# Studying the Emoticons of Twitter Users

CS 145 Data Mining
Professor Sun Yizhou

---

Group Number: 21
Group Name: Tweet Emojis
Group Members: Mark Tai, Nathan Zhang,
Jai Srivastav, Fan Hung, Michael Xiong

---

Midterm Report
November 3, 2017

# 1 Abstract

Our goal is to find and predict the general happiness of certain twitter users based on their tweets. In our work, we will extend beyond this and study periods of time following a large tragedy or incredibly happy event and analyze the general happiness of all users who were directly impacted by the event in a positive or negative way. We approach this problem by first training our neural network to detect positive or negative emoji's in tweets. This will allow us to examine all tweets and make hypotheses on the happiness index of users. Ultimately, we are expecting to see that tragic events usually cause a noticeable decrease in overall happiness for a period of time. We hope that these results can be used to simulate the general public response concerning any tragic or happy event, potentially ranging from presidential scandal to a holiday celebration.

# 2 Introduction

In today's world with applications like Twitter and Facebook, people consistently share their thoughts with the general public. Because of this, there is a significant amount of data that can be used to describe the emotions of people on an everyday basis. Many of their tweets use emojis or emoticons that give a strong indication of the happiness level of the words that they are using. We hope that this emoji-filled data will give our machine learning algorithms insight on what words or phrases generally constitute happiness or sadness.

On Twitter, it is common for users to respond to certain events that occurred and give their own personal opinion followed by a hashtag and the name of the event. This allows us to easily mine data for responses to a certain event. We hope to explore the happiness of both these specific response tweets and the overall happiness of the users after a certain event.

Twitter data has some unique problems. In tweets, syntax is much more fluid, and often, full statements are not written out. There is also a vocabulary of abbreviations and slang that is not found in typical language. One of our main solutions to this variability is the use of neural networks for classification as well as representing tweets as vectors built from context.

Tweets are also short and are made frequently, so to capture a user's happiness beyond a single tweet, we will also aim to aggregate tweets made over time.

The rest of our paper will explain the process of how we approached quantifying our data and how we will use machine learning to analyze it. Then, because we have not finished yet, we propose a schedule on the work we need to complete and discuss the progress we have already made.

## 3 Problem Definition and Formalization

In our project, we hope to identify two main points. First, we want to find the overall happiness levels of users based on their tweets. Second, we hope to find if this overall happiness is changed because of certain events, and if it is, for how long.

Mood detection is a classification problem. We are trying to classify tweets as either happy or sad, and the simplest way to approach this is setting extreme happiness as a 1 and extreme sadness as a 0. However we will not treat this as a binary problem. We will instead use a continuous scale from 0 to 1 to express the level of happiness that exists. By examining a large portion of their tweets, we hope to predict the average happiness of the user.

Examining the change in overall happiness may be a little more difficult. We are hoping to see a difference in happiness in both the general user base and in specific users following a tragic event. Because of this, we will analyze both general sentiment following a certain event and also track how certain users react for multiple events.

## 4 Data Preparation

To find and prepare data for this happiness predictor, we will be using emojis on Twitter to classify what is happy or sad, and then use that to train a neural network. To collect the data, we will search Twitter for English tweets with positive or negative emoji we predefine. Next, we will remove all links, names, vectorize the sentences with word2vec and doc2vec.

Stemming is a process where words are reduced to their most basic forms. This includes conjugating all verbs to one form and removing plurals. This process is not incredibly important because we plan to use Word2vec, which assigns each word a vector based on context of that word. It is incredibly powerful due to the fact that it recognizes relations in context. For example, the two words “king” and “queen”, vectorized should have very close vectors. Furthermore, the difference between their vectors conveys some meaning as well. An example of this is that if we computed (“king” - “queen” + “woman”), the vector we would receive would be closest to the word “man.” This shows that the concept of being male is captured inside this vector difference. Just like word2vec works on individual words, doc2vec vectorizes individual documents of any length. This not only takes into account the words used but also factors in the order they were used as well as some other hidden factors. Because doc2vec tries to find meaning in the sentences and is already programmed to deal with stop words and conjugations, we are choosing not to remove stop words and stem the words of the sentence before analysis.

