Predicting Twitter Emoji Usage with Neural Networks

Nihal Dhamani
nnd365
nihaldhamani@gmail.com

Vinay Valsaraj
vgv236
vinaygvalsaraj@gmail.com

Abstract

As emojis become increasingly common in forms of modern text, the importance of analyzing their intent and usage has increased as well. This investigation serves to create a model that learns how emojis are used to then predict an emoji that would most likely accompany any text. Our raw dataset is comprised of 2.9 million tweets, all from the month of October 2018. Our results showed both the LSTM model and CNN model heavily outperforming the baseline. We conclude by providing opportunities for further research and ideas for improving pre-processing of raw tweets.

1 Introduction

Expression of emotions were first used in the early 1990s with emoticons such as :-) and :-(. These were the first instances when emotional subtext could be added to a message, adding sarcasm, warmth, or even disgust. Rather than "You better" having a negative connotation, for example, one could imply flirting with "You better ;)". The first emoji was created in 1999 by Japanese artist Shigetaka Kurita.

After their creation, emojis had become so popular that the Unicode Consortium had decided to incorporate them into its text standard, the standard used by all computers. Apple incorporated emojis into iOS 5, allowing emojis to be used on mobile platforms. Since that time, emojis have become engrained in our culture today. Even the White House uses them (Mosendz, 2014)! As our language evolves and changes, language models must adapt and continue to best understand the textual input provided.

Companies such as Facebook, Google, and Twitter are all collecting as much information about users to build a profile on them. How they feel about certain topics, what their motives are, how they react, etc. Given young people are even adjusting texting habits to replace words, for example, "I love you" with "❤️", understanding how emojis portray intentions and motivations can be of great importance.

Emojis can also be used in ad targeting connecting with users based on their used emojis (Balonon-Rosen, 2019). Twitter, a large social network founded in 2006, has long served as a commonplace for emoji usage. Twitter is also heavily used in the NLP community for research for its diversity of topics and ease of access to the thoughts of many people.

Given a tweet, we encode the text as a sequence of words, w1, w2, ..., wN, where N is the length of the tweet. The model takes in the sequence as input and outputs predicted emoji e of the set of Z specified emojis. The label of a tweet is chosen as the most frequent emoji in the given tweet (further explained in Section 3). The trained model will be modified and tweaked through hyperparameters on a development set, and then evaluated against a separate test set. Accuracy and F1 will be calculated.

For this investigation, success will be quantified based on obtaining a higher accuracy than the baseline given by a simple perceptron model. Research has shown Recurrent Neural Networks to be one of the most powerful and useful types of networks. Different from a Feedforward Neural Network, Recurrent Neural Networks (RNN) not only examine current input but also take into account previous input. We have chosen to implement a RNN with Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997). The three models chosen to be extensively trained and tweaked are LSTM, Bidirectional LSTM, and Convolutional Neural Networks (CNN).

Finally, since emoji usage is highly subjective, a human survey was conducted in an effort to ana-

lyze human emoji usage and compare results with our predictive model.

2 Related Work

Prior investigations into emoji usage and predictions have been completed. Zhao et al. used tweets to train a LSTM and CNN classifier based on the top 50 emojis (Zhao and Zeng, 2017). Andrei et al. seeked to train simple classifiers such as Support Vector Machine (SVM) to illustrate their potential and ability to achieve similar accuracies to more complex models such as CNN (Catalin et al., 2018). Rather than solely binary sentiment classification for tweets, Wolny seeked to decipher tweets and emoji usage into a multi-way emotions classification (Wolny, 2016). These papers gave guidance and inspiration to pursuing the strongest classifier.

3 Data

The Internet Archive released a raw dataset of tweets from October 2018 scraped from the public Twitter API (Team). Twitter also has documentation regarding exploring tweet objects (twi). The data was preprocessed and split into train, dev, and test sets with a 80/10/10 split.

