MEKELLE UNIVERSITY -MIT

Facebook comment sentiment analysis-project-Documentation

Name **id**

Zena g/hiwot **mit/ur/071/09**

Submitted to instructor:

Widassie Gerezgiher

Sentiment Analysis of Facebook Comments

Abstract

While recent NLP-based sentiment analysis has centered around Twitter and product/service reviews, we believe it is possible to more accurately classify the emotion in Facebook status messages due to their nature. Facebook status messages are more succinct than reviews, and are easier to classify than tweets because their ability to contain more characters allows for better writing and a more accurate portrayal of emotions.I analyze the suitability of various approaches to Facebook status messages by comparing the performance of partS-of-speech ("POS") tagging, and POS data. I classify both binary and multi-class sentiment labeling. In both cases, a MaxEnt classifier augmented with POS data performs the best, achieving an average binary classification F1 score of approximately 80% and an average multi-class F1 score of approximately 80%.

Introduction/Background

With the transition from forum- and blog-based Internet communication among users to social networking sites such as Facebook and Twitter, there exists new opportunity for improved information mining via NLP sentiment analysis. Prior success at such analysis has been elusive due to the inherent difficulty in extracting a singleton sentiment label from long passages, where fluctuations over the document length make classification difficult. While sentiment analysis of Twitter data has surged in recent years, it too is problematic for the opposite reason: tweets are limited to 140 characters. Also, tweets often use heavy abbreviation and are more likely to be fragmented expressions, making it difficult to use a part-of-speech ("POS") tagger. This project focuses on classifying sentiment of Facebook status updates ("status updates") using binary and multi-class labels. Facebook makes a distinction between a Facebook user's status update, versus links to a news article or other source of information, versus comments that are a response to another Facebook user. Unlike tweets, status updates can use up to 420 characters. Thus, status updates more often are written in mostly sentence-like structures that can benefit from POS analysis. Why perform sentiment analysis on

Facebook status updates? According to a January 2010 article on InsideFacebook.com, users spent nearly 7 hours per person on Facebook in December 2009, far higher than the other top 10 parent companies on the Internet. From a marketing standpoint, understanding user sentiment as it relates to a topic of interest clearly allows more effective ad-targeting. If a user is trending positively or negatively about health care reform, appropriate political party ads might appear sympathetic to a user's viewpoint. Similarly, a user that tends to write playful status messages might be shown ads for a local comedy club. Beyond the obvious marketing appeal, however, is a more sociologically compelling rationale: modeling the ebb and flow of consumer sentiment across a broad swath of topics, hierarchically encapsulated by user, locally-defined user communities, regionally, nationally, and globally. To our knowledge, there exists no prior work that focuses exclusively on sentiment analysis of Facebook status messages (O'Connor, Balasubramanyan, Routledge, Smith)'s was the first paper I read that shared a similar vision of NLP work of providing insight into public opinion by modeling sentiment within various societal clusters. Importantly, however, their work performs binary labeling against a pre-defined corpus of known positive/negative words. My work aims to learn sentiment non-deterministically, training on nothing more than the sentence and a label for the sentence as a whole.

Data Collection

implementation of batch data processing makes sense in the case of high volumes data. Firstly, i chose a topic, which is popular . For each post, using setting in facebook I download all my comments but since it is Amharic based language it can not produce sentiment analysis and using Facebook graph api I download the kindle's facebook comment(kindle is most well known amazon based ebook reader), all comments have been collected during the first 30000 s. Data is stored in flat table format (e.g. text file) which is easy to save in distributed file system. The header of text file contains the following columns: [Comment] The chosen topic is "FACEBOOK COMMENT ON KINDLE", which is popular recently. Almost data will be received from Amazon Platform Facebook.and I produce 2 files in the form of csv and Plain Text(.txt).

