Image Sentiment Analysis

Christopher Graham

CKME136: Winter 2016

Project github repo: https://github.com/asterix135/CKME136

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 attempted to determine whether it is possible to build an effective predictor of image sentiment by using text-based sentiment analysis – specifically, whether text-based sentiment analysis can provide a ground-truth to effectively develop an image classifier. To do this, I classified Twitter-posted images based on textual clues. Using these ratings, I developed a Neural-Network based model to classify image sentiment.

The predictive ability of this model was tested against images that have been had sentiment scores ascribed by crowd sourcing.

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 has focused on hand-curated sets of attributes. Of late, researchers have made use of a few different tools – including Neural Networks and text analysis to impute sentiment value.

<u>You et al, “Robust Image Sentiment Analysis” [2]</u>

You et al address much the same problem I am attempting to address, and highlight the difficult nature of image sentiment classification, due in large part to the abstraction and subjective nature of sentiment, especially when applied to an image out of context. Specifically of note is the fact that they highlight the superiority of Deep Learning/Neural Networks to feature-based approaches in computer vision. Further, they show that using labels derived from social media (Flickr for their paper) can produce an effective, generalizable classifier.

<u>Cao et al, “An Adaboost Back-Propagation Neural Network for Automated Image Sentiment Classification” [3]</u>

Cao et al focus on a specific Neural Network algorithm to classify images based on colour content (including, hue, saturation, etc.). Their model is trained and tested exclusively on a small set of landscape images, and it appears that their assessment of image sentiment is based on colour data as well. This approach seems to take less advantage of some of the recent developments in deep learning algorithms to understand visual data. The main takeaway is that combining Neural Net approaches with other machine learning algorithms can boost image classification.

<u>Siersdorfer and Hare, “Analyzing and Predicting Sentiment of Images on the Social Web” [4]</u>

This is one of the (relatively) earlier attempts to classify image sentiment, using a Support Vector Machine (SVM) for classification. The authors show the value of using text data to automatically classify images. They use the Senti-Net lexicon to classify image sentiment, taking a fairly conservative (and crude) approach to text classification, achieving 70% accuracy in classification (Full details on SentiNet can be found in Borth et al [12]). The analysis was carried out on a Flickr dataset of nearly 600,000 images. Notably, their classification is based on pre-determined criteria of colour and a “bag of visual terms” approach, finding that colour was key in determining sentiment. The key take-away for this project is that textual data can be an effective way to classify image sentiment.

Matthews et al, "SentiCap: Generating Image Descriptions with Sentiments" [6]

This study again highlights the superiority of NeuralNet approaches to issues of computer sight. In this case, the authors put together a dataset of images that are captioned with emotionally-evocative adjectives and descriptive nouns, with the captions provided by crowd-sourced workers. They then use a combination of Convolutional and Recurrent Neural Networks to develop a model that predicts image sentiment. Their ultimate object is to machine generate captions that identify both objects and emotions, with strong results. Unfortunately, crowd sourcing image sentiment is expensive and thus, not always an option (especially for me!).

Narihara et al, "Mapping Images to Sentiment Adjective Noun Pairs with Factorized Neural Nets" [7]

This is yet another paper highlighting the superiority of Neural Networks for computer sight. This time, the authors rely on Factorized Neural Networks. The paper relies on the same SentiNet database as Matthews et al. In this instance, the authors attempt to develop a model that can generate an adjective noun pair (like "beautiful sunset" or "ugly baby") to describe an image. The paper highlights some of the difficulty with ascribing image sentiment due to issues of context (the same image can hold different sentiments in different contexts). They argue that Factorized Neural Networks are superior to the more "traditional" CNN approaches, and show strong results in generating adjective-noun pair descriptions.

Seo and Kang, "Study on Predicting Sentiment from Images Using Categorical and Sentimental Keyword-Based Image Retrieval [9]

The authors note the difficulty in assessing image sentiment because of issues of subjectivity and context: namely that two people can assess the same image as having different sentiment depending on the context in which that image is seen, and on the personal biases of the viewer. In spite of this, they argue that image metadata, combined with low-level features, such as colour and texture, can provide an effective way of deriving sentiment in a machine-learning context.

