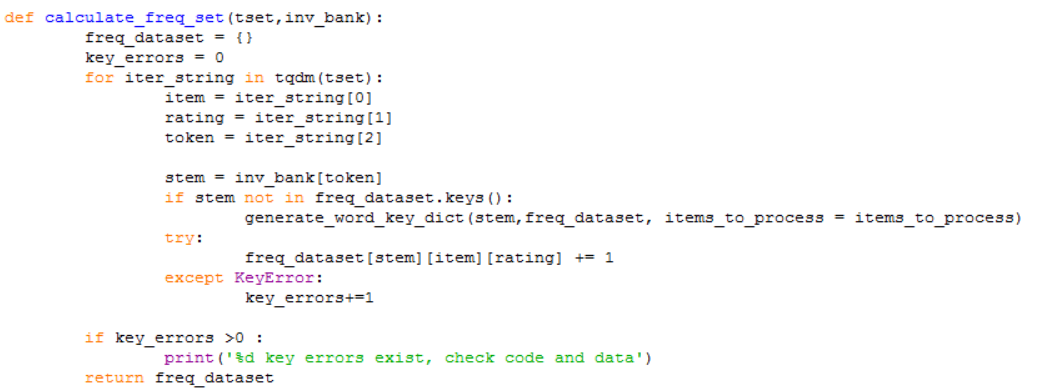
**Sentiment Detection**

In determining the sentiment of a word, sentences are first tokenized, tagged and stemmed. Tokenizing is done using NLTK’s TreebankTokenizer and stemmed with NLTK’s SnowballStemmer. Tagging is applied to tokens first to remove words that do not potentially hold sentiment; tags such as nouns, pronouns, modals and preposition were removed before stemming. The averaged perceptron tagger in NLTK is used without training.

Tokens are grouped into word stems and then the frequency is counted according to the type (reviewText and summary) and rating.



A score is then applied to determine the overall sentiment of the word, and 20 words with the heaviest positive and negative sentiments are tabulated.

Before determining the sentiment score of each word, certain heuristics were considered:

1. 5-star and 1-star ratings carry more weight than 4-star and 2-star respectively. (Score with respect to rating is not linear)
2. Although a word may appear equally for positive and negative reviews, it may still be calculated as a positive or negative word if there are far more of one type of review over the other.
3. Words that occur too frequently may not hold high significance.

Words are then scored according to their stems with the following equation :

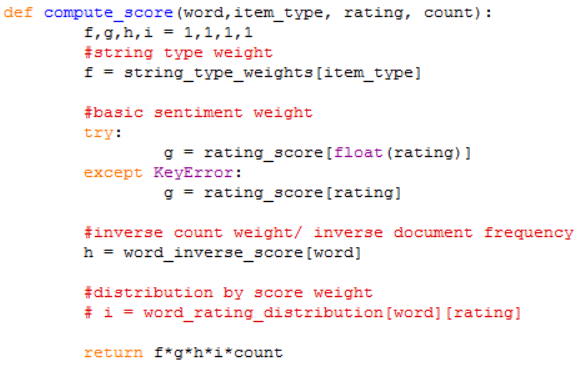
w = normalized weight with respect to rating

r = rating of review where the word is found

t = rating type weight ( ‘reviewText’ or ‘summary’)

f = term frequency

N = total number of words in dataset



**Equation Components**

**Wr**

The normalized weight given to a word depending on its rating. 2.5 for 5-star, 1.5 for 4-star, 0.5 for 3-star, -1.5 for 2-star and -2.5 for 1-star.

The weights are then normalized and adjusted to account for an unequal distribution of reviews, so that words with equal occurrence in both positive and negative sentiments will have a score close to 0.

A 3-star rating is given a nominal value of 0.5 as it is thought that they carry a slightly positive sentiment, also, giving a weight of 0 eliminates them from calculation.

