# Predicting Emojis From Tweet Text Through Sentiment Analysis

CS 585 NLP, UMass Amherst, Fall 2017

Alexander Karle, Makenzie Schwartz, William Warner
akarle@umass.edu, mwschwartz@umass.edu, wwarner@umass.edu

December 2017

**Abstract**

We attempt to classify tweet text by emoji as posed in the SemEval-2018 Task 2 through the use of sentiment features. Since emojis are often representative of emotions, we hypothesized that using sentiment as a feature in emoji classification would prove useful, and we developed a flexible pipeline that allowed us to train a sentiment classifier and use this sentiment classification as a feature during emoji classification. Despite achieving reasonable accuracy with our sentiment classifier, it provided negligible improvement to our emoji classification compared to an array of preprocessing techniques. Our best results came from the combination of Logistic Regression with unigram, bigram, emotion lexicon features, and word clustering features on preprocessed data.

## 1 Introduction

Since their inclusion in an international keyboard on the iPhone in 2011, emojis (ideograms that represent various emotions, ideas, and concepts) have gained widespread usage in digital forms of communication [2]. Emojis are now present in most popular forms of social media and offer people an effective way to quickly communicate thoughts and reactions or to color an otherwise ambiguous message with the intended emotional tone. Recognizing the importance of emojis as a means to add expressiveness to otherwise plain textual communication, the organizers of SemEval-2018 have elected to present emoji prediction given textual data as Task 2 of this year's series of evaluations. Specifically, the organizers have collected a data set of tweets (500k in English, 100k in Spanish) that originally included one of twenty emojis, stripped the emojis from the original tweets, and used the emojis as labels for the data. The task they pose is to develop a classification system that can accurately predict which emoji corresponds to any given tweet.

Given that emoji usage is often correlated with emotion or sentiment values of the surrounding text, we hypothesized that incorporating sentiment and emotion features in our classifier would improve performance and point to an underlying relationship between sentiment/emotion and emoji choice. However, our results suggest no distinguishing relationship exists between emojis and sentiment features. Considering that this is a twenty class problem (as the data set uses twenty different emojis as labels), and that sentiment classification alone only yields only two or three classes (either positive/negative or positive/neutral/negative), we posit that sentiment classification doesn't provide enough meaningful information to distinguish between all twenty emojis. Furthermore, since the

SemEval task organizers selected the twenty most frequently used emojis to use as labels, and given that the more frequently used emojis in particular carry a positive sentiment [9], sentiment classification failed to provide meaningful distinction between emojis.

We did find some merit to emotion features corresponding to emoji usage, notably because emotion incorporates a wider breadth of classes which can distinguish between messages that carry the same sentiment. For example, both "love" and "joy" have a positive sentiment, but would offer more useful information in deciding between a red heart emoji (Unicode: U+2764) and a smiling face emoji (U+1F60A). However, further exploration into this relationship between emotion features and emoji classification would be required to draw concrete conclusions.

To test our hypothesis, we built a robust pipeline that would enable us to evaluate the accuracy of different probabilistic models in conjunction with various preprocessing steps and feature combinations. We used this pipeline to first establish a baseline accuracy to compare later results with. The most common label in the dataset was the red heart emoji (U+2764) which had a frequency of 22% (see Figure 1b). Setting 22% therefore as our most naive baseline, we were immediately able to improve upon this using a Multinomial Naive Bayes model with unigram features to achieve an accuracy of 29%.

The majority of our work was then dedicated to three separate areas: (1) exploring methods for adding sentiment features to our emoji classifier, (2) testing out a variety of preprocessing steps to better improve overall accuracy, and (3) evaluating a variety of models on the classification task. For (1), we developed multiple sentiment classifiers using a separate data set to be used to generate sentiment scores for tweets. We further attempted to incorporate features from sentiment lexicons directly in the emoji classification process. For (2), we explored word clustering, spell checking, part of speech tagging, and other simpler steps. Lastly, for (3), we tested Multinomial Naive Bayes and Logistic Regression on our pipeline, and tested Multilayer Perceptron and Random Forests on an embedded version of the tweet dataset. Overall, our best classifier achieved an accuracy of 36% by employing a Logistic Regression model with unigram, bigram, emotion lexicon, and word cluster features with location data removed.

## 2 Related work

### 2.1 Sentiment Analysis

A fair amount of work has been done on sentiment analysis of tweets. Giachanou and Crestani [3] present an extremely thorough survey of prior work done in the Twitter sentiment analysis domain, first giving a brief introduction to the topic of sentiment analysis and the particularities of sentiment analysis in Tweets and then presenting the various approaches that have been used to address the topic, including: machine-learning, lexicon-based approaches, graph-based methods, and hybrid approaches. Included also are explanations for common evaluation metrics for sentiment analysis. They conclude with an overview of existing datasets and open issues pertaining to Twitter sentiment analysis. We drew on this survey in designing our approach to both the overall emoji classification problem and the sentiment classification problem, including making the decision to experiment with sentiment lexicons.

When designing our own sentiment classifier, we considered the methodology of Pak et al. [11], who built a labeled Twitter corpus for sentiment analysis and then created a classifier for the data. For the former, they used a noisy labeling technique where they pulled only tweets containing positive emoticons and labeled these as positive (and did the same for negative). For the classifier, they used Naive Bayes to classify tweets based on extracted features. Features extracted were n-grams (they compared unigrams, bigrams, and trigrams) with some filtering done to improve accuracy. They reported high accuracy.