Wang et al, "Unsupervised Sentiment Analysis for Social Media Images" [13]

This article is perhaps the closest to my original intention for this project. The authors approach the problem of trying to classify image sentiment by using a combination of image analysis and text analysis to determine the sentiment of an image. The study relies on images crawled on Instagram and Flickr. They find that this combination approach provides a better prediction of "ground truth" sentiment (as determined by crowd-sourced image scores) than other existing methodologies. Importantly, they also make use of responses to social media posts in their analysis, not just on the original post.

Text Sentiment Analysis

O'Connor et al, "From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series" [5]

While much of this paper does not apply directly to my work, the main take-away it provides is how it justifies sentiment ascription to Twitter messages. The authors argue that, given the shortness of Twitter messages, sentiment can be effectively ascribed by a simple tally of negative words and positive words in a text, with the majority count determining overall sentiment. They choose to use the OpinionFinder lexicon to do their sentiment ascription.

Maynard et al "Multimodal Sentiment Analysis of Social Media" [8]

This paper highlights the difficulty in applying Natural Language Processing techniques to understand social media sentiment, due in large part to the short nature of these communications, the lack of grammaticality, and the use of sarcasm, swear words, etc. They highlight the critical nature of hashtags to understanding social media sentiment, and the difficulties of hashtag decomposition. Finally, they note that image data can better help us to understand social media sentiment, although the approach they take is fairly simplistic, relying only on facial emotional detection.

Kouloumpis et al, "Twitter Sentiment Analysis: The Good, the Bad and the OMG!" [10]

The authors look at various techniques for assessing sentiment in Tweets, acknowledging the difficulty of ascribing sentiment to texts that are as short and often ungrammatical as tweets. They explore the effectiveness of hashtags and emoticons as methods to better classify tweets, finding that hashtags provide a significant improvement to sentiment classification, while emoticons only provide a minimal improvement in classification accuracy.

Tang et al, "Building Large-Scale Twitter-Specific Lexicon" [11]

The authors attempt to build a more effective lexicon for assessing tweet sentiment. They do this by leveraging Urban Dictionary to enhance existing lexicons, and by applying representational learning algorithms. They find that this approach provides an improvement in performance over existing, simpler lexicons. Unfortunately, this lexicon does not seem to be available online. While their methodology seems to improve the ability to classify tweet sentiment, it does not seem to provide a great boost in performance, meaning it is not necessarily worth the additional programming overhead in the context of my project.

Hutto and Gilbert, "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text" [14]

The authors develop a relatively sophisticated approach for analyzing social media text, including slang, emoticons, use of capitalization, and other contextual tools to understand and rate the relative positivity or negativity of a specific social media text, showing results that are overall superior to other text-analysis tools, and showing high correlation with human scorers. Importantly the authors have also made the code for this tool publicly available.[1]

Nielsen, "A New ANEW: Evaluation of a Word List for Sentiment Analysis in Microblogs" [15]

The author of this paper also attempts to develop a lexicon specifically aimed at understanding sentiment in Twitter-like microblogs. The lexicon uses a positive/negative 5-point scale to identify word sentiment and strength of sentiment. This scale is applied crudely to a tweet - ignoring issues of context and negation in the text (i.e. some of the fancier analysis done in VADER). In spite of this, this simple approach does a good job of determining Tweet sentiment. Also, importantly, the lexicon is freely available online[2]

Implications of Literature Review

In terms of the problem I'm attempting to solve in this project, a few key things have come through from this brief literature review:

1) It seems well-established that textual information supplied with images can be used to determine or help determine sentiment of an associated image
2) While lower-level image information such as colour and texture can be used to determine image sentiment, researchers have seen much greater success using Neural Network based approaches. There does not seem to be overall unanimity in the specific Neural Net methodology that is best, but Convolutional Neural Nets come up with some regularity and seem a good bet
3) Attributing sentiment to Twitter posts is a tricky business, due in great part to the short nature of these texts, the slangy nature of their language, and their lack of conventional grammar. As such, there are numerous methods that have been developed to assert text sentiment in tweets, most of which show good performance. As our job is only to identify positive, negative or neutral sentiment, it may not be ideal to rely on any one specific approach. Instead, it might make more sense to apply a few different approaches and use only those tweets where the different approaches are in agreement.

