Hate Speech on Twitter – Dissecting Sentiment Metrics

Amir ElTabakh

In the data transformation process, nine aggregate columns were added, seven of which are in the context of sentiment analysis. These columns are titled polarity, subjectivity, sentiment, neg, neu, pos and compound. In this document I will provide a high and low level dissection of each of these seven metrics.

What is Sentiment Analysis

Sentiment Analysis, or Opinion Mining, is a sub-field of Natural Language Processing (NLP) that tries to identify and extract opinions within a given text. The aim of sentiment analysis is to gauge the attitude, sentiments, evaluations, attitudes and emotions of a speaker/writer based on the computational treatment of subjectivity in a text.

Why is sentiment analysis so important

Businesses today are heavily dependent on data. Majority of this data, however, is unstructured text coming from sources like emails, chats, social media, surveys, articles, and documents. The micro-blogging content coming from Twitter and Facebook poses serious challenges, not only because of the amount of data involved, but also because of the kind of language used in them to express sentiments, i.e., short forms, memes and emoticons.

Sifting through huge volumes of this text data is difficult as well as time-consuming. Also, it requires a great deal of expertise and resources to analyze all of that. Not an easy task, in short. Sentiment Analysis is also useful for practitioners and researchers, especially in fields like sociology, marketing, advertising, psychology, economics, and political science, which rely a lot on human-computer interaction data.

Sentiment Analysis enables companies to make sense out of data by being able to automate this entire process! Thus, they can elicit vital insights from a vast unstructured dataset without having to manually indulge with it.

Polarity and subjectivity

We calculate the sentiment of a text through the python TextBlob library, which provides us numeric values for polarity and subjectivity. The numeric value for polarity describes how much a text is negative or positive. Similarly, subjectivity describes how much a text is objective or subjective.

But how are these numeric values computed or where do these come from? The answer to the above question is that TextBlob uses a process defined in _text.py for the sentiment calculation. These lexicons are referred to in en-sentiment.xml. Looking at the lexicons file, we can see that it does not contain any stop words i.e., the, he, have, etc. because they do not have any sentiment. Other than that, each word is defined in the lexicon file with their part of speech (POS), polarity, subjectivity, intensity, and confidence.

When calculating a sentiment for a single word, TextBlob uses the "averaging" technique that is applied on values of polarity to compute a polarity score for a single word and hence similar operation applies to every single word and we get a combined polarity for longer texts. TextBlob also handles negations and polarity is multiplied by -0.5 and doesn't affect subjectivity.

Now there is a very interesting thing about TextBlob that it handles the modifiers also known as intensifiers which intensifies the meaning of text according to its pattern. Whenever a modifier word is used, TextBlob will ignore polarity and subjectivity and just use intensity to compute the sentiment of the text. Another interesting fact about TextBlob is that when a negation combines with modifiers, the inverse intensity of the modifier enters for subjectivity and polarity.

Here is an example of a modifier. Consider the string "very great." Recognizing "very" as a modifier word, TextBlob will ignore polarity and subjectivity and just use intensity to modify the following word, "great".

TextBlob will ignore one-letter words in its sentiment phrases, And TextBlob will ignore words it doesn't know anything about. TextBlob goes along finding words and phrases it can assign polarity and subjectivity to, and it averages them all together for longer text. And while I'm being a little critical, and such a system of coded rules is in some ways the antithesis of machine learning, it is still a neat system and I think I'd be hard-pressed to code up a better such solution.

VADER scoring for sentiment, neg, neu, pos and compound

VADER (Valence Aware Dictionary for Sentiment Reasoning) is a lexicon and rule-based sentiment analysis tool that is ***specifically attuned to sentiments expressed in social media***. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labelled according to their semantic orientation as either positive or negative. It is available in the NLTK package and can be applied directly to unlabeled text data.

VADER sentimental analysis relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores. The sentiment score of a text can be obtained by summing up the intensity of each word in the text. For example- Words like *'love', 'enjoy', 'happy', 'like'* all convey a positive sentiment. Also, VADER is intelligent enough to understand the basic context of these words, such as *"did not love"* as a negative statement. It also understands the emphasis of capitalization and punctuation, such as *"ENJOY".*

VADER has been found to be quite successful when dealing with social media texts, NY Times editorials, movie reviews, and product reviews. This is because VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

The Sentiment metric takes the max between neg and positive. If neg is the max between the two, then the value in the Sentiment column is Negative, and the same if pos is the max between the two.

The SentimentIntensityAnalyzer module from the vaderSentiment library calculates how negative (neg), positive (pos) and neutral (neu) a string or body of text is. All three values are in the range [0, 1] inclusive. 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.

That's the high-level overview of these metrics, now how does VADER calculate these values?
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 to analyze the context & presentation of how sentiment is conveyed or embedded in rhetoric for a given sentence. For example, different writing styles may embed strongly positive or negative sentiment within varying proportions of neutral text -- i.e., some writing styles may reflect a penchant for strongly flavored rhetoric, whereas other styles may use a great deal of neutral text while still conveying a similar overall (compound) sentiment. As another example: researchers analyzing information presentation in journalistic or editorial news might desire to establish whether the proportions of text (associated with a topic or named entity, for example) are balanced with similar amounts of positively and negatively framed text versus being "biased" towards one polarity or the other for the topic/entity.

Importantly, the proportions represent the "raw categorization" of each lexical item (e.g., words, emoticons/emojis, or initialisms) into positive, negative, or neutral classes; they do not account for the VADER rule-based enhancements such as word-order sensitivity for sentiment-laden multi-word phrases, degree modifiers, word-shape amplifiers, punctuation amplifiers, negation polarity switches, or contrastive conjunction sensitivity.

VADER analyses sentiments primarily based on certain key points:

- **Punctuation: T**he use of an exclamation mark(!), increases the magnitude of the intensity without modifying the semantic orientation. For example, "The food here is good!" is more intense than "The food here is good." and an increase in the number of (!), increases the magnitude accordingly.
- **Capitalization**: Using upper case letters to emphasize a sentiment-relevant word in the presence of other non-capitalized words, increases the magnitude of the sentiment intensity. For example, "The food here is GREAT!" conveys more intensity than "The food here is great!"

- **Degree modifiers**: Also called intensifiers, they impact the sentiment intensity by either increasing or decreasing the intensity. For example, “The service here is extremely good” is more intense than “The service here is good”, whereas “The service here is marginally good” reduces the intensity.
- **Conjunctions**: Use of conjunctions like “but” signals a shift in sentiment polarity, with the sentiment of the text following the conjunction being dominant. “The food here is great, but the service is horrible” has mixed sentiment, with the latter half dictating the overall rating.
- **Preceding Tri-gram**: By examining the tri-gram preceding a sentiment-laden lexical feature, we catch nearly 90% of cases where negation flips the polarity of the text. A negated sentence would be “The food here isn’t really all that great”.
- **Emojis, Slangs, and Emoticons**: VADER performs very well with emojis, slangs, and acronyms in sentences.