CSE 5525 Project Report

Guoyao Cheng, William McMurtry, Ningli Xu, Jason Yao

Introduction

As the scale and prevalence of social media applications grow rapidly all over the world, emoji has become a new and popular approach people use to express their emotions. Studying how emojis are correlated with texts (e.g. tweets) posted on social media along with them is accordingly an interesting topic, which can be achieved using Natural Language Processing methodologies. In this project, our team studied a text classification problem involving tweets and emojis, used several approaches to solve it, and analyzed the results.

Problem description

The idea was presented in 2018, by a CodaLab competition: SemEval-2018 Task 2, Multilingual Emoji Prediction. Generally speaking, the task is: given only the text content of a tweet, predict the single emoji used in this tweet. Since there are multiple types of emoji in the social media, obviously this is a text multi-classification problem, with the tweet text as the input data and the emoji used as the class to predict.

The original dataset was provided with available validation and test data and labels in text files, and a Python crawler that downloads the large scale training data. Tweets in both English and Spanish were provided, which are naturally split into 2 sub-problems. In the scope of this project, we focused on the tweets in English, and experimented on the whole dataset using the chosen methods.

Dataset

There are 20 emoji in total we need to predict. Each tweet only has one emoji so this is a multiclass classification. Each emoji is mapped to its index from 0 to 20. For our task we focus on English tweets.

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

The data include train, test and trial set. Test set and trial set both include 50k tweets. Training set includes 500k. But only ~420,000 of the 500,000 tweets from training data were available. The tweets were retrieved with the Twitter APIs, from October 2015 to February 2017. Since this is an old task, perhaps some tweets were deleted. Due to Twitter restrictions, the training set were downloaded from the twitter developer account using a web crawler and an emoji extractor

Methodology

1. Baseline: random prediction

First, we chose random prediction to be the most straightforward baseline. In order to improve the blind random prediction a little, we incorporated weights which were computed based on the frequency of appearance of each class (i.e. emoji) in the dataset, namely, a more frequent emoji will be more likely to be predicted.

2. Baseline: Feed-Forward Neural Network

Another baseline we chose is the Feed-Forward Neural Network (FFNN), which is relatively simpler than the other neural network models we chose. Basically, we implemented an FFNN model that mostly resembles the FFNN model as implemented in one of the homeworks, with an embedding layer and a simple hidden layer, followed by activation function and a linear output layer with softmax probabilities.

3. LSTM

The network use Pytorch embedding layer with 1024 dimension as the input to a LSTM layer with 20 dimension hidden layer. Then the output of LSTM layer goes to nn.CrossEntropyLoss() which contains a softmax layer and an NLLLoss that compute the negative log likelihood loss.

4. BERT+LSTM

BERT (Bidirectional Encoder Representations from Transformers) is the first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus. We used BERT model to obtain pre-trained embeddings with 1024 dimension as the embedding layer, and the other network structures are the same as the LSTM model mentioned above.

Evaluation metrics:

The original competition used the macro F-score as the official evaluation metric. In addition, we also used macro precision, macro recall and accuracy for reference of comparing different methods we used. The reason why macro scores were chosen is that according to the expectation of the competition, "the fundamental idea of this task is to encourage systems to perform well overall", not just on those most common emojis.