Each tweet eventually is decomposed into a positive or negative emotion and a vectorization of word content. After both of these components are collected for every tweet, we will train the neural network with the training data. Through doing all of this, we can create an emotion classifier.

# 5 Method Description

## 5.1 Data Preprocessing

Our data preprocessing methods are described above, and will result in word2vec embeddings as our features labeled as either positive or negative examples. Because we will use k-folds cross validation, we will simply randomly partition this data into training and testing sets multiple times in order to evaluate our model. Then, tweets without emojis to label their sentiment will serve as our live data. While evaluation of our model should be primarily done with our test data set, we will also manually check some results on live data to ensure model accuracy.

## 5.2 Neural Net Training

We chose a neural net because of their flexibility. The neural net will take as input a single vectorized tweet, and output a predicted happiness score from 0 to 1 (0 being completely unhappy, 1 being completely happy). In training, the labels of the data will be binary, but we would like a continuous predictor of happiness vs. unhappiness rather than a binary classifier, so we will not threshold the sigmoid activation function (except when comparing against other binary classifiers). This is because a continuous value is more useful in ranking many tweets in their happiness, whereas binary classification is less informative.

In order to train the neural net, we will run backpropagation on our training set, and evaluate the resulting model on our test set. In order to arrive at a sufficiently accurate model, we will tune the topology of our neural net, trying various combinations of layers and nodes per layer. For each combination of hyperparameters, we will use the same training set, and cross validate with k-folds CV.

## 5.3 Aggregate User Tweets

Our neural net model will predict the happiness of a single tweet, but will not predict the mood of a user. To do this, we will aggregate a user's tweet over a period of time (by default 7 days), and combine the happiness score of their tweets over that period of time. This will give us a much better estimation of their happiness than a single tweet, as a singular happy tweet over a nice dinner may be overpowered by many tweets regarding a singular event causing much distress.

## 5.4 Post-Event Happiness Change

After aggregating a user's happiness over a certain period of time, we can then compare their baseline happiness with a period of time following a major event, such as the 2016 presidential election. We expect that such a polarizing event will manifest in a user's tweets, causing a large change in their happiness rating. Then, by comparing the change across multiple users, we may find patterns in how various users react to major events.

# 6 Experiments Design and Evaluation

## 6.1 Evaluation of Preprocessing and Dataset

As we analyze the semantics of the tweets we find, we may find that we need to prune out unintelligible words from the tweets we find. For example, urls, phone numbers, and gross misspellings are common in tweets. There are also common words like articles, which may be pruned out. To understand how many of the words found in our dataset fall under this category, it may be helpful to compute the most frequent and least frequent words and potentially prune them out. Another option is to run a more sophisticated parser or part of speech tagger such as the ones made by Carnegie Mellon's Tweet NLP group on our data to see how many words are unidentifiable.

### 6.2 Evaluation of Tweet Representation or Vectorization

The representation of semantically similar tweets as vectors should be similar given similar semantics, and if spelling is not accounted for in preprocessing, the representation needs to be robust against misspellings. To test this, we can use sets of alternate spellings and synonyms found from the clustering analysis done by the Tweet NLP group. We can vectorize tweets multiple times changing out twitter words for their alternate spellings and synonyms in the dataset, and compute cosine similarities of the found vectors against those of the unmodified tweets.

Previously unseen words that arise in the test set is also a problem for our embedding of found vectors, so we will also need to test the quality of our accounting for unseen words. To test this, we can create tweets where we substitute words with previously unseen spellings and synonyms, and compute cosine similarities against the original tweets.

### 6.3 Evaluation of Happiness Index Per Tweet

The main goal of our project is to predict whether a happy face or sad face is more applicable to a given tweet. To evaluate this, we will evaluate our model on a set of tweets not included in our training set that originally contains happy and sad face emojis (before we run our model, we will prune out the emojis) a set of tweets that contain emojis that are not the happy and sad faces, and a set of tweets that contains no emojis. For testing on a set of tweets originally containing happy and sad face emojis, we will run a k-fold cross validation on our dataset of tweets with happy/sad faces. The additional tests on the set of tweets containing non-happy/sad face emojis and the set of tweets containing no emojis will simply be run once on their respective sets. Additionally, for each of these test sets, ROC curves and Recall/Precision graphs will be plotted.