3.1 Preprocessing

Since tweets tend to be very noisy, careful planning and design for preprocessing of the massive dataset was required. First, only tweets containing emojis within the text and labeled as English were considered. Next, the top 30 most frequently occurring emojis were calculated and dataset was filtered to only use tweets containing at least one of those emojis. These steps reduced the dataset to 2.9 million tweets. For further processing, as shown in Table 1, there are four types of tweets that need to be handled: normal tweets, retweets, replies, and quoted tweets. Normal tweets do not have any affiliation with other tweets. Retweets are simply copies of other tweets with "RT:" and the twitter handle of the original user at the beginning. Both Reply and Quoted tweets only had the responding texts in the dataset. As such, the tweets had to be cleaned heavily to plain text with emojis with the following steps:

- Removing URLs, usernames, "RT", newlines, and all punctuation (e.g. @#!*)
- Handling contractions using Twitter-specific tokenizer (Erikavaris, 2017) (e.g. I'm = i am, Can't = can not)
- Searching for and removing emojis not found in top 30 set
- Removing infrequent words, i.e. words that appear 10 times or less throughout all tweets
- Converting all text to lowercase

Decisions involving heuristics for labeling, over-sampling and under-sampling, and count of emojis in our specified set all had to be made. In order to evaluate these decisions, a total of 16 different datasets were made and tested on each of the 5 different models. Reasoning and comparisons are outlined below.

3.2 Extracting Label: Duplicates vs Non-Duplicates

In order to create a fully labeled dataset from the collection of tweets, the correct label must be extracted from the text of each tweet and removed from the text. This is simple enough for a tweet with just a single emoji as we would use the only emoji present for the label. For tweets with multiple emojis, a decision was made to use the most frequently occurring emoji to represent the test. However, this begged the question of what would constitute the correct label if there were ties in the frequency of emojis. The two different approaches to tackle ties in frequency were (1) duplicate the text, each with a new label or (2) pick the last occurring emoji as the label. Different datasets of duplicates vs. non-duplicates were made in order to asses which method of extracting labels was better. Table 2 shows examples of the different label extraction methods.

3.3 Sampling vs Non-Sampling

The dataset contained a large class imbalance for certain emojis. For example, the laughing emoji, 😂, had over 600,000 tweets. Meanwhile, the halloween pumpkin emoji, 🎃, had a mere 50,000 tweets. This presented an issue of the classifier training too heavily on certain emojis, potentially decreasing prediction accuracy for infrequent emojis. In order to curb this imbalanced distribution, under-sampling the frequent emojis was evaluated. However, there existed a trade-off between having a large dataset vs. fixing the class

Type of Tweet	Example
Normal Tweet	Are there actually people who put milk in before the cereal?
Retweet	RT @natgeo Retweet to show support for Earth Day!
Reply Tweet	Just graduated!!! Congratulations @nick I still have one more final to go...
Quote Tweet	"A&M is the best university in Texas" Actually, no. It's UT.

Table 1: Examples of the four types of tweets

Tweet	Resulting Datapoints
I'm bored 😭	**text**: i am so bored **label**: 😭
I'm tired 💀 💀 🙄	**text**: i am tired **label**: 💀
I'm hungry 😩 😩 👀 👀	**text**: i am hungry **label**: 😩 **text**: i am hungry **label**: 👀
I'm sleepy 😒 😒 😭 😭	**text**: i am sleepy **label**: 😭

Table 2: Examples of label extraction methods

imbalance with under-sampling. In order to determine which part of the trade off was worth it, both sampled (50,000 tweets per emoji) and non-sampled datasets were created to be evaluated and compared.

3.4 30 vs 15 emojis

Emoji usage is highly subjective and vary drastically depending on the person. This can introduce a great deal of noise and makes it harder for classifiers to make correct predictions. Additionlly, many different emojis expressing similar sentiments exist. For example, heart emojis can be easily interchanged with a double pink heart emoji, 💕, or a purple heart, 💜. In order to improve accuracy and minimize the noise contained within the data, we chose to explore lowering the size of our emoji set from 30 to 15. Different datasets with top 30 and top 15 emojis were created to be evaluated and compared.

4 Models

In order to quantify success and determine the best model for emoji prediction, we compared five different classifiers.

4.1 Baseline Classifier

To determine a starting point for emoji prediction, a simple perceptron was used as a baseline. The baseline was used to determine how much room there was to improve and if different approaches were improving accuracy. The set of unique words in all training examples was found and indexed. Tweets were then indexed for each word and supplied to the perceptron, modifying the weight vectors of each predicted class accordingly.

