Using Neural Networks to Predict Emoji Usage from Twitter Data

Luda Zhao
Stanford University
`ludazhao@cs.stanford.edu`

Connie Zeng
Stanford University
`czeng2@cs.stanford.edu`

Abstract

As emojis have emerged as an increasingly important and standardized part of modern textual inputs, analyzing their usage patterns are becoming of great interest to any modern text system. We frame this investigation as a text classification problem, mapping input text to their most likely accompanying emoji, utilizing 1.5 million scraped Twitter tweets for our training set. Both our LSTM-RNN model and our CNN model outperform our baseline, although the CNN model surprisingly achieves much better accuracy and F1 results. We conclude the paper by proposing future works in this area.

1 Introduction

With the rise of mobile devices and chat apps, emojis have become an increasingly important part of written language. The ‘Face with Tears of Joy’(😂) emoji was even selected as the Oxford Dictionaries Word of the Year in 2015[1]. To keep up with the evolution of language, we should be able to model emoji usage based on trends seen on Twitter, Reddit, and other online Social Networks. In addition, the standardization of emojis in the Unicode Standard has ushered in more standardized usage patterns in the different social networks, which makes modeling usage patterns a significantly more tractable problem.

As one of the largest social network platforms, Twitter has had a long history with emojis, and currently, around 19.6% of all generic tweets on Twitter contains emoji[2]. The high frequency of emoji usage and the diversity of topics makes Twitter an ideal data source for building a model for emoji usage.

More concretely, we frame our investigation as a multi-class classification problem. Given a input tweet with sequence of words $w_1, w_2, ..., w_l$, where l is the length of the tweet, we aim to predict an emoji class $\hat{e}$ from the set of K emojis. In training, the input tweet is constructed by removing an emoji from the raw tweet, with removed emoji assigned as the label(more in ”Data” section). We will then evaluate our model on a separate validation set and calculate various standard classification metrics such as accuracy and F1 score for evaluation.

As the dominant deep learning model for sequential data, Recurrent Neural Networks(RNN) with Long-Term Short Memory(LSTM) has proven to be very effective in various other closely related tasks, including sentiment classification[6] and language modelling[7]. For our task, we will be building a LSTM RNN network and evaluating its performance with a baseline. In addition, we will also build a Convolutional Neural Networks(CNN) for this task, which have shown to be surprisingly effective in various text classification tasks settings[8].

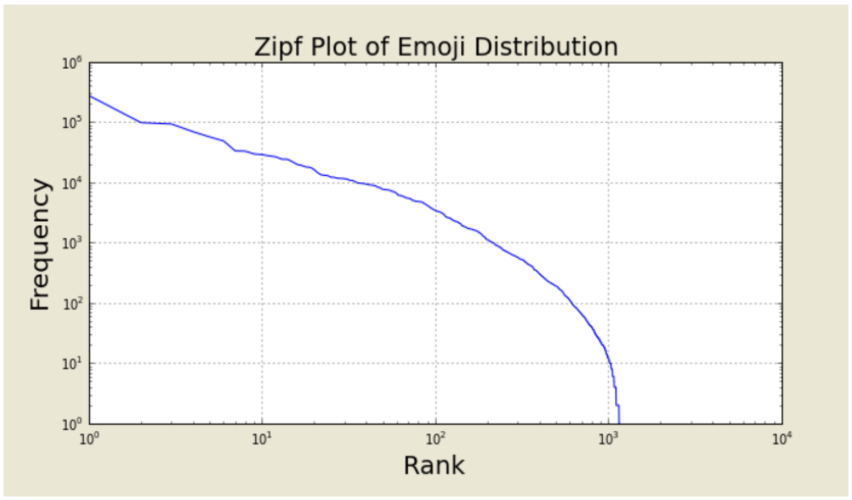
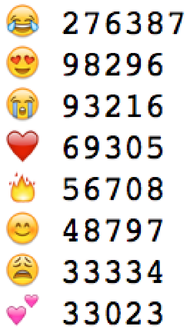