[1] Either at https://github.com/cjhutto/vaderSentiment or https://pypi.python.org/pypi/vaderSentiment. Both are written for Python2 and need adaptation to run in Python3

[2] http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010

[2] http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010

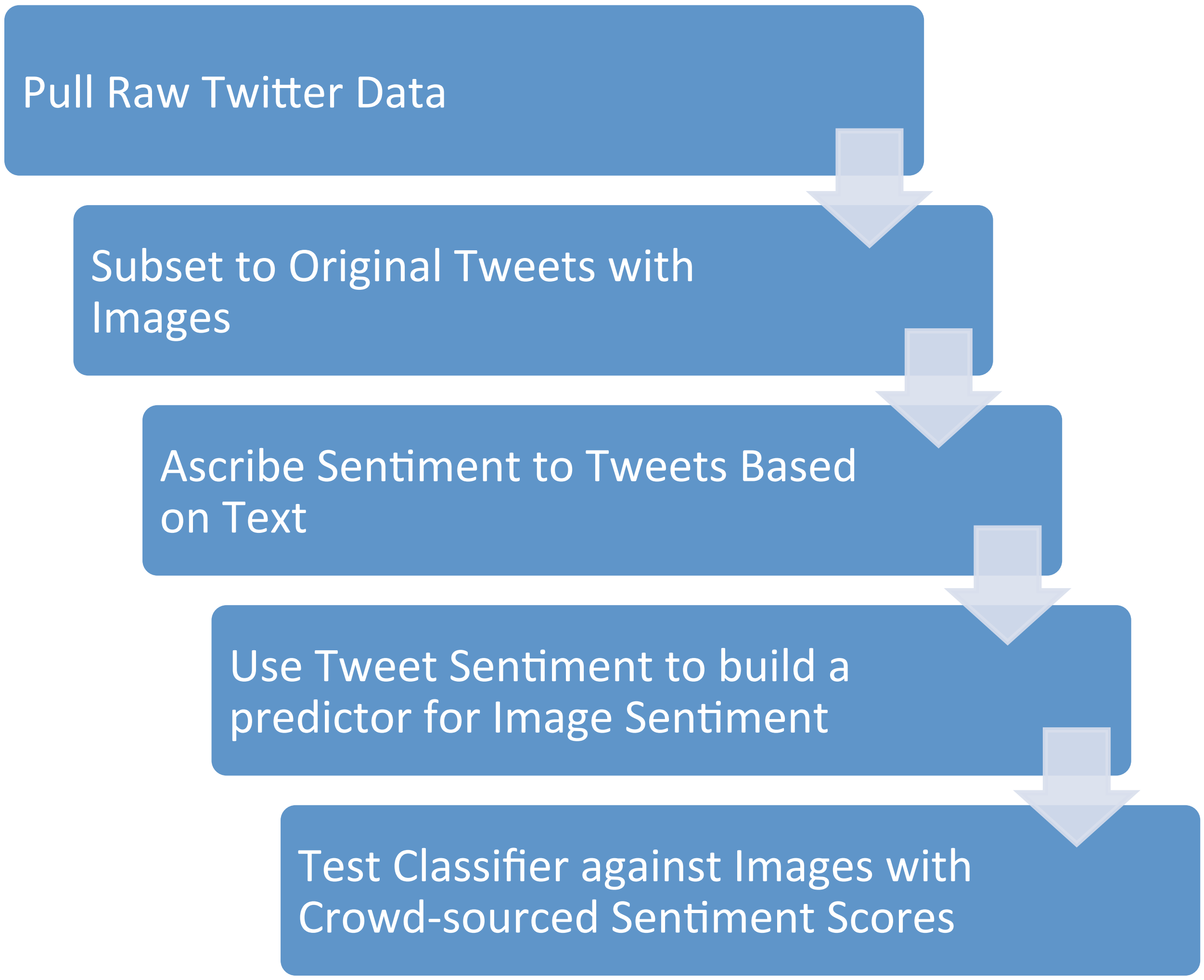
Dataset

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 was tested against a test panel from the original downloaded 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


Pull Raw Twitter Data
Subset to Original Tweets with Images
Ascribe Sentiment to Tweets Based on Text
Use Tweet Sentiment to build a predictor for Image Sentiment
Test Classifier against Images with Crowd-sourced Sentiment Scores

Step 1: Pull Raw Twitter Data

This is a fairly straightforward 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 8,129,241 + 1,404,433 tweets downloaded.