4.2 Logistic Regression Classifier

Logistic regression is an extension of the linear regression classifier. Linear regressions fit the best hyperplane and provide a linear interpolation between points, not extending to multi-class classification (Molnar, 2019). Determining the threshold on the hyperplane at which one distinguishes different classes can be tough given training data that doesn't have clearly distinct features. Logistic regression uses a logistic function to squeeze the output of linear layers into a probability between 0 and 1 of an input being a certain class. In order to classify multi-class, it divides the problem into N binary classification problems and determines the prediction for c1, c2, ..., cN. The highest probability returned from the class is tracked and returned as the predicted class.

Our logistic regression mode was implemented using the scikit-learn Python library (Pedregosa et al., 2011). In order to take in input text, word embeddings were used to represent text in a vector format. Word embeddings serve the purpose of encoding input in an N-dimensional space. Each word has their own embedding with more similar words (e.g. "man" and "boy") being closer to each other. Aggregating all and averaging the word embeddings of an input text gives an input that can be piped into the logistic regression. We decided to use the Twitter GloVe dataset having a vocab of 1.2 million unique "words" for it was built off of 2 billion tweets (Pennington). It was understood that this dataset would best understand the similarity between our words as they're from tweets as well.

4.3 Convolutional Neural Network

CNNs are similar to multi-class perceptrons containing neurons with weights and biases. However, CNN's are different in that rather than having all fully-connected layers where all neurons in a layer are connected to all neurons in the previous layer, they have steps of convolutions and pooling. Convolutions and pooling results in combining the outputs of convolved neuron cluster at one layer into a single neuron in the next layer (Cireşan et al., 2011). Our implementation comes from the help of Denny Britz (Britz, 2016) and GitHub user jiegzhan (Jiegzhan, 2018) and looks roughly similar to Figure 1. As for our word embeddings, we had the option of using either GloVe embeddings trained specifically for twitter (Pennington) or training our own embeddings from scratch. The pre-trained word embeddings had vectors for 63% of our total vocabulary. Further analysis revealed that training our own embeddings resulted in higher accuracy so we opted to train our own word embeddings from scratch.

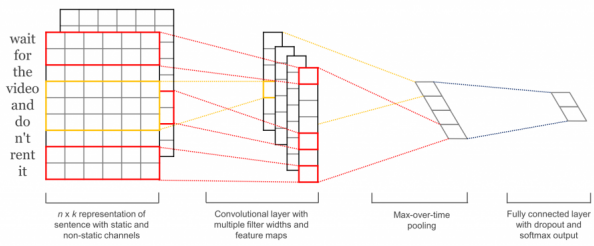

Figure 1: Network Representation of CNN (Britz, 2016)

4.4 Recurrent Neural Networks

4.4.1 LSTM Classifier

A Long short-term memory model (LSTM), as shown in Figure 3, is based on a Recurrent Neural Network (RNN) architecture (Hochreiter and Schmidhuber, 1997). An RNN differs from traditional feed-forward structure in that has feedback connections which make it ideal to process sequences of data, especially text. An LSTM is an RNN with a cell, input and output gates, and a forget gates. The gate combinations serves to regulate the flow of information over certain time intervals (Siegelmann and Sontag, 1995). For our implementation, we used to Keras (ker) to implement the LSTM model as shown in Figure 2. Once again we trained our word embeddings from

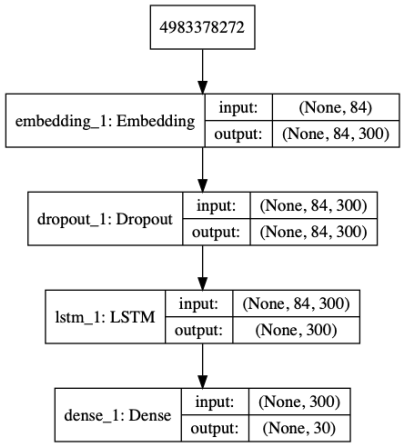

Figure 2: LSTM Model implementation in Keras with embedding dimensions = 300, a dropout layer, an LSTM layer with 300 hidden units, and a 30-class output layer with Softmax activation.

