

## **Sentiment Analysis of Book Reviews**

Computer Systems Lab 2013-2014  
Jocelyn Huang



## **Abstract**

*This project's purpose was to determine the sentiment related to given input text, with book reviews being the main focus. Sentiment analysis and natural language processing as a whole are methods of information gathering and are part of research fields that are gaining in popularity, as they play integral parts in areas such as artificial intelligence and public relations. This project employed a word-weighting algorithm in order to calculate an overall sentiment value of the input; the weightings were obtained from processing sample data.*

## **Introduction**

This project was an attempt to create a program capable of performing simple sentiment analysis through machine learning and through processing test data (which consisted of practice book reviews). The entire process consisted of three broad stages: the initial collection of the test data, data processing and weighting for machine learning, and then the sentiment analysis itself. Test data was obtained from a site called Goodreads.com and the data and words were processed using reviews found on the site, along with their associated weightings.

The front end of the project was hosted on the Thomas Jefferson High School for Science and Technology (TJHSST) server<sup>1</sup>, and ran with PHP and Python 3. However, the bulk of the coding and data collection work was performed on a personal MacBook Air laptop using a Linux virtual machine. Data collection was facilitated by Python's BeautifulSoup4 external library<sup>2</sup>, and the collected book reviews and word data were stored in a MySQL database maintained on the personal laptop.

---

<sup>1</sup> <http://www.tjhsst.edu>

<sup>2</sup> More information can be found at <http://www.crummy.com/software/BeautifulSoup/>

## **Background**

With the advent of big data, companies are now able to implement sentiment analysis algorithms in order to review public reception of their products and services on a large scale by gathering information on their customers online<sup>3</sup>. Understandably, finding out whether reception was positive or negative can be helpful in marketing, and with algorithms to automate the process, the field and its associated research and opportunities become all the more appealing. Hence, sentiment analysis, along with natural language processing as a whole, are being increasingly thoroughly researched, with IBM Watson (the famed artificial intelligence Jeopardy contender) as a particularly notable product.

However, as the algorithms used in sentiment analysis and other branches of natural language processing progress in complexity, they are still hard-pressed to accurately glean the correct meanings and nuances from natural language text they are given due to the complexity of language and its various tones that might be apparent to humans but not to machines<sup>4</sup>. Even IBM Watson slipped up

---

<sup>3</sup> A good summary can be found at <http://blogs.informatica.com/perspective/s/2011/06/27/big-data-meets-sentiment-analysis/>

<sup>4</sup> For example, sarcasm.

in its Final Jeopardy round, giving “Toronto” as an answer to a category named “U.S. Cities.”<sup>1</sup>

Implementations of sentiment analysis algorithms often focus on the words contained in an input, as common words such as “good” and “bad” (among others) in most cases denote sentiment with reasonable consistency. These key words are therefore reliable indicators of the mood of the speaker or author, rendering these methods of sentiment analysis functional at a basic level at the least. As such, this project was based on word choice, with additions made such that some special cases such as negation statements and emphasis statements were taken care of as well.

## Purpose

The project was intended to further personal knowledge of the field of natural language processing and to explore the interactions of logical algorithms with something distinctly human, namely sentiment and its expression through words. It was an attempt to create a rough working model of a sentiment analysis algorithm based on knowledge of current algorithms and prior knowledge and to write this program from scratch.

## Process

### 1. Data Collection

As mentioned, book review training data was taken from Goodreads.com (with a program written in Python and with the help of BeautifulSoup<sup>4</sup>), a site chosen for its familiarity and for its main focus on

book reviews by a variety of users (above other sites that host book reviews such as the New York Times website and the subsection of Amazon that deals with book reviews). Goodreads, a free site, maintains a living database of published books (eBooks and different versions included) by ISBN number and allows users to rate and review books that they have read, and gives suggestions based on each user’s favorite books and genres. Though one can rate a book and not write a review and vice versa, many reviews on the site are accompanied by a one- to five-star rating.

The standard rating-and-review system helpfully associates a given set of words (the book review) with what is essentially a sentiment rating (the number of stars), though admittedly the association between number of stars and positivity or negativity is often inconsistent: different users have different perceptions as to the value of a particular number of stars assigned to a book. However, for the most part, one- and two-star reviews are generally negative, and four- to five-star reviews are generally found to be positive. Three-star reviews were found to be the most inconsistent rating<sup>2</sup>, and so the data collection program removed any reviews with a three-star rating for consideration.