Code used for this step is posted at https://github.com/asterix135/CKME136/blob/master/Python_code/twitter_stream.py.

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, 129,378 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 hashtag splitting (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

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[3]. 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.

[3] I did not make specific reference to Hu-Liu in the Literature review part of this paper, but the lexicon is referred to as a touchstone in many of the papers noted above. See Ming and Liu [16] for details on the lexicon. The lexicon is being applied in the same manner as described by O'Connor et al [5]

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  0.519651  0.421767
afinn  0.519651  1.000000  0.667144
huliu  0.421767  0.667144  1.000000
```

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

```
Positive sentiment: 15545
Negative sentiment: 5422
Neutral sentiment: 50012
```

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

Step 4: Build Image Classifier

In order to build an effective image classifier, the raw images need to be transformed into a matrix-like data structure for processing.

Step One

The first step in this process was to download all the images from the tweets captured earlier in the process, convert them to jpegs, and resize them all to 400x400. Code for this step is available at:
https://github.com/asterix135/CKME136/blob/master/Python_code/images/download_images.py

Step Two

Upon reviewing the downloaded images, it quickly became apparent that certain images were repeated in multiple tweets, even though those tweets did not appear to be retweets of each other. This required two steps:

1) Find the matching images

   This was done by calculating a perceptual hashcode for each image, and then comparing the hashcodes using Hamming distance. Images with exact matches or very little difference were assumed to be matches, and taken out of the original dataset.

   Code for this step is available at
   https://github.com/asterix135/CKME136/blob/master/Python_code/images/find_duplicates.py

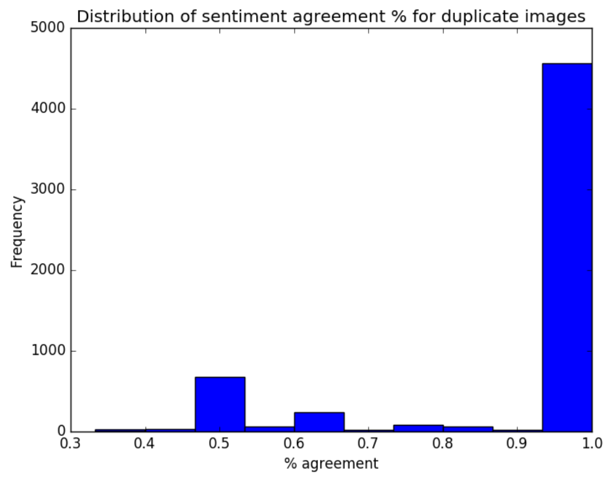
2) Decide how to treat sentiment for images that were tweeted

There were a couple of options here:

- throw out multi-tweet images
- adjust ratings for multi-tweet images to reflect majority sentiment

I ran this code to examine the correlation between repeated images: https://github.com/asterix135/CKME136/blob/master/Python_code/exploratory/images/compare_sentiment_multi_image.py

This showed that in appx. 75% of cases, repeated images generated the same sentiment value, with the distribution as shown below:



Given the high level of agreement, the "master" image's sentiment was set in accordance with the majority value. In cases where the agreement level was below 80% the "unclear sentiment" value for the image was set to 1, meaning the image would be excluded from future analysis.

Step Three: Get Data in an Appropriate Format

For most of the classifiers included in Python's sklearn package, the data is required to be in a 2-dimentison matrix (one row per observation), while for the neural network I implemented (based on the Theano package and LeNet), the data was required to be in a 4-D matrix (each observation is a 3-D array, with a separate 2-D layer for each RGB layer). As such, there are two sets of code for this stage. Each ensures an equal representation of images from each class.

1. Data preparation for general sklearn approaches: https://github.com/asterix135/CKME136/blob/master/Python_code/classifiers/preprocessing/img_preprocess.py