Result

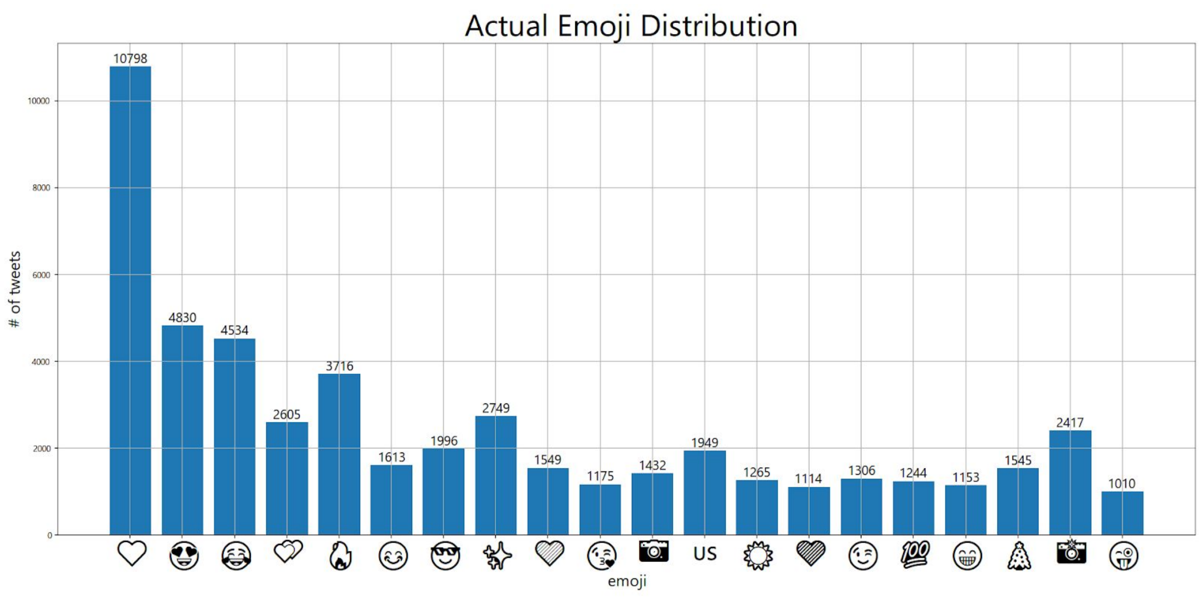
For detailed evaluation scores of each method used, see Table 1 in appendix. For the pytorch embedding+ LSTM we get F-score 6.368. For the pytorch embedding, any word that was in the test set vocabulary but not in the train vocabulary was counted as "<UNK>" (unknown). May have impacted the score in a negative way. The best performance we achieved is from the BERT+LSTM model, with 13.795 on F-score and 29.932 on accuracy.

Note that so far the best ever performance in the competition is 35.991 on F-score and 47.094 on accuracy, therefore naturally this is not an easy task. Some possible reasons are: the dataset is unbalanced with respect to labels, which can lead to unbalanced training and is not favorable for macro metrics; some emojis may have similar meanings, which are used interchangeably and hard to discriminate in real tweets; training time was limited, therefore exploration on network structure and hyperparameter tuning may not be sufficient; for some emojis, the logic in using them may not be so clear and deterministic. More analysis and illustrations will be presented in the error analysis section.

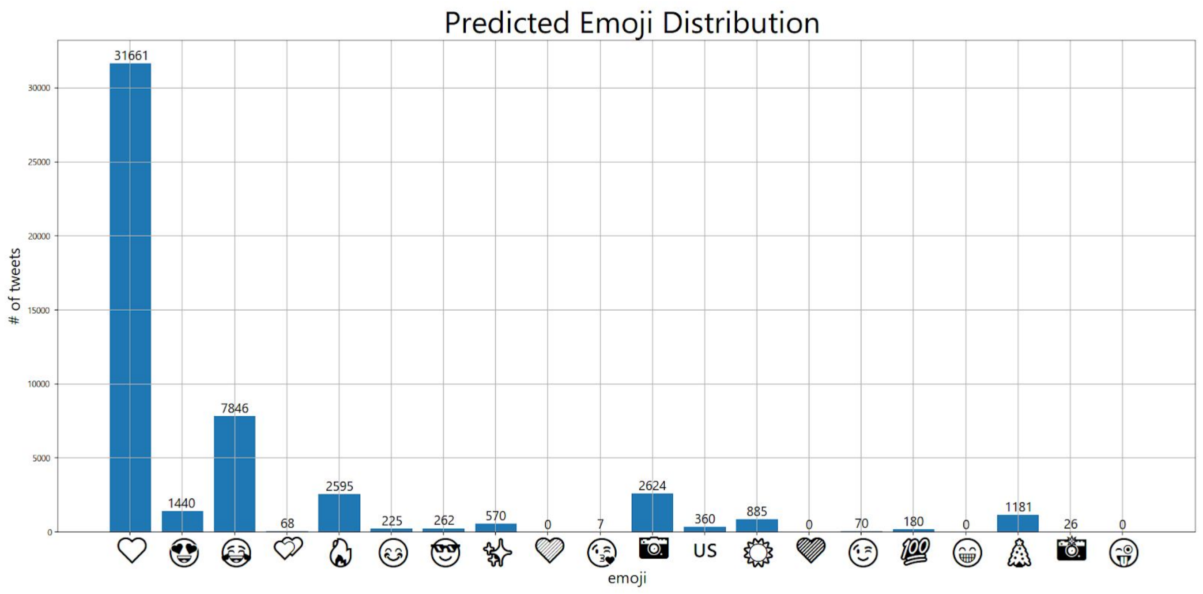
Error analysis over LSTM+BERT

Actual label distribution and predicted label distribution in the test set

(for detailed predicted label distribution over each emoji, see Table 2 in appendix)



There are 20 labels and 50k tweets in total. The label distribution shows that the top 5 highest number of emojis are red_heart❤ 10798, smiling_face_with_hearteyes 😍 4830, face_with_tears_of_joy😂 4534, fire🔥 3716, sparkles✨ 2749. The least used 3 emojis are face_blowing_a_kiss😘 1175, purple_heart💜 1114, winking_face_with_tongue😜 1010.