### 6.4 Comparison Against Other approaches:

The previous tests evaluate the effectiveness of our model at assigning a happy or sad emoji label to a tweet, but this metric itself may not necessarily match other indexes of textual happiness. To test how well our model, trained on the happy/sad emoji generalizes, we will compare our model against others that also compute a positive/negative index. To do this, we

first compute the labels provided by the alternative APIs. For APIs that compute binary labels, we can threshold our approach's results, treat the api labels as ground truths and compare true positive and true negative rates. For APIs that compute labels on a continuous sliding scale, we can rescale our results and the api results to a 0-1 scale and compute a mean squared error of the labels. However, the comparisons here will primarily serve as a benchmark, as the sentiment apis may define positivity differently.

### 6.5 Evaluation of Happiness Index for a User in a Given Time Frame:

To evaluate the happiness of an individual, we can also take advantage of other tweets they have recently made. To test our aggregate model, which will incorporate tweets over several days, we need to again, evaluate sentiment, as opposed to happy/sad face prediction. To do this we can test against the other APIs from our previous approach, and then we can measure the changes in sentiment based error between our aggregate approach and our single tweet approach.

### 6.6 Computation of Happiness Index for Positive and Negative Events:

Finally, we will observe the behavior of our model on tweets involving events in the real world, which are tragic or celebratory. We will hand select a few events, (which we determine to be happy or sad) and again label the tweets using our single tweet model, our time based aggregate tweet model, and other api models. These labellings will let us see how our models compare against others on tweets, which we assume to be heavily polarized.

## 7 Schedule

### Week 6

Our crawler will be complete by now, and we plan to use Word2Vec to vectorize our text data to create a training set that we can feed to a neural network.

### Week 7

Train neural nets with varying hyperparameters. Compare cross validation results to select optimal model for predicting happiness in a single tweet.

### Week 8

Select the best model trained and aggregate tweets over 7 days, and evaluate against other mood predictors

### Week 9

Report, debugging, miscellaneous cleanup

### Week 10

Flex time for either doing further analysis, or finishing other parts of the project

## 8 Progress Discussion

Our crawler is complete, and we are able to scrape individual tweets from Twitter. In addition to the text body, we also crawl the username, as well as date posted. While these are not needed in training the neural net, they are required to conduct the actual mood prediction of users, as we plan to aggregate tweets over a 7 day period. Furthermore, we have planned out weekly meetings to both update team members, as well as plan out further work. A rough schedule is above, but details may be subject to change.

## 9 References and APIs

- Word2vec
    - https://www.tensorflow.org/tutorials/word2vec
    - https://arxiv.org/abs/1301.3781
- Gensim (big ML library that includes word2vec)
    - https://radimrehurek.com/gensim/models/word2vec.html
    - Includes other word vector embedding strategies
- Another word vectorizer
    - https://fasttext.cc/
- CMU's tweet parser
    - http://www.cs.cmu.edu/~ark/TweetNLP/
- Stanford's probablistic parser
    - https://nlp.stanford.edu/software/lex-parser.shtml
- Other twitter sentiment analysis APIs and approaches
    - http://www.ijcaonline.org/research/volume125/number3/dandrea-2015-ijca-905866.pdf
    - http://www.ijcaonline.org/research/volume125/number3/dandrea-2015-ijca-905866.pdf
    - https://www.tweetsentimentapi.com/
    - https://www.softwareadvice.com/resources/free-twitter-sentiment-analysis-tools/
- From Slides:
    - http://www.hlt.utdallas.edu/~kirk/publications/robertsLREC2012_2.pdf
    - https://mislove.org/twittermood/
    - Johan Bollen et al., Modeling Public Mood and Emotion: Twitter Sentiment and Socio-Economic Phenomena, ICWSM'11