After collecting the data, the next step involved preprocessing the raw Facebook data, converting it into labeled data and breaking it up into training and test sets. For this project, I use naïve bayes classifier. Some massaging of the data between stages was required. my high-level sequence involved the following:

1) Raw data collection

2) Sentiment labeling

3) Transform into train/test sets for classifier-only tests

4) Transform into train/test sets for

5) Transform into train/test sets for POS

6) Transform train/test sets for final classification by classifier

7) Adjust classifier, and/or POS features and repeat until best model found

PREPROCESSING STEPS

Adapting a strategy used by (O'Connor, Balasubramanyan, Routledge, Smith),I labeled status messages by filtering for known emoticon equivalence classes, as defined by the Wikipedia article, List of Emoticons, at http://en.wikipedia.org/wiki/List_of_emoticons. i chose to filter for 14 categories: angry, evil, frown, rock-on, shock, smiley, wink, angel, blush, fail, neutral, shades, skeptical, and tongue. To prevent training on the very emoticons upon which we based 3 our labeling, we then stripped these emoticons out of the data. For binary classification, we chose to classify into positive and negative sentiment labels. These labels are represented by:

POSITIVE: smiley, wink, tongue, angel, shades, blush, rockon

NEGATIVE: frown, shock, skeptical, evil, angry, fail

When I implement my project I pass through many preprocessing steps those are given below:

- Tokenize Words and Sentences with NLTK
- Stemming and Lemmatization with Python NLTK
- Parts of Speech(POS) Tagging in Python's NLTK library

- Removing stop words with NLTK in Python
- Classification Workflow
- SENTIMENT INTENSITY ANALYZER

Tokenize Words and Sentences with NLTK

Tokenization is the process by which big quantity of text is divided into smaller parts called **tokens**.

Natural language processing is used for building applications such as Text classification, intelligent chatbot, sentimental analysis, language translation, etc. It becomes vital to understand the pattern in the text to achieve the above-stated purpose. **These tokens are very useful for finding such patterns as well as is considered as a base step for stemming and lemmatization.**

Natural Language toolkit has very important module **tokenize** which further comprises of sub-modules:

1. word tokenize
2. sentence tokenize

Tokenization of words

I use the method **word_tokenize()** to split a sentence into words. The output of word tokenization can be converted to Data Frame for better text understanding in machine learning applications. It can also be provided as input for further text cleaning steps such as punctuation removal, numeric character removal or stemming. Machine learning models need numeric data to be trained and make a prediction. Word tokenization becomes a crucial part of the text (string) to numeric data conversion.

Tokenization of Sentences

When I tokenize sentence of the facebook comment I use sent.tokenize function to tokenize the whole sentence Sub-module available for the above is sent_tokenize. An obvious question in your mind would be **why sentence**

tokenization is needed when we have the option of word tokenization. Imagine you need to count average words per sentence, how you will calculate? For accomplishing such a task, you need both sentence tokenization as well as words to calculate the ratio. Such output serves as an important feature for machine training as the answer would be numeric.

Stemming and Lemmatization with Python NLTK

Stemming

Stemming is a kind of normalization for words. Normalization is a technique where a set of words in a sentence are converted into a sequence to shorten its lookup. The words which have the same meaning but have some variation according to the context or sentence are normalized.

In another word, there is one root word, but there are many variations of the same words. For example, the root word is "eat" and it's variations are "eats, eating, eaten and like so". In the same way, with the help of Stemming, we can find the root word of any variations.in stem I try to collect the whole data from csv/txt and I stem with actual function porter_stemmer.stem(w).

Lemmatization

Lemmatization is the algorithmic process of finding the lemma of a word depending on their meaning. Lemmatization usually refers to the morphological analysis of words, which aims to remove inflectional endings. It helps in returning the base or dictionary form of a word, which is known as the lemma. The NLTK Lemmatization method is based on WorldNet's built-in morph function. Text preprocessing includes both stemming as well as lemmatization. Many people find the two terms confusing. Some treat these as same, but there is a difference between these both. Lemmatization is preferred over the former.in my project I lemmatize the whole scentence through for loop with function WordNetLemmatizer().

Parts of Speech(POS) Tagging in Python's NLTK library

python's NLTK library features a robust sentence tokenizer and POS tagger. Python has a native tokenizer, the `.split()` function, which you can pass a separator and it will split the string that the function is called on on that separator. The NLTK tokenizer is more robust. It tokenizes a sentence into words and punctuation.
Based on previous experience, I expected n-gram word features to be the most significant such features, but also felt word-shape features would play a bigger role in this project, where sentiment polarity is often expressed by "!!", "??", "?!?!", "……", and so on. To this end, consistent across all my experiments, I split words in based on the regular expression, [,] (comma, space). i expected that the degree to which i needed to tweak features between the binary and muli-label classification tasks would indicate whether Facebook messages truly encoded sentiment such that NLP models scale well to multiple sentiment labels.after lemmatize my facebook comment I use nltk.pos_tag() function to pos tag the whole comment.