Collected book reviews were all placed in a MySQL database with the review text as the primary key. Two other fields included were of the rating (i.e. number of

---

<sup>1</sup> An article about IBM Watson’s slip-up can be found on TIME Magazine’s site at <http://techland.time.com/2011/02/16/why-did-watson-think-toronto-is-a-u-s-city-on-jeopardy/>

---

<sup>2</sup> Some users write somewhat positive reviews (reasoning that three is more than half of five), some others write neutral reviews (as it is the middle value), and the rest write somewhat negative reviews (these users see one- to two-star ratings as taboo).

stars associated with the book review in question) and a binary value of whether the review had been used as practice data yet. As each review was used, it was not removed from the database for debugging and administrative purposes; the “used” value was just changed such that the review would not be accidentally recycled and mistakenly used again.

## 2. Method

The learning algorithm was based off of word choice of test data and associated each word in a given test book review with the review’s rating. Each word in the English language was initially assigned a default value of 50, which was arbitrarily set to be the neutral sentiment value. This dictionary of words and their constantly changing sentiment values were stored in a separate MySQL table. Bounds for these sentiment values were set at zero for the extreme negative case and 100 for the opposite, positive case.

A program was written that retrieved reviews and their associated ratings from the MySQL database to process. It first removed punctuation from each of the reviews, though it particularly replaced hyphens and slashes with spaces, as these punctuation markings are often associated with separating distinct words like in phrases such as “good-looking” and “pizza/hamburgers.” Periods were also replaced with spaces, as experience showed that an alarming number of users failed to place spaces after periods to separate sentences<sup>1</sup>.

The words in the reviews were then identified and separated based on spaces, and each word’s current sentiment value was retrieved from the dictionary MySQL

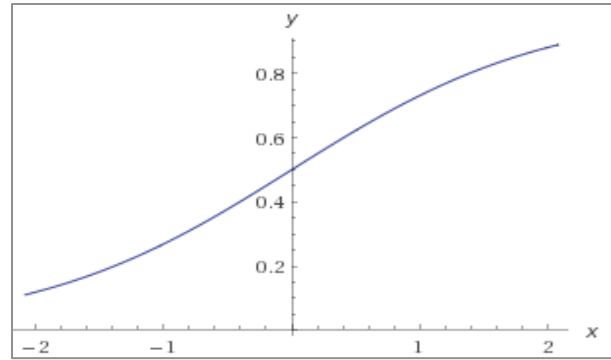


Figure 1. The sigmoid function as graphed by Wolfram Alpha.

table and then altered based on the rating associated with the review.

The changes to the value were dependent on two factors: most obviously the perceived sentiment of the review from which it came (negative for one- to two-stars, positive for four- to five-stars), but also on the previous (i.e. not-yet-altered) value of the word. The latter dependency dictated that values that were closer to the default value of 50 would be changed with more significant weighting than values that had been closer to the extremes. The reasoning here was to even out the weighting distributions for less common words, as heavier weighting at neutral values would help to boost the words’ values out of the neutral zone and into a categorization of either positive or negative.

The particular weighting method was broadly based off of the sigmoid function (Figure 1):

$$S(x) = \frac{1}{1 + e^{-x}}$$

A few tweaks made to coefficients and with an absolute value adjustment. The final variant is as follows, where the variable  $x$  denotes the original sentiment value of the word (Figure 2):

$$S_v(x) = \left[ \frac{10}{1 + e^{0.1 * |x - 50|}} \right]$$

<sup>1</sup> One wonders whether these people are qualified to write said book reviews.

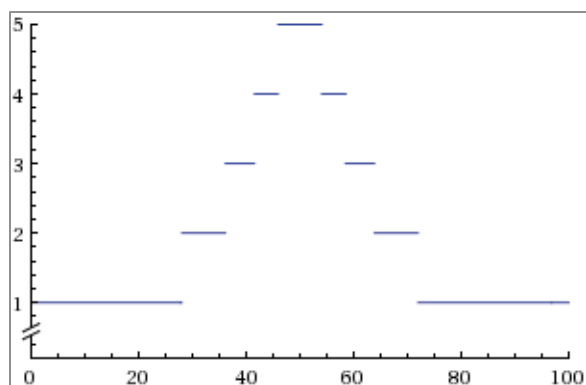


Figure 2. Graph of the implemented function.

With this function, words that have an initial value of 50 are altered for a base value of five sentiment value points, while words with initial values of 20 or 90 are altered by a base value of one. And, of course, values have been capped at the extremes such that a base change for values at zero and 100 is zero.

Shortly after the inception of this stage of the project, it was decided that short words of three letters or less, with a few exceptions<sup>1</sup> would be discarded. Shorter words such as “the,” “and,” and “or” are typically the more prevalent vocabulary in everyday use. As such, they essentially become the offal of reviews, as their weightings become skewed depending on how much of the test data was positive or negative<sup>2</sup>. Whenever the program came across these words, it would skip over them for consideration. They were also removed from the MySQL dictionary table as well.