2. Data preparation for Neural Network:
   https://github.com/asterix135/CKME136/blob/master/Python_code/classifiers/preprocessing/nn_preprocess.py

Step Four: Build Machine Learning Models

Non-Neural Network Approaches

As I am more familiar with implementing some of the more traditional machine learning algorithms, my first attempt was to apply a couple of different methodologies to the data:

- Support Vector Machine:
  https://github.com/asterix135/CKME136/blob/master/Python_code/classifiers/svm.py
- Multi-class Logistic Regression:
  https://github.com/asterix135/CKME136/blob/master/Python_code/classifiers/logistic.py
- Random Forest:
  https://github.com/asterix135/CKME136/blob/master/Python_code/classifiers/rf.py

The code above all includes options to apply PCA to reduce the dimensionality of the image data (a 400x400 RGB image produces nearly half a million data points per image), but there was really no effect on classification efficiency.

Neural-Network Based Approaches

Two approaches were taken here: Both a multi-level perceptron model, based on the Theano package, and a Neural Network approach that leveraged the LeNet engine.

Code for these two approaches is as follows:

- Multi-Layer Perceptron:
  https://github.com/asterix135/CKME136/blob/master/Python_code/classifiers/neural_net/mlp.py
- Convolutional Neural Network:
  https://github.com/asterix135/CKME136/blob/master/Python_code/classifiers/neural_net/nn_theano_v3.py

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 consisted of 2 separate steps:

1) Testing against twitter images. This was a simple randomized test/train split of the original twitter data.
2) Testing against images with human-scored sentiment. As noted in the "Data" section, Crowdflower has made available a set of images that 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).

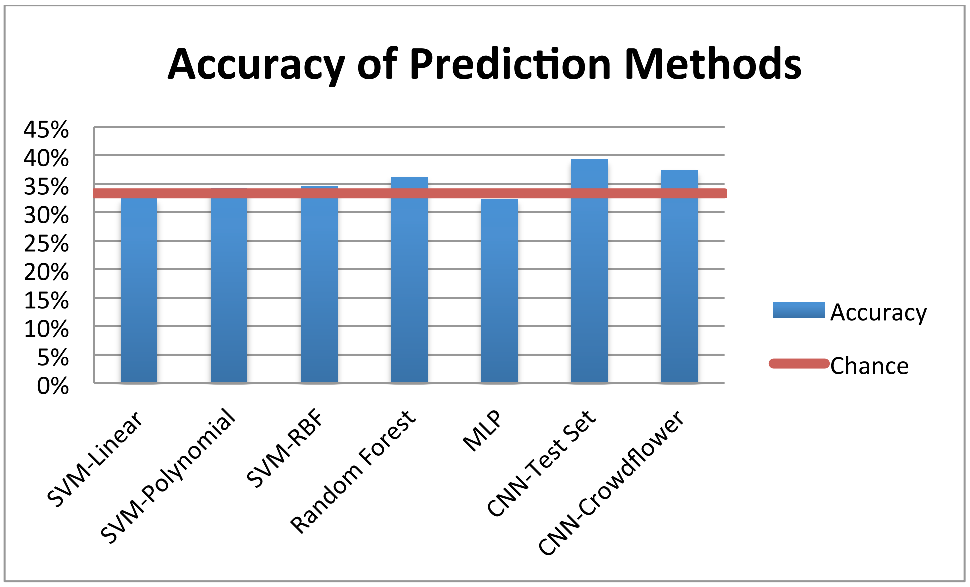
Base success was assessed as results that are better than chance, although results will also be assessed against previously reported sentiment classifiers.

In most cases, the testing code is included in the model code indicated in step 4, with the exception of the final CNN model:
https://github.com/asterix135/CKME136/blob/master/Python_code/classifiers/neural_net/nn_theano_compare_datasets.py

Note that the Crowdflower images are only scored as having positive or negative sentiment; there is no "neutral" category, unlike the scoring system I developed for the Twitter data.

Results

Essentially, this approach failed. In every case, our classifier was unable to achieve results that were much different from chance. The best results came from the Neural Network, at 39% accuracy when applied to the Twitter test set, and 37% on the Crowdflower images. While this is somewhat better than chance is not a level of accuracy that serves any particularly useful purpose. And it's far from the levels of accuracy seen in the literature referenced earlier in this paper.



In contrast, when I turned the neural Network to the Crowdflower data, I saw much better results, with accuracy rates at 74% (compared to a chance value of 50%), meaning the Neural Network could indeed identify image sentiment with coherently divided data.

Specific Findings

Rather than dumping a whole bunch of statistics here, the results for the non-Neural-Net classifiers are made available at
https://github.com/asterix135/CKME136/blob/master/Python_code/classifiers/results.txt

As the premise of this paper is that a Neural Network would be most effective at classifying these images, I am including the results of from the CNN-based classification on the Twitter test panel:

