Emojify: Emoji Prediction from Sentence

Chen Huang
Stanford University
chuang4@stanford.edu

Xueying (Shirley) Xie
Stanford University
xueyingx@stanford.edu

Boyu (Bill) Zhang
Stanford University
bzhang99@stanfor.edu

Abstract

Emojis are small images that are commonly included in social media text messages. The combination of visual and textual content in the same message builds up a modern way of communication. Despite being widely used in social media, emojis' underlying semantics have received little attention from a Natural Language Processing standpoint. In this project, we investigate the relation between words and emojis, studying the novel task of predicting which emojis are evoked by text-based tweet messages. We experimented variant of word embedding techniques, and train several models based on Multinomial Naive Bayes and LSTMs in this task respectively. Our experimental results show that our model can predict reasonable emoji from tweets.

1. Introduction

People use emojis every day**??**. Emojis have become a new language that can more effectively express an idea or emotion. This visual language is now a standard for online communication, available not only in Twitter, but also in other large online platform such as Facebook and Instagram. Right now, the keyboard on iOS can predict emojis but only base on certain keywords and tags that are associated with emojis.

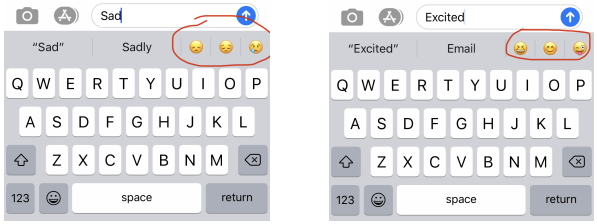

Figure 1: Examples of emoji prediction in iOS13 keyboard

Emoji prediction is a fun variant of sentiment analysis. When texting your friends, emoji can make your text messages more expressive. It would be nice if the keyboard can predict emojis based on the emotion and meaning of the whole sentence you typed out.

Moreover, despite its status as language form, emojis have been so far scarcely studied from a Natural Language Processing (NLP) standpoint. The interplay between text-based messages and and emojis remains virtually unexplored.

In this application project, we aims to fill this gap by investigating the relation between words and emojis, studying the problem of predicting which emojis are evoked by text-based tweet message. We build classifiers that learns to associate emojis with sentences. The models we explore here is Multinomial Naive Bayes and Bidirectional LSTM. Standard Bag of Word TF-IDF and pre-trained GLoVe model are used as word embedding, respectively. We train a large dataset of sentences with emojis labels aggregated from Twitter messages. In the Inference stage, the trained classifier takes as input a sentence and finds the most appropriate emoji to be used with this sentence.

2. Related Work

2.1. Emojis

Emojis, also known as ideograms or smileys, can be used as compact expressions of objects, topics and emotions. Being encoded in Unicode, they have no language barriers and are diffused on the Internet rapidly. The prevalence of emojis has attracted researchers from various research communities such as NLP, multimedia and data mining [6]. The various non-verbal functions of emojis play an important role in their wide adoption to the extent that they can have their unique linguistic purpose alongside written text [5]. Generate emojis automatically from a sentence is essentially a text classification problem. In [1], the author use millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm.

2.2. Sentiment Analysis

sentiment analysis refers to use the NLP techniques to study the subjective informations from a sentence. Currently, sentiment analysis is a topic of great interest and de-

velopment, since it can be widely used for getting insights from social media comments, survey responses, product reviews, and making data-driven decisions. Some applications, such as recommender systems employ binary classification to determine if a user expresses like or dislike of some product or service or even polarity [4] of a statement. Other forms of sentiment analysis use unique, categorical labels similar to the emoji tags to predict on those distinct emotions, such as happy, confused, tired, surprised, etc. Since emojis have multi-contextual representation and is readily used across all languages [2], it serves as a great sentiment label that can encapsulate the nuances in a sentence.

3. Dataset and Features

3.1. Dataset

Given the prevalence of emoji usage and digital opinions on Twitter, it was obvious to extract a rich dataset from Twitter. The Twemoji dataset provides an extensive set of almost 13 million tweet status ID and emoji annotation per tweet. We fetched the corresponding tweets through the twitter API with the provided tweet ID to aggregate our dataset of sentence-emoji ID pairs. The data collection process is shown in Fig 2.