After some initial testing or a prototype, it was determined that other, longer common words such as “their” and

<sup>1</sup> Notably, “bad” and “mad,” among others.

<sup>2</sup> Specifically, if the test data consisted of more positive reviews than negative, the common words would have sentiment values that are skewed positive as well, corrupting the results.

“where” were of similar prevalence as the shorter words mentioned above, and so they were also removed from the table and from consideration.

Phrases of special cases were taken care of in at least a basic way; negations such as “not good” and emphasis statements such as “very good” were recognized through additions to the algorithm and dealt with individually.

Phrases that contained negations, recognized through key words (most obviously with “not” but also with others like “didn’t” and “can’t”) resulted in flipped weightings. Under this procedure, for example, “not good” would register as negative instead of positive.

On the other hand, emphasis statements, which were recognized in a similar way but with different identifiers such as “very” and “extremely.” The algorithm increased weighting changes of these phrases and words to account for the more intense emotion that is presumed to accompany such phrases.

Periodically, the words with values greater than or less than 50 (default values were excluded) were exported along with their sentiment values to a .txt file, a format that was chosen for ease of digital transport and ease of reading from a program.

### 3. Front End

The front end of the program is hosted on the TJHSST server, as this project is part of the Computer Systems Lab program at TJHSST. The web page itself was written in PHP, while the program behind it runs in Python 3.

The front end presents the “face” of the project and the actual sentiment analysis algorithm that was developed over the year. It accepts a text input in a given text

**Book Review Sentiment Analysis**

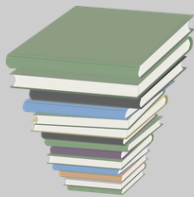
**Project Overview:**

This sentiment analysis program will take an input (preferably a book review, since that is what it has been trained on) and return a positivity or negativity rating depending on what the input text is. The scale used ranges from 0 to 100, with 0 being the most negative and 100 being the most positive; 50 is the neutral default value.

Currently, the program is in its final stages!

**Enter your text to Analyze:**

This was a good book



**Results:**

```

*****
71.0
That's positive. :)
*****

```

Or you can go [back](#).

Figure 3. The front-end display of the final product.

box and processes the input in a similar way to the learning algorithm. As above, punctuation is removed and words are separated using spaces as indicators. The program also implements the 100-point scale for sentiment value, with similar checks for negation and emphasis statements.

The program keeps what is essentially a running total of the sentiment values of each word in the input, where applicable. After tallying up the relevant sentiment values of each applicable word in the input, found by referencing the most recent dictionary .txt file that results from the learning program, the front-end program returns an average sentiment value.

It, like the learning program, does not take into account the aforementioned short and common words, and does not count them in its total input word count

in computing the average sentiment value of the text given.

Through the front-end program's procedure of referencing each separate word in the given dictionary for sentiment values, when an unrecognized word is encountered it is discarded. This has a fortunate side effect of removing nonsense words or other extraneous characters<sup>1</sup> from consideration.

## Results

The algorithm works for simple text and is able to determine sentiment as long as input does not implement complex sentence or grammatical structures. Some

<sup>1</sup> These take the form of such text as "lol" (not an actual word) and "dasfg" (presumed optimistically to be an accidental finger spasm), along with foreign words, which have not been accounted for, and numbers.

common words that were not caught continue to skew the input sentiment weightings, though this is expected to improve with continued removal.

If straightforward sentiment is expressed with a minimum of complex or obscure and ambiguous language, the program performs respectably.

## **Conclusion**

Due to its nearly exclusive focus on word choice, the algorithm does, admittedly, have noticeable downfalls. Most obviously is the effect of a limited vocabulary; as the project only spanned the course of a few months, a full dictionary with entirely reliable values could not be compiled, and so less common words may not be accounted for. The weighting of names also tends to skew results due to the appearance of character names in test reviews.

However, the algorithm works reasonably well and is expected to increase in accuracy given more learning data and some removals of problematic words and names.

Recognition of more complex sentence structures may require more effort and may prove to be difficult due to the various nuances in the English language, but if implemented could vastly improve sentiment analysis when coupled with the already existing algorithm that analyses word choice.



## References

Castillo, Michelle. "Why Did Watson Think Toronto is a U.S. City on Jeopardy?" *TIME*. 16 Feb. 2011. Web. 3 Jun. 2014.

<http://techland.time.com/2011/02/16/why-did-watson-think-toronto-is-a-u-s-city-on-jeopardy/>

DeLua, Julianna. "Big Data Meets Sentiment Analysis!" *The Informatica Blog*. 27 Jun. 2011. Web.

<http://blogs.informatica.com/perspectives/2011/06/27/big-data-meets-sentiment-analysis/>

Richardson, Leonard. Beautiful Soup Documentation. Web resource.

<http://www.crummy.com/software/BeautifulSoup/>