```
             precision    recall  f1-score

         -1       0.38      0.39      0.39
          0       0.40      0.42      0.41
          1       0.39      0.37      0.38

avg / total       0.39      0.39      0.39
```

```
Accuracy score
==============
0.392712550607

Confusion Matrix
================
[[643 493 518]
 [489 672 446]
 [546 508 625]]
```

Conclusions

As noted, this approach did not provide an effective method for image sentiment classification, as the Neural Network only returned an accuracy rate of 39%. While somewhat better than chance, and better than all the other predictive methods, this is not a level of accuracy that has any utility.

The problem is likely with the way the data set was constructed. We have seen, from the literature, and from applying the Neural Network to the Crowdflower data set that Neural Nets can be effective ways of classifying image sentiment. In this case, however, we were ascribing sentiment based on the textual content of the accompanying tweet. This approach probably ignores the fact that tweets often use images ironically. In addition, a large number of images seem to be commercial, and thus, perhaps with image sentiment not necessarily linked to the text content of the tweet.

In short, Twitter, at least as it is being used now, does not seem to have any significant relationship between text sentiment and image sentiment. Other platforms, such as Flickr, which have been used by other researchers, likely offer better data for unsupervised learning of image sentiment.

Bibliography

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

[2] Quanzeng You, Jiebo Luo, Hailin Jin and Jianchao Yang, *Robust Image Analysis Using Progressively Trained and Domain Transferred Deep Networks*, Web arXiv.org, 2015

[3] Jianfang Cao, Junjie Chen and Haifang Li, *An Adaboost-Backpropagation Neural Network for Automated Image Sentiment Classification*, The Scientific World Journal 2014

[4] Stefan Siersdorfer and Jonathon Hare, *Analyzing and Predicting Sentiment of Images on the Social Web,* Proceedings of the 18th ACM International Conference on Multimedia, 2010

[5] Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, Noah A. Smith, *From Tweets to Polls, Linking Text Sentiment to Public Opinion Time Series*, Proceedings of the International AAAI Conference on Weblogs and Social Media, 2010

[6] Alexander Matthews, Lexing Xie and Xuming He, *SentiCap: Generating Image Descriptions with Sentiments*, Web arXiv.org, 2015

[7] Takuya Narihara, Damian Borth, Stella X. Yu, Karl Ni and Trevor Darrell, *Mapping Images to Sentiment Adjective Noun Pairs with Factorized Neural Nets*, Web arXiv.org, 2015

[8] Diana Maynard, David Dupplaw and Jonathan Hare, *Multimodal Sentiment Analysis of Social Media*, BCS SGAI Workshop on Social Media Analysis, 2013

[9] Sanghyun Seo and Dongwann Kang, *Study on Predicting Sentiment from Images Using Categorical and Sentimental Keyword-Based Image Retrieval*, Journal of Supercomputing, 2015

[10] Efthymios Kouloumpis, Theresa Wilson and Johanna Moore, *Twitter Sentiment Analysis: The Good the Bad and the OMG!*, Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media, 2011

[11] Duyu Tang, Furu Wei, Bing Qin, Ming Zhou and Ting Liu, *Building Large-Scale Twitter-Specific Sentiment Lexicon: A Representation Learning Approach*, Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers, 2014

[12] Damian Borth, Rongrong Ji, Tao Chen, Thomas Breuel and Shih-Fu Chang, *Large-Scale Visual Sentiment Ontology and Detectors Using Adjective-Noun Pairs*, Proceedings of the 21st International Council on Multimedia 2013

[13] Yilin Wang, Suhang Wang, Jiliang Tang, Huan Liu and Baoxin Li, *Unsupervised Sentiment Analysis for Social Media Images*, International Joint Conference on Artificial Intelligence, 2015

[14] C.J. Hutto and Eric Gilbert, *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*, Association for the Advancement of Artificial Intelligence, 2014

[15] Finn Årup Nielsen, *A New ANEW: Evaluation of a Word List for Sentiment Analysis in Microblogs*, Web ArXiv.org 2011

[16] Minqing Hu and Bing Liu, *Mining and Summarizing Customer Reviews*, KDD, 2004