Image Sentiment Analysis

Christopher Graham

CKME136: Winter 2016

# Introduction

Social media has proven to be a rich field for making sense of popular opinion on a number of different topics. From companies trying to understand how their brand is being received, to political actors striving to get a reading on popular opinion, Twitter mining has become an invaluable tool.

Much of this work has focused on text-based sentiment analysis. As a result, text-based sentiment analysis in social media is a fairly well-evolved area of machine learning.

But focusing exclusively on textual sentiment analysis misses the fact that social media is increasingly image-based. With image-centric platforms like Instagram and Snapchat becoming increasingly important, it is foolish to assume that text analysis alone can provide an accurate indicator of social media sentiment on any particular topic.

Unfortunately, much of the work in image analysis has focused on object identification, rather than sentiment determination. And much of the work that has been done on image sentiment has taken a hand-crafted, rather than machine-learning approach to sentiment ascription. Additionally, image sentiment analysis has been hampered by the fact that it is difficult to assemble sufficiently robust data image data sets that are labeled with reliable sentiments.

In this project, I will attempt to determine whether it is possible to build an effective predictor of image sentiment in by using text-based sentiment analysis can provide a ground-truth to effectively develop an image classifier. To do this, I propose to first classify Twitter-posted images based on textual clues. Using these ratings, I will then develop a Neural-Network based model to classify image sentiment.

The predictive ability of this model will be tested against images that have been had sentiment scores ascribed by crowd-sourcing. The model will be compared against benchmarks, and against a model derived solely from images whose sentiment has been crowd-sourced.

# Literature Review

There are a couple of different categories of literature that need to be reviewed to ensure this project is properly grounded in best practices. These are set out systematically below. See Bibliography at the end for full citations.

## Image Sentiment Analysis

Image sentiment analysis is a relative latecomer to the machine learning game. Until relatively lately, much of the effort in attempting to derive image sentiment have focused on hand-curated sets of attributes.

## Text Sentiment Analysis

My analysis here is a bit more succinct as this project is not about creating a text sentiment classifier, so much as it is about using text sentiment classification tools to establish a probable ground truth category for images.

Text sentiment analysis has

# Dataset

The data used for the bulk of this analysis is derived from captured Twitter stream data. The data has been downloaded via Twitter’s public API. Methodology and code are detailed in the “Approach” section below. Specifically, from each tweet deemed relevant, the tweet’s id, text content and image reference are captured for further analysis.

The text of each downloaded and relevant tweet is used to determine a positive or negative sentiment for that tweet. Images associated with these tweets (accessed via public URL) are used to build the image classifier.

The final image classifier will be tested against new Twitter data, and more importantly against a set of images that have had sentiment ascribed to them by human scorers in a crowd-sourced methodology. This human-scored image data set is publicly available from the firm Crowdflower. The data set is available at <http://cdn2.hubspot.net/hub/346378/file-2650954351-csv/Sentiment-polarity-DFE.csv?t=1454540299275>, and the description is available at <http://www.crowdflower.com/data-for-everyone>.

# Approach

## Step 1: Pull Raw Twitter Data

This is a fairly straight-forward pull of streaming Twitter data using the GET STATUSES/SAMPLE command from Twitter’s public API. Because of the amount of data needed, this was done in a number of samples over a 2-week period, creating a number of different files for a total of 6,832,783 tweets downloaded.

Code used for this step is posted at <https://github.com/asterix135/CKME136/blob/master/Python_code/twitter_stream.py>. Please note that unlike the rest of the code in this project, this is written for Python2, and is designed to be called from the command line. The introductory docstring provides details.

A sample raw pull is posted at <https://github.com/asterix135/CKME136/blob/master/Data/output_jan24.txt>

**Note:** I had originally hoped to be able to pull responses to tweets, and to use the content of those responses as a way to better define tweet sentiment. Unfortunately, Twitter’s API limitations make this more or less impossible, so the project is progressing using only the content of an original tweet.

## Step 2: Subset to Original Tweets with Images

In order to effectively ascribe sentiment to an image in a tweet, we first need to ensure three things at a minimum:

1. the tweet is in English
2. the tweet contains an image
3. the tweet contains text

Additionally, it is useful to try and ensure that the tweet is original, so as to avoid inadvertently including multiple retweets of the same image and text.