In the predicted distribution, a huge portion of the prediction go to the red heart. 💕,💙,💜,😘,😜,😁,😉,💯 are almost not predicted at all.

For donut chart distribution of predicted emojis over each actual emoji, see Figures in appendix.

Obeservation:

1. For each tweets no matter what emoji at least 30% probability to predict ❤.
2. For red heart, blue_heart, purple_heart, double_heart, face_blowing_a_kiss, United_Status which are ❤,💕,💙,💜,😘,🇺🇸 mainly predict ❤ (at least 80 percent, while 🇺🇸 have 8 percent predict right)
3. For each tweets 😂 is the second emoji that the model tends to predict unconditionally. Especially the model tends to always predict tweets with actual emoji 😜,😁,😉,😘 to 😂 except the effect of ❤

4. 😍 have 5.859% predicted correctly which is second highest accuracy among face emojis.
5. For tweets have emoji 📸, the model always predict 📷
6. 📷,🎄,🔥,☀ have some higher accuracy (more than 20%), while ✨,🇺🇸,💯 have relatively low accuracy

Reasoning:

1. The model cares too much about the frequency of emojis.
2. There are no too much context difference between emojis like 💕,💙,💜,😘 with ❤
3. There are no too much context difference between emojis like 😜,😁,😉,😎 with 😂, maybe because they all have meaning like joy, happy, laughing
4. There are no context difference between 📸 and 📷. People only use the two emoji interchangeably. Nearly no information can show the difference.
5. 📷,🎄,🔥,☀ have its special meaning which is shown in the context with some keywords maybe, while ✨,🇺🇸,💯 don't have strong keywords to show the difference.

Experiment:

Furthermore, we also tried removing the most dominant emoji (i.e. the red heart) from the dataset, to alleviate its imbalance, and trained the model on the other 19 classes. The result did show some but not much improvement, with F-score increased (19.916) but accuracy nearly the same (29.876). This is reasonable, as most of the improvement came from the recall (see appendix), indicating that removing red hearts did reduce some of the wrong predictions (false negatives with regard to other classes).

Another possible reason of the poor performance is that LSTM might not be a suitable approach for this task, since most of the real world tweets are short texts that do not strictly follow grammar and syntax rules.

Future work

According to error analysis, we can add some penalty to constrain the model so that the prediction will not be affected by the frequency of words. And we can modify the model in the direction that it can fully capture the context difference of words. If there are no indication of context difference between two emoji, then the prediction should be based on the frequency of the emoji.

Moreover, further exploration on network structure and hyperparameter tuning can be done in the future, in order to improve the current poor performance of our model. Some other machine learning models such as SVMs can also be employed, in case that simple LSTM may not be an appropriate method for this specific task.

References

[1] Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa-Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. (2018). SemEval-2018 Task 2: Multilingual Emoji Prediction. In Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), New Orleans, LA, United States. Association for Computational Linguistics.

[2] Çöltekin, Ç., & Rama, T. (2018). Tübingen-Oslo at SemEval-2018 Task 2: SVMs perform better than RNNs in Emoji Prediction. SemEval@NAACL-HLT.

Appendix

Table 1: Detailed evaluation scores of all methods

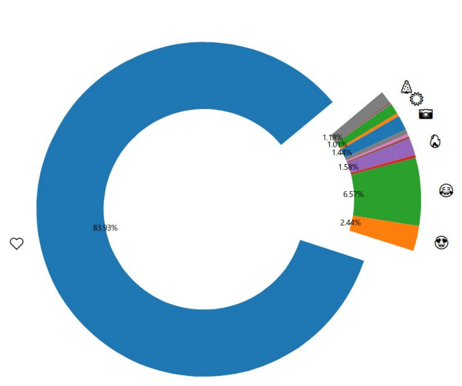
	Macro F-score	Macro precision	Macro recall	Accuracy
Random	4.97	5.023	5.028	8.884
FFNN	2.631	3.979	5.048	10.54
LSTM	6.368	14.503	8.446	25.396
BERT+LSTM	13.795	21.714	15.401	29.932
BERT+LSTM (red hearts removed)	19.916	22.531	22.441	29.876