The value is normalized as such :

r = rating

f r = frequency wrt. rating

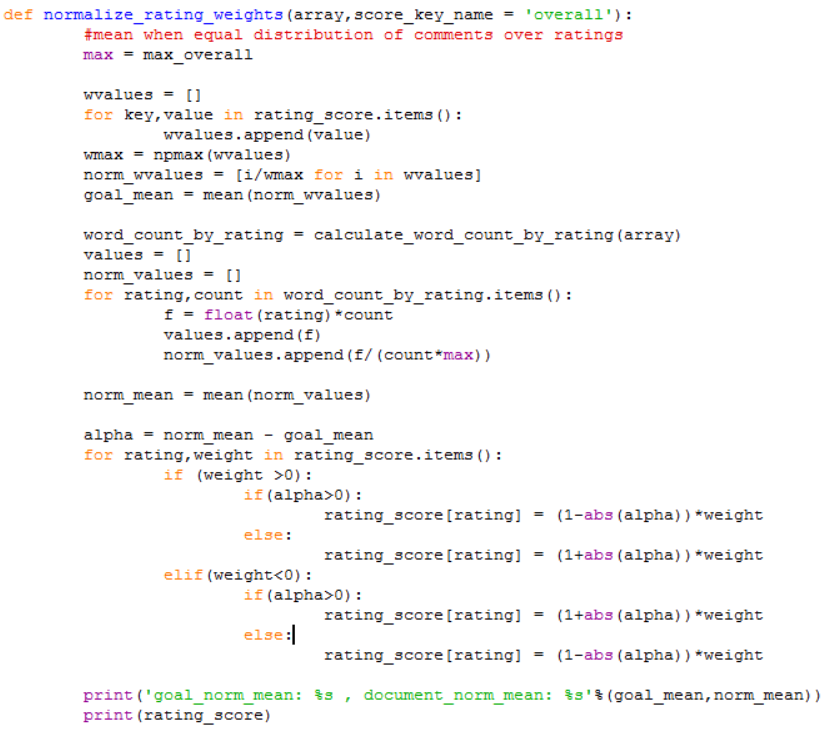
rmax = maximum rating

Given if a word occurs once across all ratings, the normalized mean score of this word should approximate to 0.04. The normalized mean for words in the whole dataset is calculated, and the original weights is adjusted.

For this dataset, the normalized weight obtained was 5-star = 1.1, 4-star = 0.66, 3-star = 0.22, 2-star = -2.34, 1-star = -3.9



As negative sentiments are weighted more heavily than positive sentiments, it can be deduced that there are much more words In positive ratings as compared to negative ones.

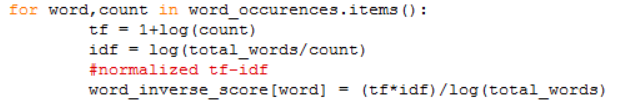


**t**

As the summary is the first part of the review that is read, it makes sense that reviewers would put words that relay the intended sentiment here. Hence, words found in the summary are given a higher weight.

We gave a weight of 2.5 for words in the summary and 1.0 for words in the review text.

This is a form of tf-idf used to reduce the relevance of overly-common words. The weight is normalized by ln(N) to prevent it from being too large.



The resulting stems obtained are listed here :

Top 20 Positive Sentiment Words

#1 : great value = 364040

#2 : good value = 171847

#3 : love value = 171285

#4 : use value = 164739

#5 : charg value = 123026

#6 : nice value = 118705

#7 : easi value = 84906.7

#8 : best value = 82853.4

#9 : need value = 74174.1

#10 : perfect value = 73076

#11 : littl value = 68379

#12 : look value = 65356.3

#13 : protect value = 63209.6

#14 : usb value = 60949.5

#15 : recommend value = 55427.3

#16 : excel value = 54394.5

#17 : work value = 53387.5

#18 : awesom value = 41684.5

#19 : keep value = 39884.9

#20 : want value = 39181.3

Top 20 Negative Sentiment Words

#1 : poor value = -31183.3

#2 : return value = -29984.2

#3 : disappoint value = -22437.1

#4 : broke value = -20482.7

#5 : cheap value = -19067.4

#6 : horribl value = -18537

#7 : terribl value = -18366.8

#8 : bad value = -18289.8

#9 : wast value = -15643.3

#10 : stop value = -13372.2

#11 : pay value = -10794

#12 : worst value = -10390.2

#13 : send value = -8922.71

#14 : defect value = -8774.63

#15 : useless value = -7904.36

#16 : fell value = -6386.65

#17 : sent value = -6064.02

#18 : broken value = -5970.47

#19 : fail value = -5752.77

#20 : wors value = -4903.03

While most of the negative sentiment words made logical sense. Some of the positive sentiment words did not. Words like ‘use’ was derived from a mix of neutral words such as ‘using’ or ‘used’ and positive words such as ‘useful’. Despite the removal of nouns and stopwords, words such as ‘usb’ made it through the filters and obtained a high score, without training the perceptron tagger used for tagging, the tagger could not recognise the word and tagged it as an adjective. (‘JJ’)

A list of stems in the top 20 list and their constituent tokens can be found in results.txt in the source code.