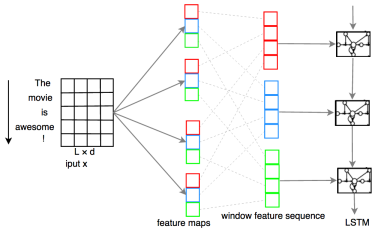

Figure 3: LSTM model schematic (Kwan-Yuet)

scratch.

4.4.2 Bi-Directional LSTM Classifier

A Bi-Directional LSTM is very similar to the LSTM described in Section 4.4.1 with one important caveat. The Bi-LSTM model duplicates the first recurrent layer in the network so that there are two layers of opposite directions mapping to the same output. The first layer provides data just like a regular LSTM layer; however, the second layer provides a reversed copy of the data (Schuster and Paliwal, 1997). The two layer outputs are then merged with in a specified manner. In our case, we concatenated the two results. We suspect this architecture to prove useful as the model could learn on much more context than a regular LSTM. Once again, we trained word embeddings from scratch.

5 Results

Not only was this investigation searching for the best classifier, but also for the best dataset. Accuracy is calculated by dividing correct predictions by total predictions. F1 score is a harmonic aver-

age of precision and recall, with a range between 0.0-1.0, 1.0 being perfect precision and recall. Accuracy and F1 score results for the dev set for all classifiers are compared in Tables 3 and 4. These were trained with similar hyperparameters in order to ensure that they could be compared against one another. Further discussions and observations are presented below.

Set of 30 Emoji				
	ND NS	**ND S**	**D NS**	**D S**
PER	Acc: 5.66 F1: 0.04	Acc: 6.21 F1: 0.05	Acc: 6.13 F1: 0.04	Acc: 8.04 F1: 0.06
LR	Acc: 28.58 F1: 0.21	Acc: 23.23 F1: 0.22	Acc: 27.6 F1: 0.20	Acc: 22.15 F1: 0.21
CNN	Acc: 30 F1: 0.30	Acc: 30 F1: 0.30	Acc: 30 F1: 0.30	Acc: 30 F1: 0.30
LSTM	Acc: 52.98 F1: 0.52	Acc: 45.82 F1: 0.46	Acc: 48.03 F1: 0.47	Acc: 42.2 F1: 0.42
Bi-LSTM	Acc: 30 F1: 0.30	Acc: 45.75 F1: 0.46	Acc: 47.9 F1: 0.47	Acc: 42.1 F1: 0.42

Table 3: Results on dev set for each classifier on all 30 Emoji datasets. ND=Not Duplicate, NS=Not Sampled, D=Duplicate, S=Sampled

Set of 15 Emoji				
	ND NS	**ND S**	**D NS**	**D S**
PER	Acc: 11.86 F1: 0.09	Acc: 11.46 F1: 0.09	Acc: 9.95 F1: 0.06	Acc: 10.23 F1: 0.05
LR	Acc: 37.85 F1: 0.3	Acc: 31.07 F1: 0.3	Acc: 36.5 F1: 0.29	Acc: 30.61 F1: 0.29
CNN	Acc: 30 F1: 0.30	Acc: 30 F1: 0.30	Acc: 30 F1: 0.30	Acc: 30 F1: 0.30
LSTM	Acc: 59.81 F1: 0.58	Acc: 51.91 F1: 0.52	Acc: 55.74 F1: 0.54	Acc: 47.67 F1: 0.47
Bi-LSTM	Acc: 59.91 F1: 0.59	Acc: 50.27 F1: 0.50	Acc: 55.72 F1: 0.54	Acc: 47.5 F1: 0.47

Table 4: Results on dev set for each classifier on all 15 Emoji datasets. ND=Not Duplicate, NS=Not Sampled, D=Duplicate, S=Sampled

5.1 Duplicates vs. Non-Duplicates

According to tables 3 and 4, non-duplicates outperformed duplicates in both accuracy and F1 score. It is hypothesized that this may be because of the noise that occurs when classifiers receive different labels for the same training data. Comparing datasets 30 ND NS and 30 D NS, accuracy and F1 score jump by 4% and 0.05, respectively.