Table 2: Predicted label distribution over each emoji (row is actual emoji, column is predicted emoji)

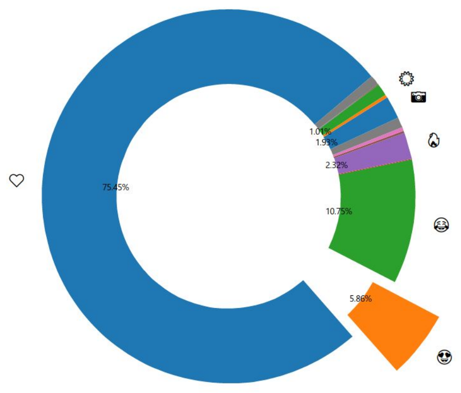
	❤	😍	😂	💕	🔥	😊	😎	✨	💙	😘	📷	us	☀	💜	😉	💯	😁	🎄	📸	😜
❤	83.932%	2.436%	6.566%	0.315%	1.584%	0.232%	0.241%	0.482%	0.0%	0.009%	1.445%	0.333%	1.009%	0.0%	0.065%	0.157%	0.0%	1.176%	0.019%	0.0%
😍	75.445%	5.859%	10.745%	0.062%	2.319%	0.166%	0.331%	0.911%	0.0%	0.0%	1.925%	0.269%	1.014%	0.0%	0.062%	0.021%	0.0%	0.87%	0.0%	0.0%
😂	36.745%	2.206%	52.272%	0.0%	3.088%	0.199%	0.706%	0.265%	0.0%	0.0%	2.272%	0.463%	0.772%	0.0%	0.11%	0.176%	0.0%	0.728%	0.0%	0.0%
💕	85.758%	2.303%	5.489%	0.422%	1.42%	0.115%	0.192%	0.806%	0.0%	0.0%	1.536%	0.154%	0.883%	0.0%	0.038%	0.115%	0.0%	0.768%	0.0%	0.0%
🔥	36.868%	2.637%	19.833%	0.0%	31.405%	0.323%	0.565%	1.346%	0.0%	0.027%	4.171%	0.538%	1.103%	0.0%	0.108%	0.646%	0.0%	0.323%	0.108%	0.0%
😊	69.746%	4.402%	13.949%	0.062%	2.108%	1.798%	0.992%	0.744%	0.0%	0.0%	1.736%	0.31%	2.542%	0.0%	0.31%	0.186%	0.0%	1.116%	0.0%	0.0%
😎	53.858%	3.357%	21.393%	0.05%	7.365%	1.353%	1.854%	1.353%	0.0%	0.0%	2.655%	0.601%	4.509%	0.0%	0.251%	0.501%	0.0%	0.752%	0.15%	0.0%
✨	66.351%	3.056%	9.931%	0.073%	5.056%	0.691%	0.4%	5.966%	0.0%	0.036%	3.674%	0.582%	1.31%	0.0%	0.182%	0.837%	0.0%	1.819%	0.036%	0.0%
💙	83.473%	2.195%	7.36%	0.194%	1.614%	0.323%	0.258%	0.968%	0.0%	0.0%	1.162%	0.452%	1.356%	0.0%	0.0%	0.129%	0.0%	0.452%	0.065%	0.0%
😘	77.702%	2.723%	9.787%	0.511%	2.809%	0.596%	0.34%	0.596%	0.0%	0.085%	1.447%	0.426%	1.362%	0.0%	0.255%	0.17%	0.0%	1.191%	0.0%	0.0%
📷	35.824%	1.536%	11.802%	0.0%	4.33%	0.279%	0.419%	1.117%	0.0%	0.0%	41.76%	0.279%	0.908%	0.0%	0.14%	0.419%	0.0%	0.978%	0.209%	0.0%
us	71.78%	1.847%	7.029%	0.103%	3.284%	0.359%	0.564%	0.77%	0.0%	0.0%	2.463%	8.415%	2.206%	0.0%	0.154%	0.257%	0.0%	0.718%	0.051%	0.0%
☀	59.051%	2.609%	7.668%	0.158%	2.609%	0.711%	1.265%	1.581%	0.0%	0.0%	2.451%	0.395%	21.028%	0.0%	0.079%	0.079%	0.0%	0.316%	0.0%	0.0%
💜	82.496%	3.411%	5.925%	0.269%	2.693%	0.18%	0.18%	0.898%	0.0%	0.0%	1.436%	0.718%	1.167%	0.0%	0.18%	0.09%	0.0%	0.359%	0.0%	0.0%
😉	52.297%	3.982%	30.398%	0.0%	2.603%	1.072%	1.455%	1.991%	0.0%	0.0%	2.067%	0.383%	1.455%	0.0%	0.613%	0.613%	0.0%	0.842%	0.23%	0.0%
💯	51.527%	1.125%	26.768%	0.0%	10.209%	0.482%	0.884%	0.804%	0.0%	0.0%	2.814%	0.563%	1.045%	0.0%	0.161%	3.457%	0.0%	0.161%	0.0%	0.0%
😁	57.849%	4.597%	24.545%	0.0%	3.729%	1.474%	0.781%	1.041%	0.0%	0.173%	2.168%	0.52%	0.954%	0.0%	0.173%	0.173%	0.0%	1.735%	0.087%	0.0%
🎄	39.353%	1.618%	5.178%	0.0%	0.906%	0.518%	0.065%	1.424%	0.0%	0.0%	0.971%	0.259%	0.712%	0.0%	0.129%	0.0%	0.0%	48.867%	0.0%	0.0%
📸	32.354%	1.614%	12.536%	0.0%	5.627%	0.29%	0.248%	1.117%	0.0%	0.0%	43.484%	0.414%	0.91%	0.0%	0.207%	0.496%	0.0%	0.414%	0.29%	0.0%
😜	49.01%	3.564%	34.554%	0.0%	4.653%	0.693%	0.891%	0.792%	0.0%	0.099%	1.386%	0.792%	1.287%	0.0%	0.495%	0.891%	0.0%	0.891%	0.0%	0.0%