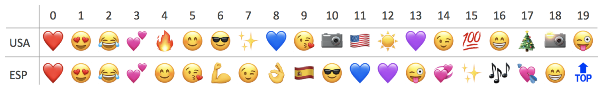
Rozental et al. [12] gives us an example of how to estimate the sentiment of a tweet on a five point scale (very negative, negative, neutral, positive, very positive) using two systems. The first system makes use of Recursive Neural Network models that were trained on tweets that were cleaned and processed using word replacement. The second system was designed for feature extraction using neural networks, Naïve Bayes, and logistic regression. Their conclusions reinforce our intuition that neural networks would be apt for this problem.

Kiritchenko et al. [5] further reinforced our confidence in the appropriateness of sentiment lexicons for this problem. The authors present the creation of their NRC-Canada Sentiment Analysis System for short pieces of text like tweets or SMS messages. Their classifier was produced by using supervised training on a linear-kernel SVM and leveraging multiple preexisting and newly-created, Twitter specific lexicons to improve performance. Their system performs well notably with respect to negated text, which they achieved by utilizing a lexicon for affirmative text and a separate lexicon for negated text.

### 2.2 Emoji Classification

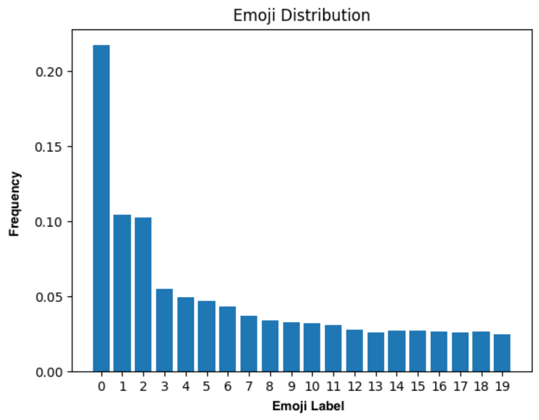
Given our hypothesis that sentiment features would improve emoji classification accuracy, we examined work relating emojis to sentiment analysis to both ground and motivate our own research. Miller et al. [7] investigate the degree in which the sentiment and meaning of emojis are interpreted differently by different people, with a particular emphasis on the effect that different renderings of the same Unicode emoji description (i.e. Apple's version of U+1F601 "grinning face with smiling eyes" vs. Google's version) have on said variance. The authors found that sentiment interpretations of emojis from the same platform vary 25% of the time, and the variance increases when the platforms are different. The survey was only done on English speakers from the United States to control for regional and cultural variations in interpretation. We considered this ambiguity inherent in emoji interpretation to be a potential source of inaccuracy for our classifiers.

Novak et al. [9] discusses the creation of an Emoji Sentiment Lexicon which is the product of 86 human annotators labeling over 1.3 million Tweets in 13 different languages (including English and Spanish). The final lexicon provides sentiment scores for the 751 most used emojis based on the dataset used. The authors concluded that most emojis have a positive sentiment, notably the more popular ones. In addition, they found that there was no major difference in emoji sentiment scores between the 13 languages that the original tweets were were written in. The authors' findings directly relate to our hypothesis and proposed methodology, in that emojis having predominantly positive sentiment would be likely to render sentiment classification useless as a feature in emoji prediction. Indeed, the results of our efforts could be seen to be a reaffirmation of Novak et al.'s findings.



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| USA | ❤️ | 😍 | 😂 | 💕 | 🔥 | 😊 | 😎 | ✨ | 💙 | 😘 | 📷 | 🇺🇸 | ☀️ | 💜 | 😉 | 💯 | 😁 | 🎄 | 📸 | 😜 |
| ESP | ❤️ | 😍 | 😂 | 💕 | 😊 | 😘 | 💪 | 😉 | 👌 | 🇪🇸 | 😎 | 💙 | 💜 | 😜 | 💞 | ✨ | 🎶 | 💘 | 😁 | 🔝 |

(a) Emoji Classes for English and Spanish datasets



(b) Distribution of labels for English dataset

Figure 1: SemEval Train Set Stats

## 3 Data

### 3.1 SemEval18 Task 2 (Emoji Prediction) Dataset

Our training dataset contains 491,526 English tweets containing a single emoji out of a set of the twenty most popular emojis in the US and around 100,000 Spanish tweets containing one of the twenty most popular emojis in Spain. We only used the English portion of the dataset. The emojis, shown in order of their popularity, are shown in Figure 1a.

This data was collected by crawling twitter using a SemEval-2018 provided seed and twitter crawler. The crawler initially saved the tweets as JSON data, and we processed this JSON data into our dataset using a SemEval provided script. The script stripped the emojis from the tweets and saved the tweet contents to text and label files. The label file contains a number from 0-19 on each line to represent the emoji associated with each tweet, and the text file contains the entire contents of each tweet including hashtags, location tags, and @user tags. The distribution of the emoji labels for the English dataset is shown in Figure 1b.

### 3.2 ArkTweet Twitter Word Clusters

Arktweet put together a hierarchical word cluster dataset that was collected with unsupervised learning. The clusters are designed to group words found in tweets to a cluster that represents intended meaning. There are 216,856 words that map to 1000 clusters in the Arktweet data. We use this data to try to make it easier for our models to train on the tweets and learn the important text features.

### 3.3 Sentiment140 Dataset

The dataset we used to train our sentiment classifiers for binary sentiment classification (positive/negative) was the Sentiment140 dataset, created by Go et al. [4]. Although the full dataset is not publicly available, the train and test set for English is available.