Figure 1: Emoji relative frequencies, ranked by most freq. to least



😂	276387
😍	98296
😭	93216
❤️	69305
🔥	56708
😊	48797
😩	33334
💕	33023

Figure 2: Top emojis in dataset, raw counts

2 Prior Work

There has been prior work on predicting emoji usage. Barbieri et al.[4] use a skip-gram model to create embeddings for words and emoji, which are then used in similarity-matching process to produce predictions. A team at Dango extends this work, building an ”emoji assistant” that provides real-time suggestions for emoji to include in a chat message, based on the written content. It uses an RNN to convert sentences and emoji to embeddings, and then finds the emoji with the highest cosine similarity to each sentence[3]. Similar work was done in Cappallo et. al.[**?**] These investigations establishes the feasibility of the task, and we follows similar paths in building our RNN network.

3 Data

We obtained a set of tweets dating from June 2016 from the Internet Archives[2]. The tweets were obtained by the Twitter Stream API and were not filtered in any way. Thus, we performed a series of filtering steps to select tweets for our data. First, we selected only tweets containing at least one emoji. Here, we explicitly define our set of emoji as the 1624 emojis defined by the Unicode v6.0 Standard[1]. Roughly 20% of the tweets are selected by this criteria. We then filtered our tweets using a language heuristic, to select only English tweets. This resulted in our raw dataset of 1.5 million tweets. The distribution of the frequencies of emojis within our is shown here 1. The top emojis by raw count is shown here 2.

We then performed a series of preprocessing on the dataset, which included punctuation stripping, misspelled word correction, the removal of re-tweet symbols, and the removal of URLs and other

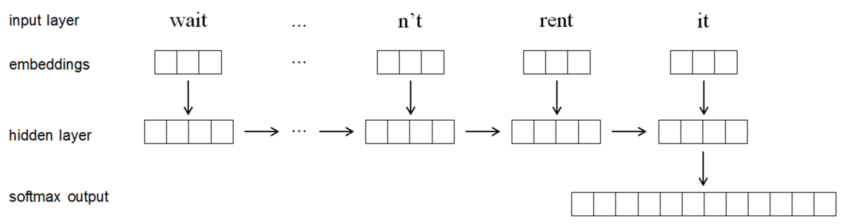


Figure 3: LSTM schematic

non-standard elements. We removed rare words by replacing every word with a total word count of less than 5 to the <UNK> token. After these steps, we were able to obtain the final vocabulary size of 23045.

Around 38.32% of the tweets contains more than one emojis. To create our training labels of one emoji per tweet, we used the following simple heuristic: We use the emoji that occurs the latest in the tweet. Our hypothesis is that the last-occurring emoji has the best chance of ”summarizing” the tweet. After extracting the last emoji as the label, we filter all emojis of the same type for the given tweet.

As we were in the process of building our model, we decided to simplify the classification task by only predicting from the top 50 emojis by frequency. Thus, we took a subset of our dataset which only contains at least 1 emoji from the 50 top emojis. This reduces our dataset to 1032831 tweets.

4 Approach

4.1 Baseline Classifier

As a simple baseline, we trained a logistic regression classifier. Tweets were featurized as TF-IDF vectors, with sublinear TF scaling due to the unbalanced distribution of the emojis. The classifier was trained with a one-vs-all strategy.

4.2 LSTM Classifier

The main network architecture we worked on was a recurrent neural network (RNN) with one or more Long-Short Term Memory (LSTM) hidden layer and softmax output layer. The LSTM unit performs the following calculations

$$\begin{aligned}
i_t &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b_i) \\
f_t &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b_f) \\
o_t &= \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b_o) \\
\tilde{c}_t &= \tanh(W^{(c)}x_t + U^{(c)}h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}$$

and the overall network can be represented as Figure 3.

To feed into the network, each word was represented as a trainable 50-dimensional embedding vector. Tweets were input in batch sizes of 32 and either truncated or padded to a length of 120 words(the maximum tweets length), with full back-propagation through the entire tweet. Dropout regularization was included during training to prevent over-fitting.

Three versions of the LSTM were compared:

(a) **Single-Layer, Random Embeddings**:
The initial model was a single-layer LSTM with randomly initialized word embeddings.

(b) **Single-Layer, Pretrained Twitter Glove Embeddings**:
Embeddings were instead initialized with GloVe Twitter word embeddings [9] to provide a more informative starting point.