Figures: Donut charts for predicted distribution over tweets of each actual emoji

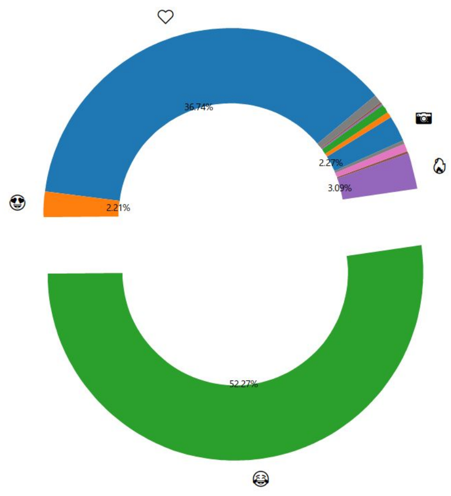
Predicted Distribution over tweets with acutal emoji ♡



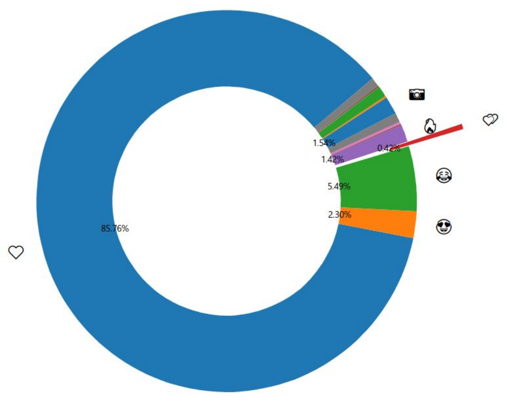
Predicted Distribution over tweets with acutal emoji 😍



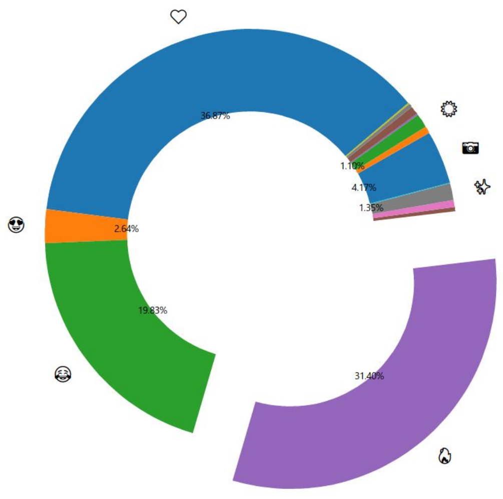
Predicted Distribution over tweets with acutal emoji 😂