The train set consists of 1.6 million tweets scraped from Twitter and labeled using "distant supervision". In short, the authors pulled tweets with positive emoticons (such as :) and :-) ) and labeled those as positive and then labeled negative tweets similarly with negative emoticons. More on their collection technique can be found in [4].

The test set consists of 498 hand labeled tweets, labeled either as positive, negative, or neutral. As no tweets were labeled as neutral in the train set, we created our own test set from the train set, and found a separate dataset for training a sentiment classifier for the three class problem (see 3.4).

### 3.4 SemEval17 Task 4 (Sentiment Classification) Dataset

Task 4 of SemEval17 was sentiment classification of tweet text, and Subtask A involved classifying tweets as positive, negative, or neutral. As such, we used this dataset for training a sentiment classifier for this three class problem. This dataset consists of a train set of 20,632 tweets, which we split into train and test sets.

### 3.5 SentiWordNet

Our initial attempts to utilize sentiment lexicons made use of SentiWordNet [1]. SentiWordNet assigns positive, negative, and objectivity (objectivity = 1 - [positive + negative]) sentiment scores to each of the synsets in the WordNet [6] corpus. A synset is defined as a group of synonyms expressing a unique concept. The WordNet 3.0 corpus contains 117,659 synsets (82,115 nouns, 13,767 verbs, 18,156 adjectives and 3,621 adverbs). We make use the Natural Language Toolkit (NLTK) platform to load in and access the SentiWordNet scores.

### 3.6 NRC Emotion and Sentiment Lexicons

We were able to acquire access to seven different sentiment lexicons created by the National Research Council Canada (NRC) by emailing Dr. Saif M. Mohammad, Senior Research Officer at the NRC and one of the main creators of the lexicons. We made use of the NRC Word-Emotion Association Lexicon (aka EmoLex) which gives binary associations of 14,182 words for eight emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust) and two sentiments (negative and positive) manually annotated on Amazon's Mechanical Turk [8]. More information on this dataset can be found here: `http://saifmohammad.com/WebPages/lexicons.html`.

## 4 Method

### 4.1 Pipeline Description

Before diving into the specifics of our attempts, we first present an overview of the pipeline we have created (see Figure 2). This pipeline was developed to both produce sentiment classifiers (which it saves to file for use in emoji prediction) and emoji classifiers. All of this training and evaluation happens in the file "run_me.py". As arguments, run_me.py requires the directory of a data set and the name of a JSON file which specifies what classifiers to use, what preprocessing steps to complete, and which features to extract. From these arguments, it generates all permutations to

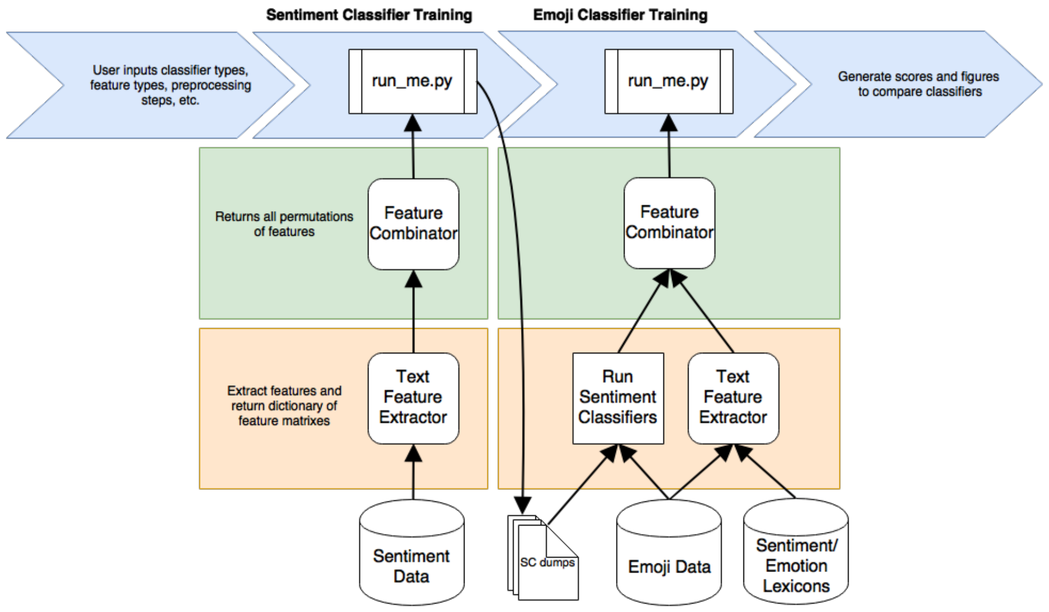

Sentiment Classifier Training
Emoji Classifier Training
User inputs classifier types, feature types, preprocessing steps, etc.
run_me.py
run_me.py
Generate scores and figures to compare classifiers
Returns all permutations of features
Feature Combinator
Feature Combinator
Extract features and return dictionary of feature matrixes
Text Feature Extractor
Run Sentiment Classifiers
Text Feature Extractor
Sentiment Data
SC dumps
Emoji Data
Sentiment/ Emotion Lexicons

Figure 2: Pipeline overview

test and evaluate and outputs results. In this way, the pipeline is flexible and engineered to allow the exploration of our guiding hypothesis that sentiment features may be useful all while making possible the search for the best emoji predictor.