Tweet ID	Emoji ID(s)
743318311333572608	1447
748211447319134208	586
742068066952065025	1447

Twitter API →

id_str	Emoji ID(s)
man i hate when that happens	1457
pasta for my bfast	586
honestly guys i m drunk at work	1447

Figure 2: Twitter data API

3.2. Pre-processing

From the Twemoji dataset, we extracted 167,685 ¡sentence - emoji¿ pairs for our training. However, the dataset is heavily unevenly distributed, as shown in Fig 3, where emoji has over 50% representation power of the whole dataset. So we have done several data preprocessing steps before training.

- **noise removal**, filter out emojis which has less than 10000 correspondence sentence.
- **stopemojis**, like the stopwords technique in NLP, remove high frequent emojis which are everywhere and do not have specific semantic meaning.
- **data un-bias**, equalize the number of sentences for each emoji.

After the data pre-processing, the dataset is reduced to 13251 valid examples which belongs to 13 emoji categories. The emoji we used in our project is shown in Fig 4.

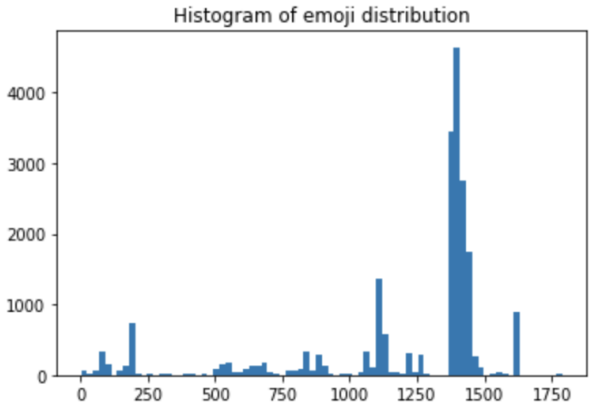

Figure 3: Raw data class distribution

Unnamed: 0	category	title	shorts
👌	people	ok hand sign type	[ok_hand_tone4]
💔	symbols	broken heart	[broken_heart]
💗	symbols	growing heart	[heartpulse]
😂	people	face with tears of joy	[joy]
😆	people	smiling face with open mouth and tightly close...	[laughing, satisfied]
😉	people	winking face	[wink]
😙	people	kissing face with smiling eyes	[kissing_smiling_eyes]
😚	people	kissing face with closed eyes	[kissing_closed_eyes]
😜	people	face with stuck out tongue and winking eye	[stuck_out_tongue_winking_eye]
😨	people	fearful face	[fearful]
😩	people	weary face	[weary]
😲	people	astonished face	[astonished]
🙀	people	weary cat face	[scream_cat]

Figure 4: Emoji category after data pre-processing

4. Method

Predicting emojis is more like a sentence classification problem. We first investigate the word embedding techniques to use, then try machine learning algorithms we learned in the class, such as Multinomial Naive Bayes, SVM and Bi-LSTM to do the emoji prediction.

4.1. Word Embedding

BoW-TFIDF Instead of representing the sentences with matrix of corpus and counts like we've seen in class, we can employ Bag of Words representation with TD-IDF trying to capture more indicative keywords in a sentence. The dictionary size is 1834 after stopwords and stemming. We can feed this into our traditional methods for Naive Bayes and SVM.

$$W_{i,j} = tf_{i,j} \times \log(\frac{N}{df_i}) \quad (1)$$

where $tf_{i,j}$ is the number of occurrences of i in j, df_i is the number of documents containing i, while N is the total number of documents in corpus.

GLoVe For the deep learning and SVM methods, we tried to use the pre-trained GLoVe to see if adding in global statistics of our Twitter corpus would help with neural net performance. GLoVe is an unsupervised trained model which mapping words into a meaningful space where the distance between words is related to semantic similarity. In this project we used the pre-trained model GLoVe-50 and GLoVe-300 from Stanford. From there, we split our dataset into 90% training and 10% test.