5.2 Sampling vs. Non-Sampling

Unexpectedly, it appears that non-sampling outperformed sampling in both accuracy and F1 score. Under-sampling emojis with over 50,000 data points lowered accuracy. This could be because of the important distinctions in classifiers

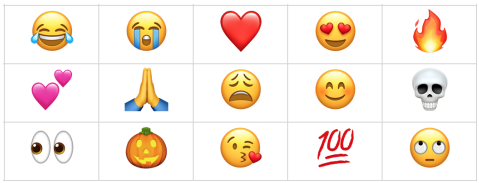

Figure 4: Final 15 emojis selected

make with certain data points that it was not able to capitalize on. Comparing datasets 30 ND NS and 30 ND S, accuracy and F1 score jump by 6% and 0.06, respectively.

5.3 30 vs 15 emojis

The classifiers performed better when only predicting 15 emojis versus 30 emojis. This was as expected because the classifier had less subjectivity between emojis being interchangeable and less room for error with only 15 possible classes. 15 NDS performed the strongest with an accuracy of 51.91% and F1 score of 0.50.

5.4 Final Dataset

A final dataset and model was chosen based on the results presented in Tables 3 and 4. Figure 4 shows the 15 emojis that were selected.

The final dataset selected:

- 15 emojis. Not sampled, no duplicates
- Size of training set: 1,726,968
- Size of dev and test sets: 215,871

5.5 Final Model and Results

The final model selected was an LSTM. The model was then further tuned by varying hyperparameters in order to achieve higher accuracy. Accuracies and F1 scores were then calculated for both the dev and test sets.

The following hyperparameters were used in the final model:

- Optimizer: ADAM
- Embedding Dimensions: 600
- Hidden Layers: 600
- Batch Size: 128
- Dropout: 0.1
- Epochs: 6

The results, shown in Table 5 for the final model are quite promising. Given that random guessing gets an accuracy of 6.67% and ours is roughly

	Accuracy (%)	F1 Score
dev set	61.27	0.6
test set	61.99	0.6

Table 5: Final model results

62%, it is safe to say that we succeeded in predicting and analyzing Twitter emoji usage, despite the noisy data and subjective behaviour of emojis. Additionally, the similar accuracies between the dev and test sets suggest that our model is robust. The F1 scores are on the lower end of the spectrum; however, still suggest that our model is mostly capable of producing low false positives and low false negatives.

5.6 Human Survey

Naturally, the usage of emojis is inherently subjective. Emojis can be seen as being open to interpretation and can render differently on different viewing platforms, potentially leading to communication errors (Miller et al., 2016). Given the extremely subjective nature, we were further interested in analyzing how varied emoji usage can be and how well our model holds up against human competition.

In order to facilitate this survey, a Google form (goo) was created with 15 sentences, ranging from short to long, and with 15 emoji options that the user could select for each of the sentences. It is important to note that the form was mainly distributed to college kids, which could introduce some bias as emoji usage might differ across generations. In the end, we were met with 49 total responses summarized in Table 6.

There were few key observations we took away from the survey. Even though the survey picked emoji and predicted emoji were not always they same, we found that the model predicted emoji was often the 2nd most frequent picked emoji in the survey. This again is due to the related nature of certain emojis and the similar sentiments they share. For instance, the kissy face emoji, 😘, and the hearts emoji, 💕, were often picked interchangeably. Additionally, we noticed that the shorter the sentence, the more likely it was for both the survey and the model to produce the same emoji. Finally, upon further analysis we noticed that certain key words were automatically associated with emojis. For instance, 'die' would bring up the skeleton emoji, 💀, 'red hot' would bring up the fire emoji, 🔥, and 'bless' would bring up the prayer hands emoji, 🙏.