In order to effectively extract text features and combine them with our own features, like sentiment scores, we make use of two more classes of our own creation: text_feature_extractor.py (FE) and feature_combinator.py (FC). To extract text features from a data set, we pass a list of the desired features (for example, [‘unigram’, ‘bigram’]) to FE, which makes use of scikit-learn’s CountVectorizer to extract these features. FC takes in a list of text features and a list of sentiment classifier scores and can be used to permutate over combinations of the various features in order to compare the results of using different subsets of features.

If training sentiment classifiers, either Sentiment140 or SemEval17 is used as a dataset, and the resulting classifier is saved to file using Python’s pickle library. Then, when training emoji classifiers, run_me.py loads in the SemEval18 emoji data and is able to load this pickled classifier and use it to classify each tweet as an additional feature on top of any textual features used for emoji classification.

Sentiment lexicon data can be used to generate features for both sentiment classifiers and emoji classifiers. FE utilizes NLTK to access SentiWordNet and custom code to load NRC Emolex to generate these features.

## 4.2 Preprocessing

### 4.2.1 Removal of Location Data

Many of the tweets in the SemEval-2018 data set contain a location tag which is always located at the end of the tweet (when it exists). The words in the location tag provide very little context for the tweet itself. For example, in this following tweet, we posit that the location provides no useful information:

Juice and be glad! ++ zozazindan @ Nashville, Tennessee

In addition, many tweets contained only a location tag and no other content. To remove the locations, we ran a script to remove the tag from each tweet or remove the tweet if the only contents were the location. The accuracy with location removed was slightly better in both Logistic Regression and Naive Bayes.

#### 4.2.2 Part of Speech Tagging

In an attempt to disambiguate further than simple n-gram features would allow, we decided to include a part of speech tagging preprocessing step. For the tagging itself, we make use of the Twitter part of speech tagger developed by the Noah's ARK group while at CMU [10]. The output of the POS tagger is then combined with the original unigrams themselves to produce tokens in the format of:

POS_N_word

where each token begins with POS, followed by an underscore, then the POS tag, a second underscore, and finally the original word itself. Naturally, when using the POS tagger, we also employ the tokenizer developed along with it—which contrasts from when we don't do POS tagging, where we make use of scikit-learn's CountVectorizer tokenization with some of our own customization instead. POS tagging also allows us to do a few othe preprocessing steps like stripping @ mentions, discourse markers and URLs. Using POS tagging alone did not improve classifier accuracy significantly.

#### 4.2.3 Word Clustering

The language used in tweets can be varied and can make use of a significant amount of internet slang and abbreviation. This means that there are many different words that can be used to achieve the same meaning. To try to standardize words, we used the Arktweet Twitter Word Clusters dataset to perform word replacement. When a word in a tweet matches one in the cluster dataset, we replace the word with a cluster id. This preprocessing method replaces a large portion of the words in our twitter datasets. This word replacement on its own results in a decline in accuracy with both LR and NB classifiers. It may smooth over subtleties that are important to classification of emojis.

#### 4.2.4 Spell Correction

Due to the fact that people are often more lax about spelling things correctly when tweeting (or communicating via digital platforms in general), we thought it might be wise to do some basic spell correction on the tweets before classification. Using a simple system that makes use of the free Enchant spell checking library by AbiWord (and the pyenchant packages that exposes it in Python), we added an optional preprocessing step that checks each word of a tweet and replaces each word Enchant doesn't recognize with Enchant's top suggestion (the suggestion with the smallest edit distance from the original).

However, there are several potential problems with this. First, in cases where there are multiple suggestions with the same edit distance, there is no way to disambiguate between them and the result of the spell correction can actually introduce more ambiguity than previously existed. Also, the use of alternate spellings is a widely accepted phenomenon in digital communication and so the spell checking might distort intentional misspellings that might be useful to preserve into something else.

#### 4.2.5 Other Preprocessing Steps

Other optional preprocessing steps in our pipeline include:

1. Lowercase: converts all text to lowercase before tokenization
2. Stop Word Removal: removes common words after tokenization
3. Strip Accents: converts unicode characters with accent marks to an equivalent without the accent mark
4. Count Single Character Tokens: by default, CountVectorizer doesn't count tokens with length less than two; this overrides that behavior and counts single character words as tokens
5. Strip @ Mentions: removes @user mentions from the tweets
6. Strip Punctuation: removes punctuation after tokenization
7. Strip Hashtags: removes #hashtags from the tweets
8. Strip Discourse Markers: removes discourse markers like RT from the tweets
9. Strip URLs: removes URLs from the tweets

### 4.3 Text Features Extracted

#### 4.3.1 N-grams

We found the simplest and most productive features to be n-grams: notably, unigram and bigrams. We extracted n-grams using scikit's CountVectorizer. Unigram features alone usually produced relatively high accuracices, while bigram features alone usually produced some of the lowest accuracies. More often than not, unigram and bigram features in conjunction would prove to be the most productive in terms of accuracy. We ran some experiments with trigrams, but decided against using n-grams where n $>$ 3 due to the relatively short length of tweets.

#### 4.3.2 Word Clusters as Features

In addition to using the Arktweet Twitter Word Clusters for preprocessing, we also used them as features by appending cluster IDs related to the words in the tweets to the end of the tweet. When the tweet text gets processed into N-grams, the cluster IDs become additional text features in relation to the tweet text. This change yielded a noticeable increase in accuracy and was much more effective than replacement.

