, 'pos': 0.0, 'compound': -0.5461}
Today only kinda sux! But I'll get by, lol---------------------- {'neg': 0.179, 'neu
↪': 0.569, 'pos': 0.251, 'compound': 0.2228}
```

CHAPTER

SEVEN

ABOUT THE SCORING

- The `compound` score is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 (most extreme negative) and +1 (most extreme positive). This is the most useful metric if you want a single unidimensional measure of sentiment for a given sentence. Calling it a 'normalized, weighted composite score' is accurate.

  It is also useful for researchers who would like to set standardized thresholds for classifying sentences as either positive, neutral, or negative. Typical threshold values (used in the literature cited on this page) are:

1. **positive sentiment**: `compound` score >= 0.5
2. **neutral sentiment**: (`compound` score > -0.5) and (`compound` score < 0.5)
3. **negative sentiment**: `compound` score <= -0.5

- The `pos`, `neu`, and `neg` scores are ratios for proportions of text that fall in each category (so these should all add up to be 1... or close to it with float operation). These are the most useful metrics if you want multidimensional measures of sentiment for a given sentence.

CHAPTER

EIGHT

PORTS TO OTHER PROGRAMMING LANGUAGES

Feel free to let me know about ports of VADER Sentiment to other programming languages. So far, I know about these helpful ports:

1. **Java** VaderSentimentJava by apanimesh061
2. **PHP** php-vadersentiment by abusby
3. **Scala** Sentiment by ziyasal
4. **JavaScript** coming soon

CHAPTER

NINE

LINKS

- search
- Source code: https://github.com/vaderSentiment/vaderSentiment
- Contact: cjhutto@gatech.edu
- license = 'MIT License'