4.2. Multinomial Naive Bayes Classifier

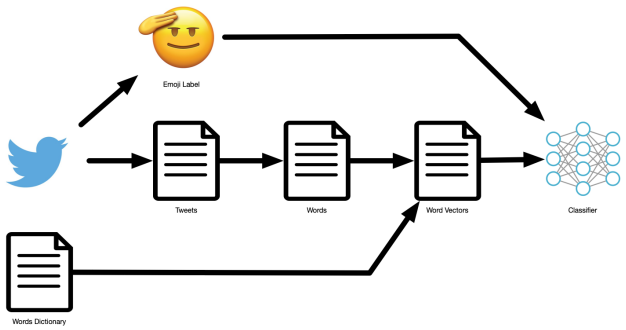

Figure 5: Overall pipeline

The whole pipeline for NB classifier is shown as Fig 5. From the twitter we can collect data which contains tweets message and its corresponding emoji as label. Then by using a word dictionary we transform texts into word vector via bag of words with td-idf as described above. Finally here we apply a Multinomial Naive Bayes Classifier to train the model for text classification. Since Naive Bayes is a simple classifier to use, it acts as a good baseline and starting point to tackle this problem.

$$\phi_{k|y=h} = \frac{1 + \sum_{i=1}^{n} \sum_{j=1}^{d_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = h\}}{|V| + \sum_{i=1}^{n} 1\{y^{(i)} = h\} d_i} \forall h \in 1...13 \quad (2)$$

Similarly, we also employed the pipeline to SVM.

4.3. Bi-LSTM Classifier

For the RNN model, the first layer is embedding layer, embedding layer will represent each word as a matrix. The embedding layer is loaded from a Pre-trained GLoVe model and the weight is fixed during training. The Glove will map similar word to close vector in high-dimensional feature space. The next layers are two 1-D convolution layers, followed by a bidirectional LSTM layer. LSTM is a type of RNN network. It allows previous outputs to be used as inputs while having hidden states, so it is suitable for analyzing sentences. RNN suffers from the vanishing gradient problem which can be handled by LSTM. Lastly, we use a softmax layer which outputs a vector that represents the probability distribution of a list of potential emoji. The network architecture is shown in Fig 6

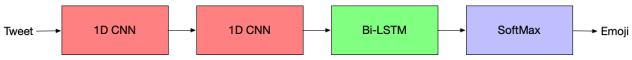

Figure 6: LSTM Architecture

The code is implemtened in Keras. After each convolution and LSTM layer, ReLU activation and batch normalization is used. We use Adam as optimizer, the learning rate is 0.001. The LSTM layers have a L2 regularization with rate to be 0.01 to prevent over-fitting. The training batch size is 128 and we trained 100 epochs.

5. Experiments

5.1. overall accuracy

Right now the twitter dataset we collected contains 12,997,220 tweets, and there are 1,791 emojis associate with these tweets. From our preprocessing, we reduced it to 167,685 tweets and 13 emojis. To evaluate the model, we split 90% tweets into training set and 10% into testing set. The results from test sets will be assessed using accuracy score. Our current accuracy on these dataset is shown in Tab 1

Word Embedding	BoW	GLoVe-50	GLoVe-300
Multinomial NB	19.53%	N/A	N/A
SVM	9.195%	16.376%	14.966%
Deep CNN	N/A	15.168%	15.906%
Deep Bi-LSTM	N/A	15.705%	15.57%

Table 1: Emoji prediction accuracy

5.2. Result Analysis

Here we did some further analysis on each method.

5.2.1 Naive Bayes and SVM Classifiers:

Here is some analysis on the performance of Naive Bayes Classifier. The model we choose is multinomial Naive Bayes model because it can take word frequency into account. The word dictionary's size is 2791, which is created from just training set.

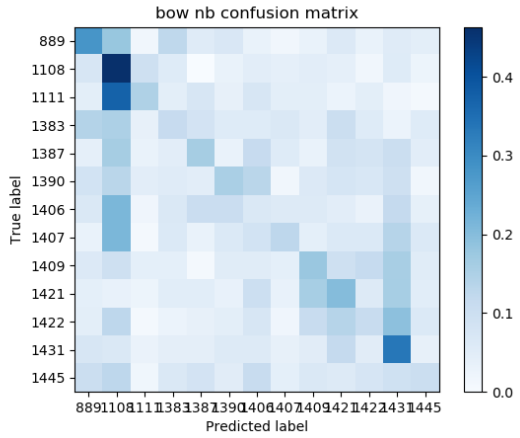

Figure 7: Confusion Matrix for Naive Bayes

The overall NB classification accuracy is 19.53%, while SVM did quite poorly with only 9.1% on bag of words embeddings and much better on GLoVe embeddings, as shown in Fig 7 and 8.