#### 4.3.3 Sentiment Features Using Lexicons

Our initial attempts to incorporate sentiment lexicons utilized SentiWordNet. For most nouns, verbs, and adjectives, SentiWordNet provides a positive and negative sentiment score. We developed two different features using this lexicon: the first would iterate through each word in a tweet, lemmatize it, and retrieve both the positive and negative sentiment score from SentiWordNet (if it was contained in the lexicon) and sum them. It would then sum all of the individual word totals to produce a sentient score for the entire tweet, with negative scores indicating negative sentiment and positive scores indicating positive sentiment (0, in the unlikely case it occurred, would represent neutral sentiment). The second method performed a similar operation on each tweet, but simply returned -1 if the sentiment was negative, 1 if positive, and 0 if neutral.

Due to the potential negative value that could be output, we only evaluated these on Logistic Regression models (as Multinomial Naive Bayes does not permit features to have negative values).

Our second attempt relied instead on NRC EmoLex to provide values. NRC EmoLex provides binary inclusion values (that is, a 1 if the word is associated the category, 0 otherwise) for words in ten different categories: positive sentiment, negative sentiment, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. Our method involved iterating through each word in a tweet and incrementing counts for each of the ten categories for each word that was associated with it. Thus, for each tweet this method would return a list of the counts for each of the categories. For example, if the tweet passed in had the content "I love friendship!", the result might be [2,0,0,0,0,0,1,0,0,1], containing a 2 for positive sentiment (since both 'love' and 'friendship' have positive sentiment), 1 for joy (from 'love') and 1 for trust (from 'friendship').

Thus, instead of returning one feature value, this method returns 10, providing a little more information to distinguish between emojis than just sentiment alone.

#### 4.3.4 Sentiment Classification

As a text feature, we extracted a sentiment score from each tweet with the use of our own trained sentiment classifiers. To train these classifiers, we used mainly simple n-gram features. We trained three types of sentiment classifiers: binary classifiers (positive/negative), three-class classifiers (positive/negative/neutral), and a binary classifier that outputted not the predicted class but instead the probability of the positive class. The hypothesis behind the latter was that the positive/negative classification may not be very helpful as the emojis were largely positive in sentiment; as such, the probability of being positive may be useful as a sort of regression type score. We were unable to train a true regression model due to lack of labeled training data.

### 4.4 Models Evaluated

On this pipeline (as discussed in 4.1), we were only able to test Multinomial Naive Bayes and Logistic Regression classifiers. The reason for this limitation was due to tractability constraints. We found that the ngram vocabulary of the full emoji dataset was too large to train a Multilayered Perceptron or Random Forest, the other two classifiers we were interested in comparing. As such, we created an alternate pipeline that used document embeddings for features as opposed to unigrams and bigram such that the dimensionality of the data was lesser and the training tractable (to be discussed later in 4.6).

### 4.5 Evaluation

To evaluate our classifiers, we used the pipeline described in 4.1 to iterate over a series of all possible subsets of desired features, all with the same preprocessing applied. After training the classifiers specified for the run on the subset, we would then evaluate them based on their accuracy on a held out test set. In all cases, we used a 70/30 train/test split on the dataset used (either for sentiment classification or emoji classification).

In order to test different combinations of features and preprocessing steps, separate runs of the pipeline needed to occur.

### 4.6 Document Embeddings as an Alternate Pipeline

Due to the large dimensionality of the unigram and bigram features (the main text features that were used in the original pipeline), we were unable to train more complex models such as Random Forests and the Multilayer Perceptron. As we wished to explore these classifiers, due to the success of both of them in many problems, we created an alternate pipeline that used document embeddings as the feature space. This drastically reduced the dimensionality of the data, from hundreds of thousands (or millions in the case of bigrams) of features, to only 100 features.

To be specific, we used gensim's Doc2Vec implementation to convert each tweet into a 100 dimensional feature vector. Doc2Vec is a variant of the famous Word2Vec, but adapted for a document size embedding. We chose the number 100 heuristically, but did not have time to further explore the optimization of the embeddings.

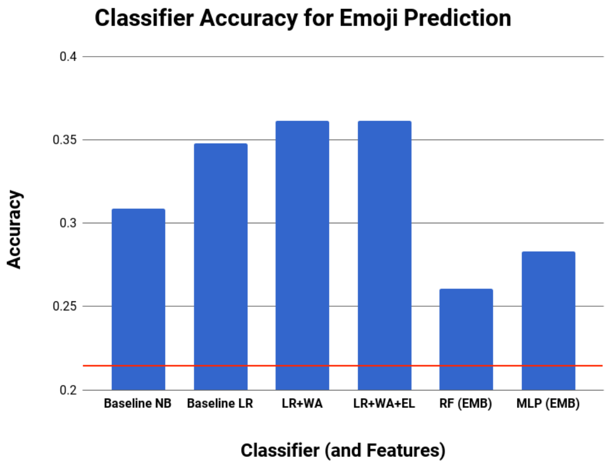
After generating the document embeddings, we proceeded to train both a Multilayer Perceptron and Random Forest over the data, using crossvalidation to select the optimal hyperparameters for both models. The models were evaluated on a held-out test set (using a 70/30 train/test split as done previously). The evaluation metric used was accuracy, to be consistent with our previous results.