(c) **Double-Layer, Pretrained Twitter Glove Embeddings**:
A second LSTM hidden layer was added, to potentially capture deeper or more complex dependencies.

4.3 CNN classifier

A convolutional neural network (CNN) was implemented based on Kim et. al 2012 [8]. see Figure 4. Similarly to the RNN, we utilize word embeddings for each word in the vocabulary, but in this configuration, we reshape each sentence into a 2-D matrix of word embeddings. Let $x_i \in R^k$ be the k-dimensional word embeddings corresponding to the i-th word in the tweet. The input can be represented as:

$$x_{1:n} = x_1 \oplus x_2 \oplus .. \oplus x_n$$

where n is the length of the tweet.

We then use 64 filters of size $hx50$ (where 50 is the size of the embedding) to slide over the embeddings and output features c_i.

$$c_i = f(w * x_{i\ i+h-1} + b)$$

We use h of size 3, 4, and 5.

The features c_i are concatenated, and a max_pool is applied per individual filter horizontal slides to obtain $\hat{c} = max\{c\}$

Then, these features passed to a fully connected softmax layer whose output is the probability distribution over the emojis.

Dropout is also performed for regularization purposes. The static vs. dynamic word channels mentioned in Kim et. al. are not implemented here for simplicity[8].

4.4 Evaluation Metrics

We evaluated our models against our baseline using standard metrics of accuracy and multi-class weighted F1 score.

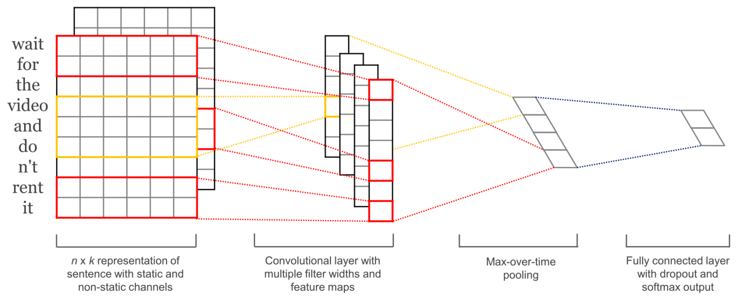

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 4: CNN schematic

5 Results

The classification statistics from all models are compared in Table 1. Some example predictions are shown in 2.

	Baseline	1-layer LSTM, random embed	1-layer LSTM, GloVe	2-layer LSTM, GloVe	CNN
Accuracy	0.271	0.370	0.378	0.353	0.403
F1 score*	0.170	0.315	0.327	0.294	0.458
Precision*	0.380	0.441	0.433	0.409	0.645
Recall*	0.270	0.370	0.378	0.353	0.403

Table 1: Comparison of different models. Summaries of the overall accuracy, F1 score, precision, and recall for classification on the top 50 emoji. The multiclass metrics (*) are weighted by support.

Table 2: Example Predictions

Text Input	Predicted Emoji
Im tired but I cant sleep	😩
Get into a relationship with someone whos gonna rep you right.	💕
Ill be lying if I said I wasnt missing you,Love you Brah	😘

The baseline model achieved an accuracy of 0.271 and a F1 score of only 0.17. It almost always predicted the most common emoji label (‘Face with Tears of Joy’(😂)), which drove down the precision for that class while losing recall on the remaining classes.

The LSTM networks provided some improvement, with an accuracy of 0.378 and a F1 scores of about 0.3.

Incorporating the GloVe vectors offered a small boost in performance, but did not have as much of an impact as expected. This could be a result of using different Twitter datasets, or it might indicate that emojis are less nicely correlated with word occurrences.

The CNN achieved the best score out of all the models, with an accuracy of 0.403 and F1 score of 0.45.

The confusion matrix for the top 50 emojis is shown in Figure 5. The density along the diagonal of the matrix indicates that most predictions were aligned to the correct class, but still the predictions were still biased toward the more common classes. Taking a closer look at the top 10 emojis in Figure 6, we can make sense of some of the incorrect predictions. For instance, the ’Red Heart’ emoji is most commonly confused with ‘Two Hearts’(💕), ’Face Blowing a Kiss’(😘), and ‘Smiling Face With Heart-eyes’(😍) which are all somewhat related and similar in meaning. The ’Loudly Crying Face’(😭) is also frequently predicted as ’Face with Tears of Joy’(😂), and although the two emojis were designed with opposite sentiments, they can be confused by users and are commonly used in similar contexts.

