SentiStrength Manual

SentiStrength is a sentiment analysis tool developed by Mike Thelwall etc [1] in 2010. In this manual, we will first briefly introduce this tool, describe its functions, and show how to use this tool. More details can be found in its paper and SentiStrength’s official website [2].

Introduce

SentiStrength is a tool developed from comments on social networking sites (MySpace [3]). Its core function is to use dictionary-based algorithms to analyse sentiment for text. Specifically, it first assigns priori sentimental scores to words according to the sentiment dictionary, and then adjusts the assignation result with several heuristic rules. It can give a sentimental score pair (ρ, η) for each input text, where ρ represents the positive score of the text, and η represents the negative score. The scale and meaning for ρ and η are as below:

[no positive sentiment] “+**1”– “+2” – “+3” – “+4” – “+5”** [very strong positive sentiment]

[no negative sentiment] “-**1” – “ -2” – “-3” – “-4” – “-5”** [very strong negative sentiment]

Core Function

SentiStrength's core function is to use a set of dictionaries and several heuristic rules to conduct sentimental analysis on the text. The key elements of SentiStrength are listed below.

* **UC-1 Assigning** **Sentiment** **Scores for Words**: The core of the algorithm is the **sentiment** **word** **strength** **list** (EmotionLookupTable in SentiStrength). EmotionLookupTable contains 2546 words or wildcards (hereinafter collectively referred to as items). Each item was preseted a sentimental score, which is an integer ranging from -5 to 5. SentiStrength will assign sentimental scores to each word in the sentence based on the EmotionLookupTable. If the word does not exist in the EmotionLookupTable, it defaults to neutral. It is noteworthy that the word “**miss**” was allocated a positive and negative strength of 2. This was the only word classed as both positive and negative. It was typically used in the phrase “I miss you”, suggesting both sadness and love.
* **UC**-**2** **Assigning Sentiment Scores for Phrases**: EmotionLookupTable is used to assign sentimental scores for uni-grams, while IdiomLookupTable is used to assign scores for the phrases which often contain multiple words. When an idiom is recognized, the sentimental score of the idiom will cover the sentimental score of the single word that constitutes the idiom. For example, In the text “It’s a killer feature.”, “killer feature” is a phrase in the dictionary with positive score 02. Although the word ‘kill’ carries negative sentiment, its effect is overridden by the sentimental score of the enclosing phrase. So the text is analyzed as positive finally.
* **UC-3 Spelling Correction**: A algorithm identifies the standard spellings of words that have been miss-spelled by the inclusion of repeated letters. For example, hellllloooo would be identified as "hello" by this algorithm. The algorithm (a) automatically deletes repeated letters above twice (e.g., helllo -> hello); (b) deletes repeated letters occurring twice for letters rarely occurring twice in English (e.g., niice -> nice) and (c) deletes letters occurring twice if not a standard word but would form a standard word if deleted (e.g., nnice -> nice but not hoop -> hop nor baaz -> baz). EnglishWordList will be used to check whether the spelling of the word is correct.
* **UC-4 Booster Word Rule**: A booster word list (BoosterWordList) contains words that boost or reduce the emotion of subsequent words, whether positive or negative. Each word increases emotion strength by 1 or 2 (e.g., very, extremely) or decreases it by 1 (e.g., some).
* **UC-5 Negating Word Rule**: A negating word list (NegatingWordList) contains words that invert subsequent emotion words (including any preceding booster words). For example, if "very happy" had positive strength 4 then "not very happy" would have negative strength 4. The possibility that some negating terms do not negate was not incorporated as this did not seem to occur often in the pilot data set.
* **UC-6 Repeated** **Letter** **Rule**: Repeated letters above those needed for correct spelling are used to give a strength boost of 1 to sentimental words, as long as there are at least two additional letters. The use of repeated letters is a common device for expressing emotion or energy in MySpace comments, but one repeated letter often appeared to be a typing error.
* **UC-7 Emoji** **Rule**: An emoticon list (EmotionLookupTable) with associated strengths (positive or negative 2) supplements the sentiment word strength list (and punctuation included in emoticons is not processed further for the purposes below).
* **UC-8 Exclamation** **Mark** **Rule**: Any sentence with an exclamation mark was allocated a minimum positive strength of 2.
* **UC-9 Repeated** **Punctuation** **Rule**: Repeated punctuation including at least one exclamation mark gives a strength boost of 1 to the immediately preceding emotion word (or sentence).
* **UC-10 Negative** **Sentiment** **Ignored** **in** **Questions**. For example, the question "are you angry?" would be classified as not containing sentiment, despite the presence of the word "angry". This was not applied to positive sentiment because many question sentences appeared to contain mild positive sentiment. In particular, sentences like "whats up?" were typically classified as containing mild positive sentiment (strength 2). QuestionWord is used to identify question words.