## 5 Results

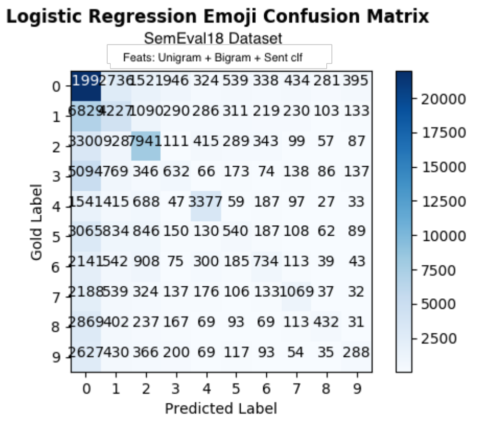
Table 1: Emoji Classification by Accuracy

| **Classifier** | **Features** | **Accuracy (%)** |
|---|---|---|
| Baseline | Guess only red hearts | 22.08 |
| Naive Bayes | uni + bi *(Baseline NB)* | 30.88 |
| Logistic Regression | uni +bi *(Baseline LR)* | 34.77 |
| Logistic Regression | uni + bi + Sentiment Clf. | 34.80 |
| Logistic Regression | uni + bi + wca | 36.12 |
| Logistic Regression | uni + bi + wca + Sentiment Prob | 36.11 |
| Logistic Regression | uni + bi + wca + NRC Emo. Feats | 36.15 |
| Random Forest | Document Embeddings (100 dim Doc2Vec) | 26.07 |
| Multilayer Perceptron | Document Embeddings (100 dim Doc2Vec) | 28.31 |

*Feature Key: uni = Unigrams, bi = Bigrams, wca = Word Cluster Append*



(a) Accuracy scores for notable classifier/feature combinations. Red line indicates naive baseline (22%). WCA is word cluster append; EL is EmoLex features; EMB is document embeddings.



(b) Confusion Matrix for 10 most frequent emojis (others not pictured, but predicted)

Figure 3: Classifier accuracy

A preliminary attempt to determine a baseline accuracy was found by simply predicting the most frequent label, the red heart emoji (U+2764), which achieved an accuracy of 22.08% ("Baseline"). By using a Logistic Regression model with unigram and bigram features (and no preprocessing), we achieved an accuracy of 34.77% ("Baseline LR"). Further accuracy scores achieved from different feature combinations and preprocessing steps were evaluated with respect to these baselines.

In summary, our findings suggest that our original hypothesis—that sentiment features would improve emoji classification accuracy—was incorrect. Our experiments with sentiment classification and sentiment lexicons proved not to impact classifier accuracy at all, although there is some evidence that emotion features might have some value. Rather, most of our improvements upon our established baseline scores came from appending the word clusters as a feature and various combinations of preprocessing steps. Having experimented briefly with document embeddings, we believe this reduced feature space in conjunction with different classifiers might offer improved accuracy results given more exploration.

Figure 3a depicts classifier accuracy scores for the different models we experimented with. Notably, Logistic Regression with unigram, bigram, NRC emotion features, and word cluster append features performed best; Random Forest and Multilayered Perceptron performed poorly, even below Multinomial Naive Bayes with no preprocessing, but we believe that with more effort these accuracies could be significantly raised.

The confusion matrix presented in Figure 3b demonstrates our classifiers tendency to overfit the most frequent label in the dataset. Indeed, as depicted in Figure 1b, the most frequent label (the red heart emoji) is over twice as frequent in the data set than the next most frequent label (the smiling face with heart eyes). The skew in distribution is the most likely cause for our overfitting problem.

We present our results in four parts. First, we detail the improvements in accuracy achieved by our most effective preprocessing steps. Then we illustrate the ineffectiveness of sentiment classification as a feature and offer our analysis on why it was so ineffective. Next we examine the marginal improvements achieved by using sentiment lexicons. Finally, we explore results from our alternate pipeline which made use of document embeddings, which we feel is the best path forward to creating better classifiers for this problem.

## 5.1 Preprocessing

The choice of preprocessing steps has a noticeable effect on the accuracy of the classifier although to varied degrees of effectiveness. Due to the fact that location tags had very little relation to tweet text, we found that stripping location data gave us an empirical improvement in all classifiers. Due to this consistent improvement, we created a modified dataset with location stripped for use in all of our tests.

No other preprocessing step on its own was very notable in terms of accuracy improvements and some on their own even seemed to decrease accuracy; however, most of the steps can only help the classifier by reducing the sparsity of features. Steps like lowercase, spell correction, strip accents, strip URLs, and stop word removal directly reduce feature sparsity and the high dimensionality of the data, so we often included these in our test runs.

Overall, however, we cannot assert in good faith that more or better preprocessing alone would significantly improve classifier accuracy—even our greatest improvements produced less than a 2% total improvement.

## 5.2 Sentiment Classification

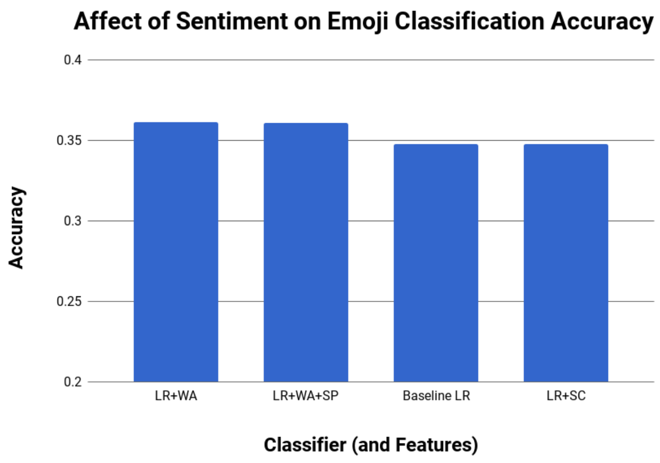
We trained three types of sentiment classifiers with our pipeline (Figure 2): binary classifiers, three-class classifiers, and a binary classifier that output its certainty of a tweet being positive.