Removing stop words with NLTK in Python

The process of converting data to something a computer can understand is referred to as **pre-processing.** One of the major forms of pre-processing is to filter out useless data. In natural language processing, useless words (data), are referred to as stop words.
Stop Words: A stop word is a commonly used word (such as "the", "a", "an", "in") that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.
Since the comment may have unnessesary words I use the function stop_words() to remove words from becoming tokenize and analyzed.

Classification Workflow

Whenever i perform classification, the first step is to understand the problem and identify potential features and label. Features are those characteristics or attributes which affect the results of the label. The classification has two phases, a learning phase, and the evaluation phase. In the learning phase, classifier trains its model on a given dataset and in the evaluation phase, it tests the classifier performance. Performance is evaluated on the basis of various parameters such as accuracy, error, precision, and recall.in this category I use naïve bayse method of classification.and I catograize my whole data into sujective and objective.

SENTIMENT INTENSITY ANALYZER

What is Sentiment Analysis?

Sentiment Analysis, or **Opinion Mining,** is a sub-field of 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 sentiment analysis?

- **Business:** In marketing field companies use it to develop their strategies, to understand customers' feelings towards products or brand, how people respond to their campaigns or product launches and why consumers don't buy some products.
- **Politics:** In the political field, it is used to keep track of political view, to detect consistency and inconsistency between statements and actions at the government level. It can be used to predict election results as well! .
- **Public Actions:** Sentiment analysis also is used to monitor and analyse social phenomena, for the spotting of potentially dangerous situations and determining the general mood of the blogosphere.

VADER Sentiment Analysis

I used VADER Sentiment Analysis to analyze my facebook **comments.**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***. VADER uses a combination of A Sentiment lexikon is a list of lexical features (e.g., words) which are generally labelled according to their semantic orientation as either positive or negative.**

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**.

Advantages of using VADER

VADER has a lot of advantages over traditional methods of Sentiment Analysis, including:

- It works exceedingly well on social media type text, yet readily generalizes to multiple domains
- It **doesn't require any training data** but is constructed from a generalizable, valence-based, human-curated gold standard sentiment lexicon
- It is fast enough to be used online with streaming data, and
- It does not severely suffer from a speed-performance tradeoff. That is why I used to analyze my facebook comments.