The above factors were applied separately to each sentence, with the sentence being assigned with both the most positive and most negative sentiments identified in it. Each overall text was assigned with the most positive of its sentence sentiments and the most negative of its sentence sentiments. Sentences were split either by line breaks in comments or after punctuation other than emoticons. The example in Table 1 shows how SentiStrength analyses.

# Other Functions

Table 1 The Sample of SentiStrength

| **Sample** | **ρ** | **η** | **Dictionary/Rule**  **Used** | **Explanation** |
| --- | --- | --- | --- | --- |
| It’s a good feature. | 2 | -1 | EmotionLookupTable | The sentimental score of the word *‘good’* is pre-assigned to 02; so the sentence is assigned positive score 02. |
| It’s a very good feature. | 3 | -1 | EmotionLookupTable BoosterWordList | As booster word ‘very’ is used before the sentimental word, the sentence is assigned a positive score 03. |
| It’s not good feature. | 1 | -2 | EmotionLookupTable  NegatingWordList | Sentimental polarity of the sentimental word is inverted in here due to the use of the negation word ‘not’ before sentimental word. |
| It’s a good feature! | 3 | -1 | EmotionLookupTable  “！” Rule | “!”will strengthen the sentimental strength |
| It’s a gooood feature. | 3 | -1 | Repeated Letter Rule | Repeated letters above those needed for correct spelling are used to give a strength boost of 1 to sentimental words. |

SentiStrength was initially released in the form of a jar package. In this section, we will introduce some non-core functions and explain how to set options to use these functions when analysing. For a more comprehensive description of options, please refer to the manual on the official website.

# Complete Different Classification Tasks (6)

SentiStrength can classify individual texts or multiple texts and can be invoked in many different ways. This section covers these methods although most users only need one of them.

# UC-11 Classify a single text

text [text to process]

The submitted text will be classified and the result returned in the form +ve –space- -ve. If the classification method is trinary, binary or scale then the result will have the form +ve –space- -ve –space- overall. E.g.,

java -jar SentiStrength.jar sentidata C:/SentiStrength\_Data/ text i+love+your+dog.

The result will be: 3 -1

# UC-12 Classify all lines of text in a file for sentiment [includes accuracy evaluations]

input [filename]

Each line of [filename] will be classified for sentiment. Here is an example.

java -jar SentiStrength.jar sentidata C:/SentiStrength\_Data/ input myfile.txt

A new file will be created with the sentiment classifications added to the end of each line.

If the task is to test the accuracy of SentiStrength, then the file may have +ve codes in the 1st column, then negative codes in the 2nd column and text in the last column. If using binary/trinary/scale classification then the first column can contain the human coded values. Columns must be tab-separated. If human coded sentiment scores are included in the file then the accuracy of SentiStrength will be compared against them.

# UC-13 Classify texts in a column within a file or folder

For each line, the text in the specified column will be extracted and classified, with the result added to an extra column at the end of the file (all three parameters are compulsory).

annotateCol [col # 1..] (classify text in col, result at line end)

inputFolder [foldername] (all files in folder will be \*annotated\*)

fileSubstring [text] (string must be present in files to annotate)

Ok to overwrite files [overwrite]

If a folder is specified instead of a filename (i.e., an input parameter) then all files in the folder are processed as above. If a fileSubstring value is specified, then only files matching the substring will be classified. The parameter overwrite must be specified to explicitly allow the input files to be modified. This is a purely safety feature. E.g.,

java -jar SentiStrength.jar sentidata C:/SentiStrength\_Data/ annotateCol 1 inputFolder C:/textfiles/ fileSubstring txt

# UC-14 Listen at a port for texts to classify

listen [port number to listen at - call OR

This sets the program to listen at a port number for texts to classify, e.g., to listen at port 81 for texts for trinary classification:

java -jar SentiStrength.jar sentidata C:/SentiStrength\_Data/ listen 81 trinary

The texts must be URLEncoded and submitted as part of the URL. E.g., if the listening

was set up on port 81 then requesting the following URL would trigger classification of

the text "love you": http://127.0.0.1:81/love%20you

The result for this would be 3 -1 1. This is: (+ve classification) (-ve classification) (trinary

classification)

# UC-15 Run interactively from the command line

cmd (can also set options and sentidata folder). E.g.,

java -jar c:\SentiStrength.jar cmd sentidata C:/SentiStrength\_Data/

This allows the program to classify texts from the command prompt. After running this every line you enter will be classified for sentiment. To finish enter @end

# UC-16 Process stdin and send to stdout

stdin (can also set options and sentidata folder). E.g.,

java -jar c:\SentiStrength.jar stdin sentidata C:/SentiStrength\_Data/

SentiStrength will classify all texts sent to it from stdin and then will close. This probably the most efficient way of integrating SentiStrength efficiently with non-Java programs. The alternatives are the Listen at a port option or dumping the texts to be classified into a file and then running SentiStrength on the file.