Text	Survey	Model Pred
I love you	❤️ per: 63.3%	💕 prob: 27.7%
I hate you	🙄 per: 57.1%	💀 prob: 35.3%
I cant believe he offered me his coat to walk on over the puddle	😍 per: 46.9%	🙄 prob: 99.6%
Bless up	🙏 per: 89.8%	🙏 prob: 68.6%
I couldnt believe how beautiful she looked in the fashion show!	😍 per: 49.0%	😍 prob: 67.8%
James Harden is a baaad man	🔥 per: 24.5%	😂 prob: 29.5%
My dad randomly started singing Old Town Road this morning while making breakfast	😂 per: 38.8%	😩 prob: 18.9%
Why do all my favorite Game of Thrones characters get killed off??	😭 per: 57.1%	😩 prob: 42.6%
Really craving some pizza rn	😩 per: 51.0%	😩 prob: 77.0%
If I pay $40 for a haunted house I better die	💀 per: 30.6%	💀 prob: 41.7%
Grind don't stop	💯 per: 61.2%	💯 prob: 34.2%
You know boomers had it good because their go to midlife crisis move was buying an expensive car	🙄 per: 30.6%	😩 prob: 54.9%
Y'all wanna talk about ghosting, let's talk about jobs you apply for and never get denied or accepted	🙄 per: 28.6%	👀 prob: 46.4%
Harden is red hot from 3!	🔥 per: 83.7%	🔥 prob: 49.6%

Table 6: Results from 49 survey responses.

6 Conclusion and Future Work

A 50% gain in accuracy was made using the LSTM model versus the baseline classifier. There are a few things that could have improved the classifiers further.

Our approach for sampling included undersampling all classes to 50,000 data points. Instead, we could explore optimizing the data points chosen for each class (most words/context) and also using certain ratios for each emojis frequency. Additionally, we could explore oversampling as a method as well.

Tweets will continue to be very noisy and subjective with emoji usage. Further analysis of filtering tweets for their important keywords and removing noise could help classifiers learn more distinct features. Many emojis are interchangeable and having different meanings to different people making emoji prediction perfect near impossible. In the future, potentially grouping emojis together with similar meanings or creating vector represen-

tations of them, e.g. different colored hearts or angry emojis, could greatly increase accuracy.

Our heuristic of using the most frequent emoji, and in case of tie, most recent emoji, for selecting labels could potentially be incorrect in some cases where the emoji selected doesn't fully represent the text. Another potential avenue to explore would be determining where emojis fit best in tweets. Not only classifying text with the best accompanying emoji, but also incorporating the spatial aspect of emojis into training and predicting their location. In order to best predict emoji usage of a specific user, models could even be trained specifically to that user given enough training data.

References

Google forms: Free online surveys for personal use.

Keras: The python deep learning library.

Tweet object - twitter developers.

Peter Balonon-Rosen. 2019. That emoji you just tweeted could determine the next ad you see.

Denny Britz. 2016. Implementing a cnn for text classification in tensorflow.

Andrei Catalin, Giacomo Zara, Yaroslav Nechaev, Gianni Barlacchi, and Alessandro Moschitti. 2018. Exploiting deep neural networks for tweet-based emoji prediction. In *NL4AI@AI*IA*.

Dan C. Cireşan, Ueli Meier, Jonathan Masci, Luca M. Gambardella, and Jürgen Schmidhuber. 2011. Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1237–1242. AAAI Press.

Erikavaris. 2017. erikavaris/tokenizer.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Jiegzhan. 2018. jiegzhan/multi-class-text-classification-cnn.

Ho Kwan-Yuet. Deep neural networks with word-embedding.

Hannah Miller, Jacob Thebault-Spieker, Shuo Chang, Isaac Johnson, Loren Terveen, and Brent Hecht. 2016. "blissfully happy" or "ready to fight": Varying interpretations of emoji. In *Proceedings of the 10th International Conference on Web and Social Media, ICWSM 2016*, pages 259–268. AAAI press.

Christoph Molnar. 2019. Interpretable machine learning.

Polly Mosendz. 2014. The white house learned emoji to speak millennial.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington. [link].

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

H.T. Siegelmann and E.D. Sontag. 1995. On the computational power of neural nets. *J. Comput. Syst. Sci.*, 50(1):132–150.

Archive Team. archiveteam-twitter-stream-2018-10.

Wieslaw Wolny. 2016. Emotion analysis of twitter data that use emoticons and emoji ideograms. In *ISD*.

Lian Zhao and Connie Zeng. 2017. Using neural networks to predict emoji usage from twitter data.