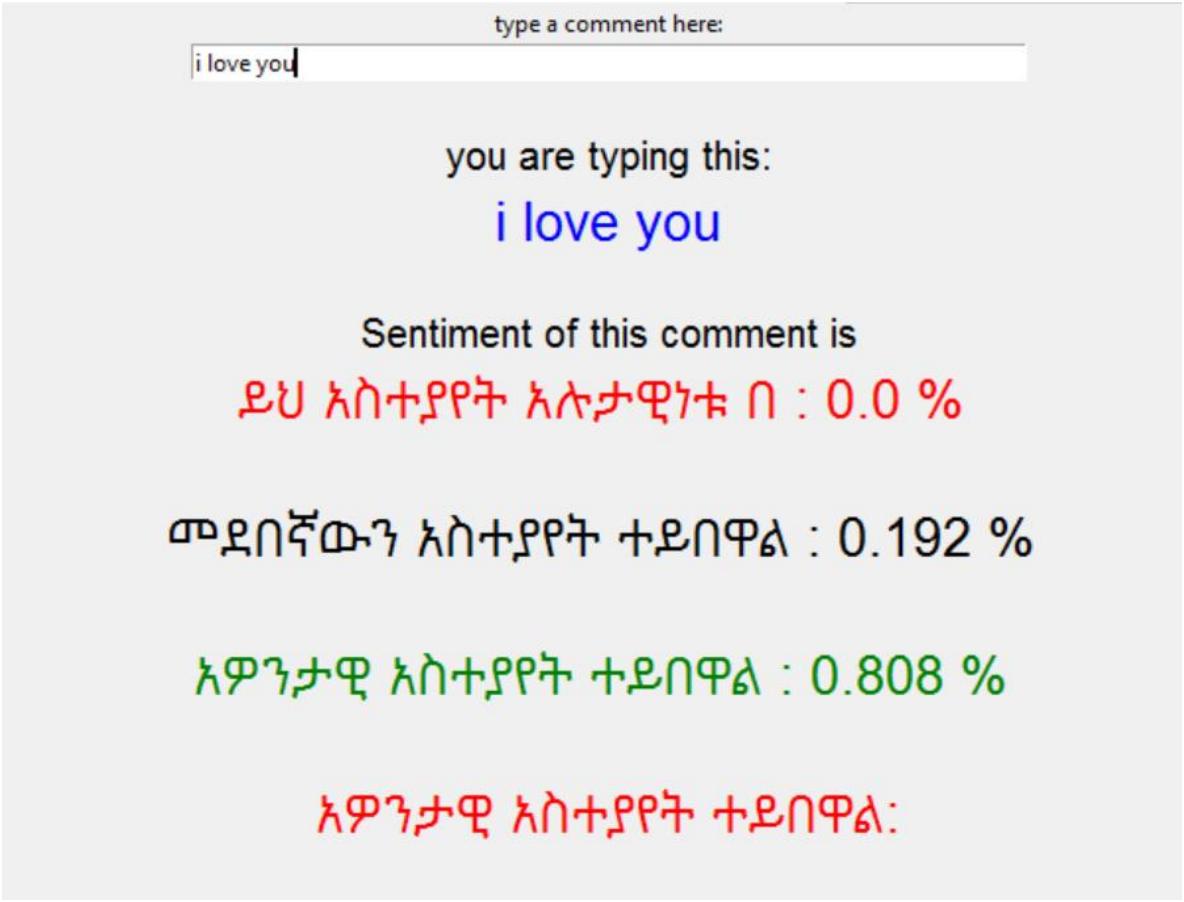
Architecture

In my project

I use both user input and text to analyze my facebook comment

, I installed featured libraries to show the user interface representation of the comment.

Here is a sample comment:



Requirements:

The project requires installed packages:

- NLTK - Natural Language Toolkit is a leading platform for building Python programs to work with human language data

- facebook-sdk - Python SDK for Facebook's Graph API
- matplotlib - Matplotlib is a Python 2D plotting library
- scikit-learn - Machine Learning library in Python
- pandas - an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming

References

Dwergs. What's the maximum length of a Facebook status update? February 19, 2009. http://reface.me/status-updates/whats-the-maximum-length-of-a-facebook-status-update/ Eldon, Eric. ComScore, Quantcast, Compete, Nielsen Show a Strong December for Facebook Traffic in the US. January 22, 2010. http://www.insidefacebook.com/2010/01/22/comscore-quantcast-compete-show-a-strongdecember-for-facebook-traffic-in-the-us/ Go, Alec; Huang, Lei; Bhayani, Richa. Twitter Sentiment Analysis. Stanford University, Spring 2009, CS224N Natural Language Processing, Professor Chris Manning. http://nlp.stanford.edu/courses/cs224n/2009/fp/3.pdf Harris, Jonathan; Kamvar, Sep. We Feel Fine. August 2005. http://www.wefeelfine.org O'Connor, Brendan. Balasubramanyan, Ramnath; Routledge, Bryan R.; Smith, Noah A. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. Carnegie Mellon University, 2010. http://anyall.org/oconnor_balasubramanyan_routledge_smith.icwsm2010.tweets_to_polls .pdf Pang, Bo; Lee, Lillian. Opinion Mining and Sentiment Analysis. Cornell University, 2008. http://www.cs.cornell.edu/home/llee/omsa/omsa-published.pdf Ramage, Daniel; Dumais, Susan; Liebling, Dan. Characterizing Microblogs with Topic Models. Stanford University. 2010. http://nlp.stanford.edu/pubs/twitter-icwsm10.pdf Ramage, Daniel; Hall, David; Nallapati, Ramesh; Manning, Christopher D. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. Stanford University, 2009. http://www.stanford.edu/~dramage/papers/llda-emnlp09.pdf