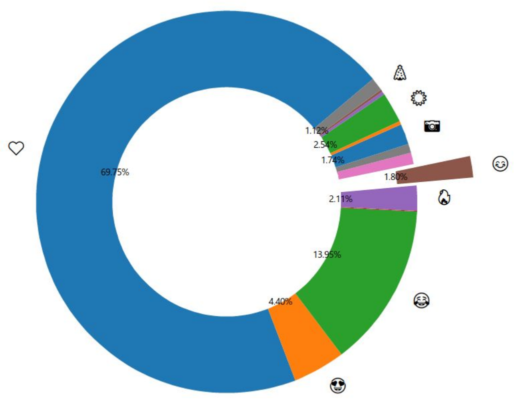
Predicted Distribution over tweets with acutal emoji 💞



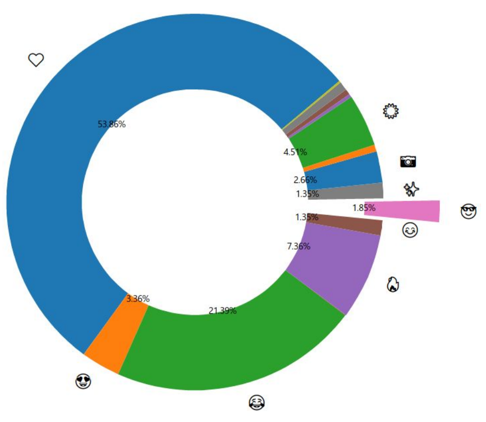
Predicted Distribution over tweets with acutal emoji 🔥



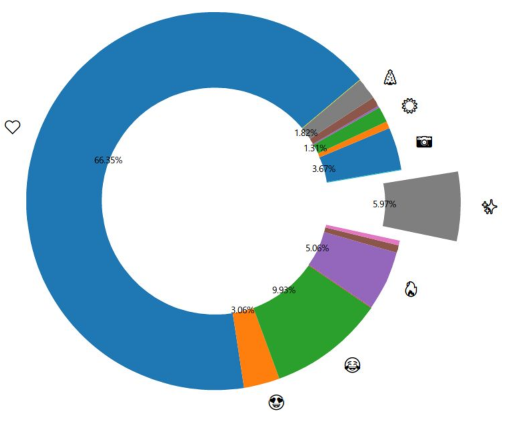
Predicted Distribution over tweets with acutal emoji 😉



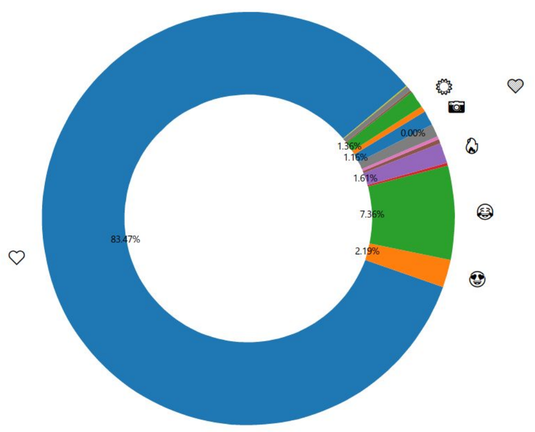
Predicted Distribution over tweets with acutal emoji 😎



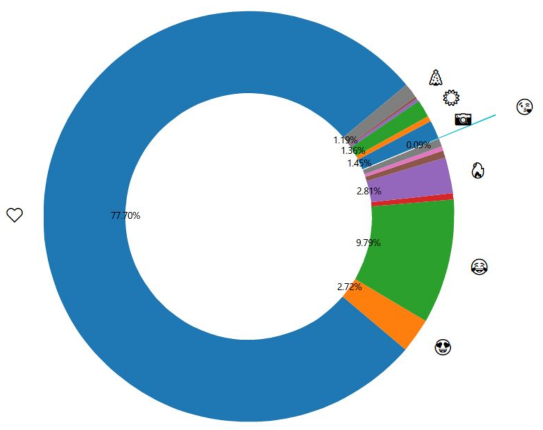
Predicted Distribution over tweets with acutal emoji ✨



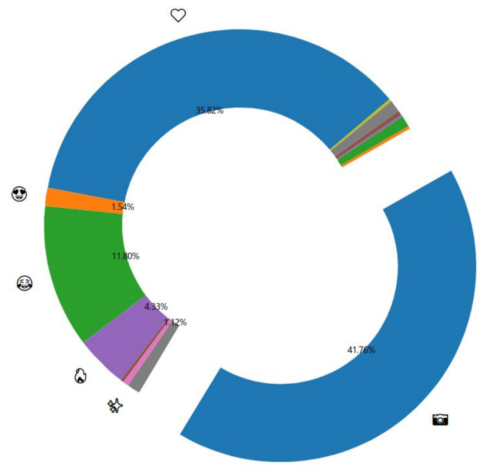
Predicted Distribution over tweets with acutal emoji ❤