The parameter textCol can be set [default 0 for the first column] if the data is sent in multiple tab-separated columns and one column contains the text to be classified. The results will be appended to the end of the input data and send to STD out. The Java loop code for this is essentially:

**while**((textToParse = stdin.readLine()) != **null**) {

     //code to analyse sentiment and return results

}

So for greatest efficiency, null should not be sent to stdin as this will close the program.

# Set Location of Data (4)

# UC-17 Location of linguistic data folder

sentidata [folder for SentiStrength data (end in slash, no spaces)]

This option is used to set which folder the tool searches for the dictionary needed for analysis ( such as EmotionLookupTable, IdiomLookupTable, etc).

# UC-18 Location of sentiment term weights

EmotionLookupTable [filename (default: EmotionLookupTable.txt or

SentimentLookupTable.txt)].

This option is used to set which file will be set as the core sentiment word strength list for the tool. The default value is EmotionLookupTable.txt or SentimentLookupTable.txt. This file must be in the directory specified by sentidata .

# UC-19 Location of output folder

outputFolder [foldername where to put the output (default: folder of input)]

This option is used to set the name of the folder to put the output.

# UC-20 File name extension for output

resultsextension [file-extension for output (default \_out.txt)]

This option is used to set what identifier will be used to mark out the output file. Its default value is “\_out.txt”. For example, if you set "input.txt" as the input file, the first output file will be “input0\_out.txt” (input file name + index + result extension).

# Set Different Type of Output (4)

# UC-21 Classify positive (1 to 5) and negative (-1 to -5) sentiment strength separately

This is the default and is used unless binary, trinary or scale is selected. Note that 1 indicates no positive sentiment and -1 indicates no negative sentiment. There is no output of 0.

# UC-22 Use trinary classification (positive-negative-neutral)

trinary (report positive-negative-neutral classification instead)

The result for this would be like 3 -1 1. This is: (+ve classification) (-ve classification) (trinary classification)

# UC-23 Use binary classification (positive-negative)

binary (report positive-negative classification instead)

The result for this would be like 3 -1 1. This is: (+ve classification) (-ve classification) (binary classification)

# UC-24 Use a single positive-negative scale classification

scale (report single -4 to +4 classification instead)

The result for this would be like 3 -4 -1. This is: (+ve classification) (-ve classification) (scale classification)

# UC-25 Explain the classification

explain

Adding this parameter to most of the options results in an approximate explanation being given for the classification. E.g.,

java -jar SentiStrength.jar text i+don't+hate+you. explain

# UC-26 Set Classification Algorithm Parameters

Please note that most of these options can be mapped to the core function of SentiStrength. They can change how the sentiment analysis algorithm works.

* alwaysSplitWordsAtApostrophes (split words when an apostrophe is met – important for languages that merge words with ‘, like French (e.g., t’aime -> t ‘ aime with this option t’aime without))
* noBoosters (ignore sentiment booster words (e.g., very))
* noNegatingPositiveFlipsEmotion (don't use negating words to flip +ve words)
* noNegatingNegativeNeutralisesEmotion (don't use negating words to neuter -ve words)
* negatedWordStrengthMultiplier (strength multiplier when negated (default=0.5))
* maxWordsBeforeSentimentToNegate (max words between negator & sentiment word (default 0))
* noIdioms (ignore idiom list)
* questionsReduceNeg (-ve sentiment reduced in questions)
* noEmoticons (ignore emoticon list)
* exclamations2 (exclamation marks count them as +2 if not -ve sentence)
* mood [-1,0,1](interpretation of neutral emphasis (e.g., miiike; hello!!). -1 means neutral emphasis interpreted as –ve; 1 means interpreted as +ve; 0 means emphasis ignored)
* noMultiplePosWords (don't allow multiple +ve words to increase +ve sentiment)
* noMultipleNegWords (don't allow multiple -ve words to increase -ve sentiment)
* noIgnoreBoosterWordsAfterNegatives (don't ignore boosters after negating words)
* noDictionary (don't try to correct spellings using the dictionary by deleting duplicate letters from unknown words to make known words)
* noDeleteExtraDuplicateLetters (don't delete extra duplicate letters in words even when they are impossible, e.g., heyyyy) [this option does not check if the new word is legal, in contrast to the above option]
* illegalDoubleLettersInWordMiddle [letters never duplicate in word middles] this is a list of characters that never occur twice in succession. For English the following list is used (default): ahijkquvxyz Never include w in this list as it often occurs in www
* illegalDoubleLettersAtWordEnd [letters never duplicate at word ends] this is a list of characters that never occur twice in succession at the end of a word. For English the following list is used (default): achijkmnpqruvwxyz
* noMultipleLetters (don't use the presence of additional letters in a word to boost sentiment)