To do this, the various files of pulled stream data were processed on the following basis:

1. the lang attribute for the tweet’s json representation has the value “en”
2. the tweet’s json representation contains a “text” attribute
3. the tweet’s json representation contains an “extended\_entities” attribute (indicative of the presence of an image)
4. the tweet’s text does not begin with “RT” (standard Twitter syntax for identifying a retweet)

In addition, to facilitate some of the later work, the text field was also pre-processed in the following way:

1. text was stripped of any URL, as this has no relevance to sentiment calculation
2. hashtags in the text were, where possible, split into words, using the default English-language dictionary supplied on OSX computers (also, I believe on Linux systems)

From the original json file, the following fields were then saved to a MySQL database:

* tweet\_id
* username (of the original poster)
* original text
* processed text
* image\_url
* timestamp when tweet was created

From the original Twitter pull, 106,955 records were selected: approximately 1.5% of the original twitter stream

Code for the main pre-processing is available at: <https://github.com/asterix135/CKME136/blob/master/Python_code/process_raw_tweets.py>

Code for the text pre-processing (called from the above routine) is available at:

<https://github.com/asterix135/CKME136/blob/master/Python_code/text_sentiment/split_hashtag.py>

SQL code to create the database is available at:

<https://github.com/asterix135/CKME136/blob/master/SQL_code/create_database.sql>

A sample of the pre-processed data is available (tab-delimited format) at:

<https://github.com/asterix135/CKME136/blob/master/Data/sample_sql_dump.txt>

**YET TO DO ON THIS STEP**

1. I am not sure that the 200,000+ word default OSX dictionary is optimal for hashtag splitting. It may be worth trying to find an approach that is based more on word likelihood. I have also not yet attempted to take camel casing into account, as this would likely result in better splitting.
2. A significant portion of the Twitter corpus consists of advertising. It would probably help to filter out some of these messages. Depending on available time, I may try to do this.

## Step 3: Ascribing Sentiment based on Twitter Text

At this step, I need to make a fundamental decision: whether to progress on the basis of classification or regression.

If I treat this as a classification problem, I will need to assign a category to each tweet: Positive, Neutral or Negative.

If I treat this as regression, I will need to assign a relative score to each tweet indicating its degree of positivity or negativity.

I have chosen to proceed on the basis of classification, as I think it will make it easier to assess success or failure.

As such, the main objective here is to identify three subsets of the downloaded and pre-processed tweets:

1. Tweets that clearly evince a positive sentiment from their text
2. Tweets that clearly evince a negative sentiment from their text
3. Tweets that clearly evince a neutral sentiment from their text

Because I am mainly concerned with finding tweets that clearly evince the sentiment ascribed to them, I need to take a fairly conservative approach to ascribing sentiment.

As such, I have made use of three different sentiment analysis techniques – as discussed in the literature review above: VADER, AFINN and the Hu-Liu lexicon. Each of these takes a somewhat different approach to ascribing sentiment. If all three approaches agreed on a tweet’s sentiment, the tweet was tagged as positive, neutral or negative. Other tweets were discarded.

As can be seen from the Pearson correlation statistics, these three methods are less correlated than one might expect at first glance. The implication is that we can be fairly sure that sentiment attested to by all three classifiers is accurate.

Correlation stats:

vader afinn huliu

vader 1.000000 0.520781 0.423340

afinn 0.520781 1.000000 0.664954

huliu 0.423340 0.664954 1.000000

This returns the following number of tweets with the indicated sentiment:

Positive sentiment: 13349

Negative sentiment: 4718

Neutral sentiment: 44806

The negative classified tweets are a bit short of the 5,000 minimum images recommended per category for “acceptable performance” in a Neural Net classifier by Goodfellow et al [1], p 19. I will probably add a few more tweets to ensure the negative category exceeds this threshold.

Code for sentiment ascription is available at: <https://github.com/asterix135/CKME136/blob/master/Python_code/text_sentiment/compare_sentiments.py>

**YET TO DO ON THIS STEP**

The implementations of the Hu/Liu lexicon and the AFINN lexicon are too simplistic, and need to be enhanced.

## Step 4: Build Image Classifier

I am currently working on this part, so the process here is not as detailed, nor do I have as much code or data to show. My anticipation of how this will progress is as follows:

1. Standardize image sizes for ease of comparison using various Python libraries. It appears this will be at about 400x400 pixels, although perhaps smaller
2. Reduce dimensionality of the images – likely through PCA or Lasso methodologies
3. Build and refine a model. My intention is to use a Convolutional Neural Network approach, though I may also compare this with a few other methodologies, depending on time.

## Step 5: Testing

The difficulty with testing this model is that we are not starting with images whose sentiment we can be sure of. Therefore, testing will consist of 2 separate steps:

1. Testing against twitter images. For this, I will assess performance against a test set of images whose sentiment has been ascribed in the same way as my training set
2. Testing against images with human-scored sentiment. As noted in the “Data” section, Crowdflower has made available a set of images who have had sentiment ascribed to them by human raters. This is similar to the testing approach taken by Wang et al (although they commissioned their own set of scored images).

Base success will be assessed as results that are better than chance, although results will also be assessed against previously-reported sentiment classifiers.

## Bibliography

[1] Ian Goodfellow, Yoshua Benjino, and Aaron Courville, *Deep Learning*, Unpublished: Book in preparation for MIT Press, <http://www.deeplearningbook.org>, 2016

[2] Minqing Hu and Bing Liu, *Mining and Summarizing Customer Reviews*,

|  |  |
| --- | --- |
|  |  |
|  | ; Proceedings of the ACM SIGKDD International Conference on Knowledge |
|  | ; Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, |
|  | ; Washington, USA |