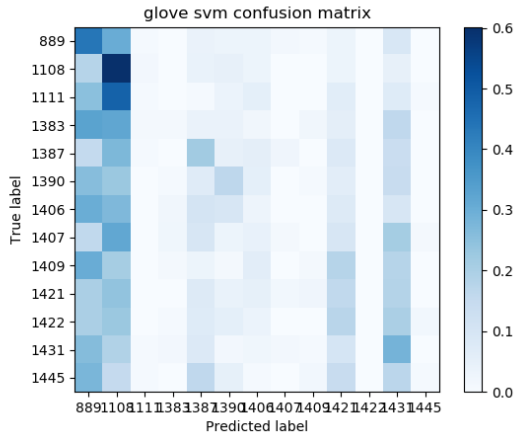

Figure 8: Confusion Matrix for SVM

Here is some prediction examples using this trained model. Some make sense and some are just wrong.

Emoji	Predicted Emoji
I'm angry	😩
I need sleep	😞
love you	❤
sad face	😩

Figure 9: prediction results from naive bayes classifier

In addition, we also know that if we further reduce the number of emojis to classify with naive bayes and thereby create an even more balanced distribution of the dataset, say to top 5 balanced classes with sufficient data, we see that the test accuracy can improve up to 41%.

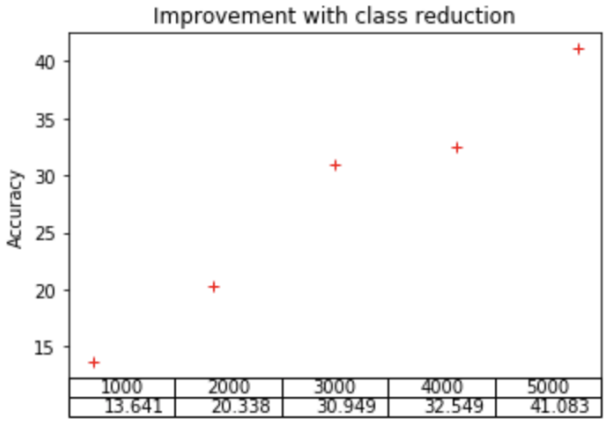

Figure 10: Accuracy improvement after reduce categories

5.2.2 Bi-LSTM Classifiers

We also show the results of Bi-LSTM with GloVe-300 confusion matrix in Fit 11.

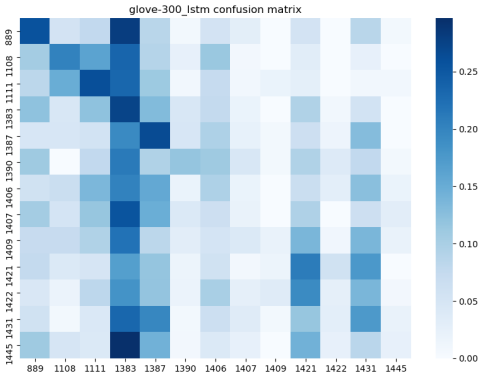

Figure 11: Confusion Matrix for LSTM with GLoVe-300

6. Conclusion

From the experiments outlined above, multinomial naive bayes performed the best among all the methods analyzed. We know from our problems sets and lectures that multinomial NB is very good for text classification, such as spam classification, so we had high confidence that naive bayes

would provide reasonable results. For the deep learning approaches, we most likely could have tried BERT or XL-Net[7] to see if it can overcome the weak semantics. More importantly, many of the related topics on this work employs data on the factor of billions to solve this problem, while we only used less than 200,000 data in total. Therefore, the data isn't generalizable enough for deep learning to outperform traditional methods.

We did discover that stop emoji is essential to handle uneven distribution. When one emoji dominates the dataset, it will makes the classifiers to prefer this emoji. Due to the nature of this problem, the emoji and sentence only have weak semantic relations, wherein any examples which share the same emoji actually express totally opposite emotion.

For future improvements, we should employ an output of more than one prediction with decreasing confidence instead of an absolute single emoji. There often isn't a 1:1 mapping between an emoji and a sentence or expression. Oftentimes, if we have a user decide on which emojis to use for a similar sentence, the emoji selection would vary quite a bit. In addition, when we calculate accuracy, it may be best to have weighted penalties for determining accuracy. The correlation matrix portrays a lot of the emoji overlap with some good reasoning. Currently our accuracy is just based on absolute correctness, which a very hard ask for a model given the weak semantic nature of this problem.

References