# Improving the accuracy of SentiStrength(2)

# Basic manual improvements

If you see a systematic pattern in the results, such as the term “disgusting” typically having a stronger or weaker sentiment strength in your texts than given by SentiStrength then you can edit the text files with SentiStrength to change this. Please edit SentiStrength’s input files using a plain text editor because if it is edited with a word processor then SentiStrength may not be able to read the file afterwards.

# UC-27 Optimise sentiment strengths of existing sentiment terms

SentiStrength can suggest revised sentiment strengths for the EmotionLookupTable.txt in order to give more accurate classifications for a given set of texts. This option needs a large (>500) set of texts in a plain text file with a human sentiment classification for each text. SentiStrength will then try to adjust the EmotionLookupTable.txt term weights to be more accurate when classifying these texts. It should then also be more accurate when classifying similar texts.

optimise [Filename for optimal term strengths (e.g. EmotionLookupTable2.txt)]

This creates a new emotion lookup table with improved sentiment weights based upon an input file with human coded sentiment values for the texts. This feature allows SentiStrength term weights to be customised for new domains. E.g.,

java -jar c:/SentiStrength.jar minImprovement 3 input C:/twitter4242.txt optimise C:/twitter4242OptimalSentimentLookupTable.txt

This is very slow (hours or days) if the input file is large (hundreds of thousands or millions, respectively). The main optional parameter is minImprovement (default value 2). Set this to specify the minimum overall number of additional correct classifications to change the sentiment term weighting. For example, if increasing the sentiment strength of love from 3 to 4 improves the number of correctly classified texts from 500 to 502 then this change would be kept if minImprovement was 1 or 2 but rejected if minImprovement was >2. Set this higher to have more robust changes to the dictionary. Higher settings are possible with larger input files.

To check the performance on the new dictionary, the file could be reclassified using it instead of the original SentimentLookupTable.txt as follows:

java -jar c:/SentiStrength.jar input C:/twitter4242.txt EmotionLookupTable C:/twitter4242OptimalSentimentLookupTable.txt

# UC-28 Suggest new sentiment terms (from terms in misclassified texts)

SentiStrength can suggest a new set of terms to add to the EmotionLookupTable.txt in order to give more accurate classifications for a given set of texts. This option needs a large (>500) set of texts in a plain text file with a human sentiment classification for each text. SentiStrength will then list words not found in the EmotionLookupTable.txt that may indicate sentiment. Adding some of these terms should make SentiStrength more accurate when classifying similar texts.

termWeights

This lists all terms in the data set and the proportion of times they are in incorrectly classified positive or negative texts. Load this into a spreadsheet and sort on the PosClassAvDiff and NegClassAvDiff to get an idea about terms that either should be added to the sentiment dictionary because one of these two values is high. This option also lists words that are already in the sentiment dictionary. Must be used with a text file containing correct classifications. E.g.,

java -jar c:/SentiStrength.jar input C:/twitter4242.txt termWeights

This is very slow (hours or days) if the input file is large (tens of thousands or millions, respectively).

**Interpretation**: In the output file, the column PosClassAvDiff means the average difference between the predicted sentiment score and the human classified sentiment score for texts containing the word. For example, if the word “nasty” was in two texts and SentiStrength had classified them both as +1,-3 but the human classifiers had classified the texts as (+2,-3) and (+3,-5) then PosClassAvDiff would be the average of 2-1 (first text) and 3-1 (second text) which is 1.5. All the negative scores are ignored for PosClassAvDiff

NegClassAvDiff is the same as for PosClassAvDiff except for the negative scores.

# UC-29 Machine learning evaluations

These are machine learning options to evaluate SentiStrength for academic research. The basic command is train.