Our best binary classifier, a Logistic Regression classifier that used unigrams and bigrams as features, obtained 82% accuracy when trained on the Sentiment140 dataset. Figure 4b displays a confusion matrix for this classifier. This was the model used for both binary classification features and positive probability features in the emoji classification problem.

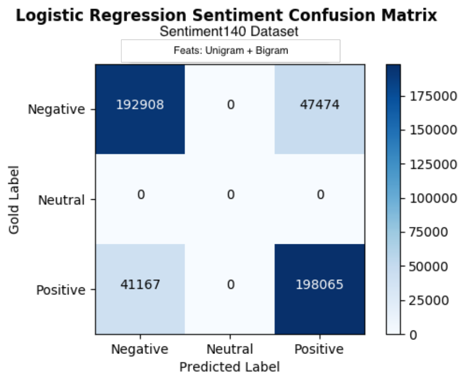
The best trained classifier on the three-class problem was also a Logistic Regression model with unigram and bigram features, this time trained on the SemEval17 twitter sentiment dataset; however, we were only able to obtain 66% accuracy, and thus decided that we would not use this as a feature as it was not reliable enough.

Thus, we only experimented with binary sentiment classification as a feature and with the probability of being positive as a feature. The former produced little to no boost in accuracy score, always contributing less than .05% accuracy. The latter produced even worse results, and in comparison to unigram and bigram features alone, it lowered the accuracy (though only marginally).

Figure 4a compares classifier accuracy scores with and without sentiment features. On the left we display the results for the Logistic Regression model with unigram, bigram, and word cluster append features; next to it is the same model with sentiment probability features included as well. On the right are the results for Logistic Regression with just unigram and bigram features compared to the accuracy of the model with sentiment classifications as an additional feature. Note that sentiment



(a) Accuracy scores for LogReg model with and without sentiment features. Baseline LR uses unigram and bigram features. WA is word clusting tags appended; SP is sentiment probabilities; SC is sentiment classification.



(b) Confusion Matrix for sentiment classifier (on held out Sent140 Test data)

Figure 4: Sentiment as a feature

features (both classification and probability) alone without ngrams produced no accuracy gain over the baseline of 22%.

After several attempts at using sentiment as a feature, we realized that its lack of contribution towards a better classifier was largely due to the positivity of all of the emojis in the label set; none of the emojis in the label set were strictly negative, and very few could be arguably situationally negative. For this reason, a binary positive/negative classification is understandably not as useful.

As far as using the probability of being positive as a pseudo-regression type score, we were hopeful that there would be some correlation between how confident the model was that a tweet was positive and the emoji used. For instance, a heart is probably more recognizably positive than a camera. This said, the lack of improvement with this feature may also be attributed to the non-perfect performance of our trained sentiment classifier. Although 82% is a respectable accuracy in a binary problem, this does not guarantee that the probability of being positive truly aligns with the positivity of the sentence; this would be one explanation of the useless nature of the feature in this data set.

After all of our negative results with experimenting with sentiment, we believe that it is useless as a feature for this particular emoji prediction problem (defined to only be the top 20 emojis); however, we believe that it may still be useful in a larger label set where negative emojis (i.e. angry face) are present.

## 5.3 Sentiment Lexicons

The two SentiWordNet features didn't provide any significant boost in classifier accuracy for either sentiment classifiers or emoji classifiers. We attribute this mainly to the overall issues we found with sentiment classification: it does not contain enough information to distinguish between twenty emojis that are categorically positive in sentiment. However, we posit that if the first method, which

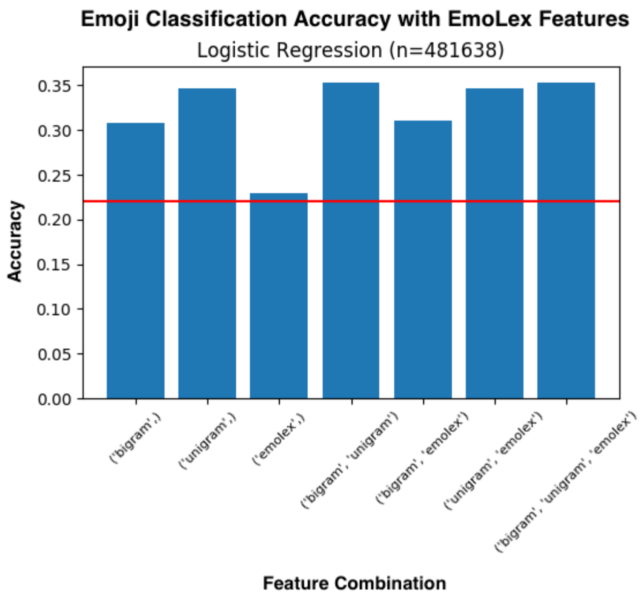


Figure 5: Effect of EmoLex features on emoji classification accuracy. Red line represents naive baseline (22%).

provides values over a continuous range, could be further engineered to provide inside into different degrees of positive sentiment, it might be a more useful feature.