6 Conclusion/Future Work

Although our initial work is promising, the models investigated still have significant room to improve. One major part of the difficulty in this task was working with a noisy dataset. Tweets can be quite unstructured, especially with emojis being used much more loosely than real words. People have different interpretations of emojis, and often combine multiple emojis without much organization. Cleaning up the dataset to remove nonsensical tweets may help improve the classification performance. Secondly, since people can use emojis for different purposes other than summarization of a tweets(for example, to remove key verbs such as ”heart” for ”love”), there are some inherent limitations to the setup of using whole tweets to predicts emojis. Training a language-model like model which does word-by-word prediction would overcome some of these limitations. Lastly, it is worth considering different selection heuristics for determining the true label from multiple emojis in a tweet, such as selecting the first emoji or the most frequent emoji within a tweet.

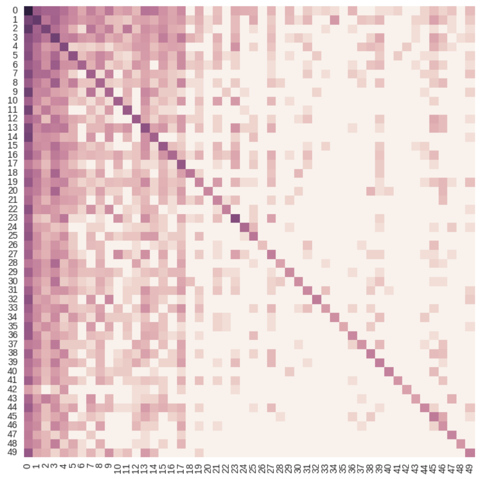


Figure 5: Confusion matrix of top 50 emojis

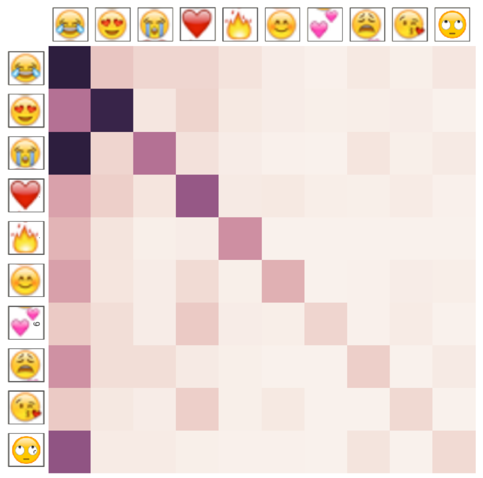


Figure 6: Confusion matrix of top 10 emojis

References

[1] http://time.com/4114886/oxford-word-of-the-year-2015-emoji/

[2] https://archive.org/details/twitterstream

[3] Snelgrove, Xavier. *"Dango - Your Emoji Assistant."* Dango - Emoji & GIF Assistant App. Whirlscope Inc., n.d. Web. 09 Feb. 2017.

[4] Barbieri, Francesco, Francesco Ronzano, and Horacio Saggion. *"What does this Emoji Mean? A Vector Space Skip-Gram Model for Twitter Emojis."* Language Resources and Evaluation conference, LREC, Portoroz, Slovenia. 2016.

[5] Cappallo, Spencer, Thomas Mensink, and Cees GM Snoek. *"Image2emoji: Zero-shot emoji prediction for visual media."* Proceedings of the 23rd ACM international conference on Multimedia. ACM, 2015.

[6] Socher, Richard, et al. *"Recursive deep models for semantic compositionality over a sentiment treebank."* Proceedings of the conference on empirical methods in natural language processing (EMNLP). Vol. 1631. 2013.

[7] Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney. *"LSTM Neural Networks for Language Modeling."* Interspeech. 2012.

[8] Kim, Yoon. *"Convolutional neural networks for sentence classification."* arXiv preprint arXiv:1408.5882 (2014).

[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. *GloVe: Global Vectors for Word Representation*