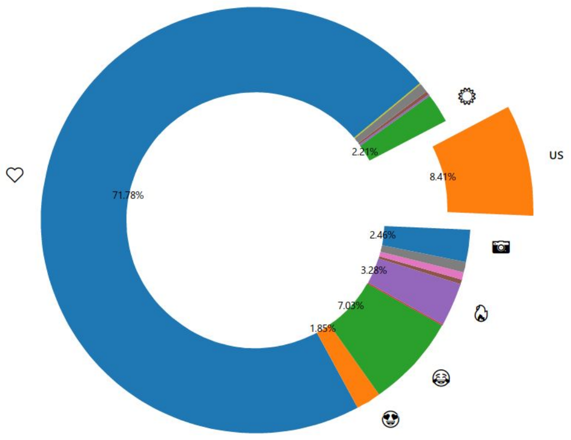
Predicted Distribution over tweets with acutal emoji 😘



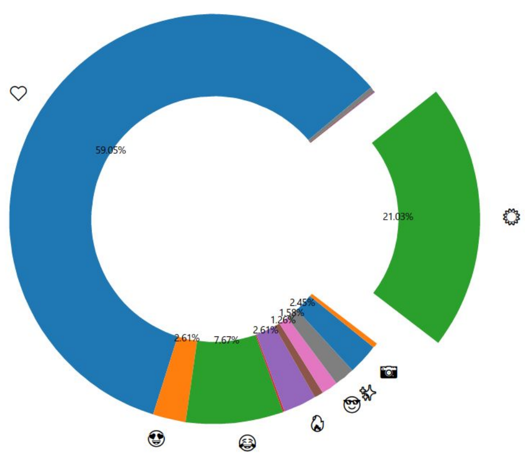
Predicted Distribution over tweets with acutal emoji 📷



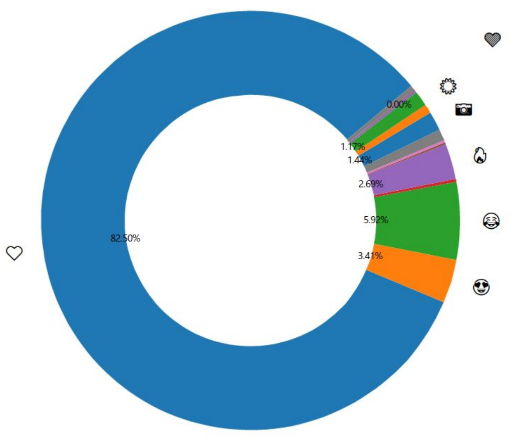
Predicted Distribution over tweets with acutal emoji us



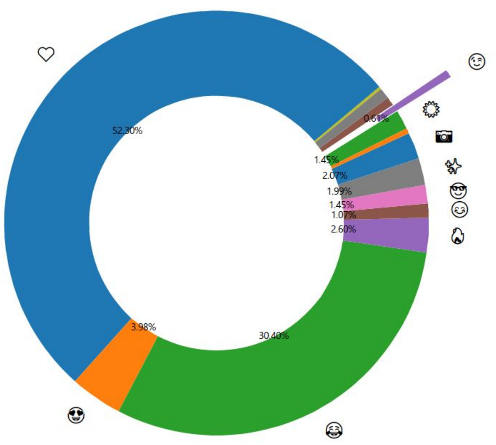
Predicted Distribution over tweets with acutal emoji ☀



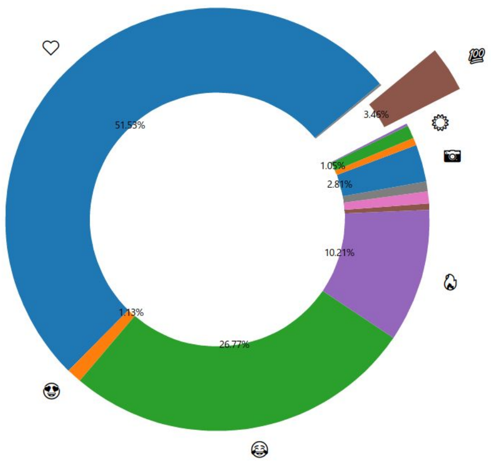
Predicted Distribution over tweets with acutal emoji 🖤