*train* (evaluate SentiStrength by training term strengths on results in file). An input file of 500+ human classified texts is also needed - e.g.,

java -jar SentiStrength.jar train input C:\1041MySpace.txt

This attempts to optimise the sentiment dictionary using a machine learning approach and 10-fold cross validation. This is equivalent to using the command optimise on a random 90% of the data, then evaluating the results on the remaining 10% and repeating this 9 more times with the remaining 9 sections of 10% of the data. The accuracy results reported are the average of the 10 attempts. This estimates the improved accuracy gained from using the optimise command to improve the sentiment dictionary.

The output of this is two files. The file ending in **\_out.txt** reports various accuracy statistics (e.g., number and proportion correct, number and proportion within 1 of the correct value; correlation between SentiStrength and human coded values. The file ending in **\_out\_termStrVars.txt** reports the changes to the sentiment dictionary in each of the folds. Both files also report the parameters used for the sentiment algorithm and machine learning. See the What do the results mean? section at the end for more information.

# Evaluation options

* *all* Test all option variations listed in Classification Algorithm Parameters above rather than use the default options
* *tot* Optimise by the number of correct classifications rather than the sum of the classification differences
* *iterations* [number of 10-fold iterations (default 1)] This sets the number of times that the training and evaluation is conducted. A value of 30 is recommended to help average out differences between runs.
* minImprovement [min extra correct class. to change sentiment weights (default 2)] This sets the minimum number of extra correct classifications necessary to adjust a term weight during the training phase.
* *multi* [# duplicate term strength optimisations to change sentiment weights (default 1)] This is a kind of super-optimisation. Instead of being optimised once, term weights are optimised multiple times from the starting values and then the average of these weights is taken and optimised and used as the final optimised term strengths. This should in theory give better values than optimisation once. e.g.,

java -jar SentiStrength.jar multi 8 input C:\1041MySpace.txt iterations 2

**Example: Using SentiStrength for 10-fold cross-validation**

**What is this?** This estimates the accuracy of SentiStrength *after* it has optimised the term weights for the sentiment words (i.e., the values in the file EmotionLookupTable.txt).

**What do I need for this test?** You need an input file that is a list of texts with human classified values for positive (1-5) and negative (1-5) sentiment. Each line of the file should be in the format:

Positive <tab> Negative <tab> text

**How do I run the test?** Type the following command, replacing the filename with your own file name.

java -jar SentiStrength.jar input C:\1041MySpace.txt iterations 30

This should take up to one hour – much longer for longer files. The output will be a list of accuracy statistics. Each 10-fold cross-validation

**What does 10-fold cross-validation mean?** See the k-fold section in <http://en.wikipedia.org/wiki/Cross-validation_(statistics)>. Essentially, it means that the same data is used to identify the best sentiment strength values for the terms in EmotionLookupTable.txt as is used to evaluate the accuracy of the revised (trained) algorithm – but this isn’t cheating when it is done this way.

The first line in the results file gives the accuracy of SentiStrength with the original term weights in EmotionLookupTable.txt.

**What do the results mean?** The easiest way to read the results is to copy and paste them into a spreadsheet like Excel. The table created lists the options used to classify the texts and well as the results. Here is an extract from the first two rows of the key results. It gives the total number correct for positive sentiment (Pos Correct) and the proportion correct (Pos Correct/Total). It also reports the number of predictions that are correct or within 1 of being correct (Pos Within1). The same information is given for negative sentiment.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Pos  Correct | Pos Correct/  Total | Neg  Correct | Neg Correct/  Total | Pos  Within1 | Pos  Within1/  Total | Neg  Within1 | Neg Within1/  Total |
| 653 | 0.627281 | 754 | 0.724304 | 1008 | 0.9683 | 991 | 0.951969 |

Here is another extract of the first two rows of the key results. It gives the correlation between the positive sentiment predictions and the human coded values for positive sentiment (Pos Corr) and the Mean Percentage Error (PosMPEnoDiv). The same information is given for negative sentiment.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Pos Corr | NegCorr | PosMPE | NegMPE | PosMPEnoDiv | NegMPEnoDiv |
| 0.638382 | 0.61354 | Ignore this | Ignore this | 0.405379 | 0.32853 |

If you specified 30 iterations then there will be 31 rows, one for the header and 1 for each iteration. Take the average of the rows as the value to use.

Reference

1. THELWALL M, BUCKLEY K, PALTOGLOU G, et al. Sentiment strength detection in short informal text[J]. Journal of the American society for information science and technology, 2010, 61(12) : 2544 – 2558.
2. <http://sentistrength.wlv.ac.uk>
3. https://myspace.com/