The NRC EmoLex features do prove to be more useful, if only slightly: using these features consistently improved accuracy for both sentiment classification and emoji classification, but only by fractions of a percentage point, .03 to be exact (see Figure 5 and Figure 3a). Note that using EmoLex features alone achieved a higher accuracy than the naive baseline. None of the sentiment features alone outperformed the naive baseline—in fact, all achieved an accuracy equal to the baseline, indicating that these features were not informative enough to discriminate between emojis. For this reason alone, we think emotion features should be explored more.

We believe the cause of this improvement to be related to the emotion features the lexicon enables us to extract. This additional information permits the classifier to further distinguish between tweets that might have the same sentiment: for example, if two tweets both have a positive sentiment, but one has a higher value for the 'love' feature and the other has a higher value for 'joy', the classifier might be able to make a more accurate emoji prediction. Indeed, having hypothesized that being able to distinguish between either intensity or quality of positive sentiment would improve classification accuracy, we posit that one might be able to more significantly improve results using emotion features that further discriminate between emotions associated with positive sentiment.

## 5.4 Document Embeddings

After exploring sentiment as a feature, we created an alternate pipeline to investigate the use of document embeddings as features, so as to have a lower dimensional feature-space in order to train more complex classifiers. In particular we were interested in exploring the Multilayer Perceptron and Random Forests.

After running hyperparameter selection over the number of layers and number of iterations in the Multilayer Perceptron, our best result was only 28.31% accuracy. This is clearly an improvement over the baseline, but was far worse than a basic Logistic Regression with unigrams and bigrams ("Baseline LR").

Similarly, hyperparameter selection over the number of estimators in the random forest and the max depth of the trees produced our best result of 26.07%. This is again lower than the Baseline LR.

We believe these failures to be largely due to the lack of time dedicated to this specific pipeline. Much more of our effort went into the first pipeline and exploring the use of sentiment as a feature. As such, we never produced more than one embedding space, and thus did not experiment with different dimensional features. It is possible we over-compressed the features with our 100 dimensional space.

## 6 Discussion and Future Work

Overall, the twenty-class emoji classification problem posed by SemEval-2018 is a difficult problem. In addition to the fact that any twenty-class problem would be more challenging than say, a two-class or three-class problem, emoji usage can't definitively be tied to text features alone. For example, what distinguishes the use of a red heart, two pink hearts, a purple heart, and a blue heart? All of these emojis appear as labels in the data set, thereby creating ambiguity. Of course, in some cases there might be textual features that prove to be useful (for example, the purple heart might be associated with military terms and the blue heart is often used when supporting awareness for autism), but more often than not it might simply be user preference or some other unknown factor that determines when a user employs one or the other. Another, related factor is emoji presentation. While Unicode provides titles for each emoji, it is left to the system or application to depict the emoji. Apple, Google, Microsoft, and several others each provide their own renderings of emojis, some of which are dissimilar enough to introduce ambiguity for users of different platforms or applications (see [7] for further discussion).

While our hypothesis that sentiment features would aid classifier accuracy largely proved to be incorrect, we believe that further work in two areas would offer significant improvement in classifier performance: emotion features and neural networks. Given the slight accuracy boosts from incorporating simple emotion features from NRC EmoLex, it seems likely that given a better engineered set of emotion features or a data set that would enable one to train a multiclass emotion classifier, one could significantly improve emoji classifier accuracy. Separately, due to the complex nature of the latent relationship between text features and emoji usage, and given the current status of neural networks as being one of the best models for machine learning at the moment, it seems likely that neural networks would prove to be well-equipped for this sort of problem (given their notable success in the sentiment-analysis space). Given more time, we would have liked to explore Doc2Vec and Word2Vec more. We would have been able to tune these unsupervised models to the twitter data and crossvalidate over the dimensionality reduced to in order to find the optimal embedded dataset for Random Forests and neural networks (of which we could try different variants). We leave this to future work.

## References

[1] Baccianella, S., Esuli, A., and Sebastiani, F. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC* (2010), vol. 10, pp. 2200–2204.

[2] Blagdon, J. How emoji conquered the world. *The Verge* (Mar 2013).

[3] Giachanou, A., and Crestani, F. Like it or not: A survey of twitter sentiment analysis methods. *ACM Computing Surveys (CSUR) 49*, 2 (2016), 28.

[4] Go, A., Bhayani, R., and Huang, L. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford 1*, 2009 (2009), 12.

[5] Kiritchenko, S., Zhu, X., and Mohammad, S. M. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research 50* (2014), 723–762.

[6] Miller, G. A. Wordnet: A lexical database for english. *Commun. ACM 38*, 11 (Nov. 1995), 39–41.

[7] Miller, H., Thebault-Spieker, J., Chang, S., Johnson, I., Terveen, L., and Hecht, B. "blissfully happy" or "ready to fight": Varying interpretations of emoji. *Proceedings of ICWSM 2016* (2016).

[8] Mohammad, S. M., and Turney, P. D. Crowdsourcing a word-emotion association lexicon. 436–465.

[9] Novak, P. K., Smailović, J., Sluban, B., and Mozetič, I. Sentiment of emojis. *PloS one 10*, 12 (2015), e0144296.

[10] Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

[11] Pak, A., and Paroubek, P. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc* (2010), vol. 10.

[12] Rozental, A., and Fleischer, D. Amobee at semeval-2017 task 4: Deep learning system for sentiment detection on twitter. *arXiv preprint arXiv:1705.01306* (2017).