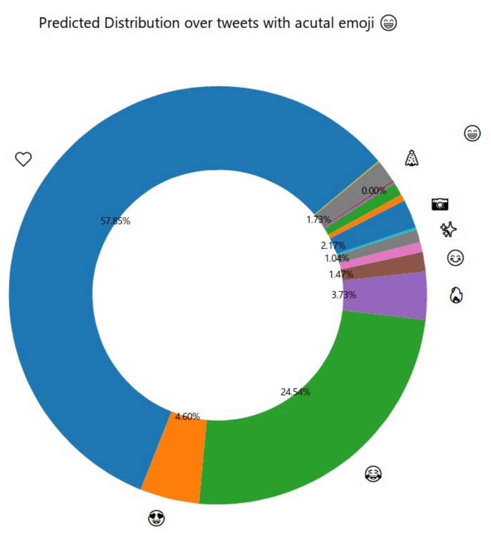


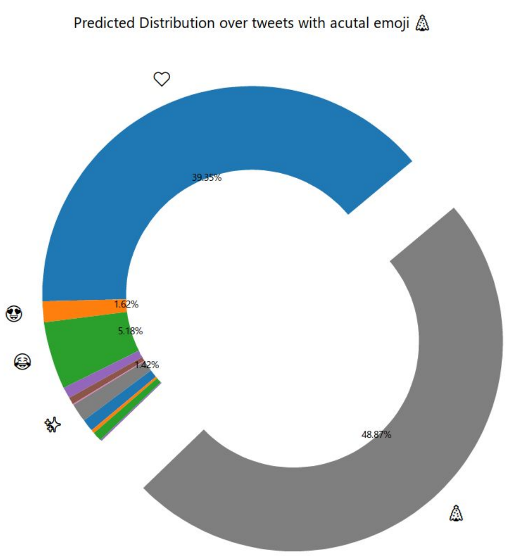
Predicted Distribution over tweets with acutal emoji 😉

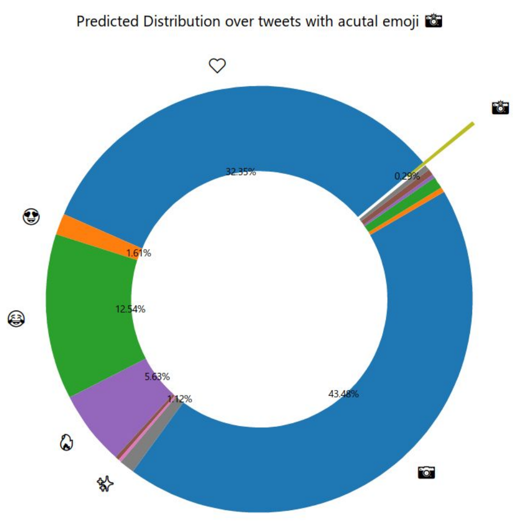


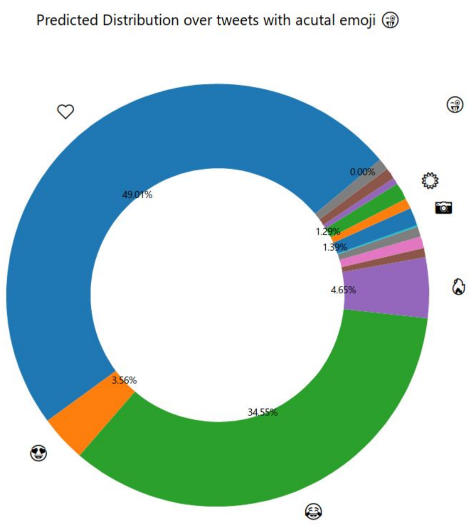
Predicted Distribution over tweets with acutal emoji 💯




Predicted Distribution over tweets with acutal emoji 😁
57.85%
0.00%
1.73%
2.17%
1.04%
1.47%
3.73%
24.54%
4.60%


Predicted Distribution over tweets with acutal emoji 🎄
39.35%
1.62%
5.18%
1.42%
48.87%


Predicted Distribution over tweets with acutal emoji 📸
32.35%
0.29%
1.61%
12.54%
5.63%
1.12%
43.48%


Predicted Distribution over tweets with acutal emoji 😜
49.01%
0.00%
1.29%
1.39%
4.65%
3.56%
34.55%