# Emoji Prediction with Transformer Models

Stanford CS224N Custom Project
TA Mentor: Benjamin Louis Newman, External Mentor: Gabriel Poesia

**Wenna Qin**
wennaqin@stanford.edu

**Jiacheng Ge**
kevinge1@stanford.edu

## Abstract

Text messaging has become a dominant form of communication, and emojis play a crucial role in conveying emotions for many people, which makes accurate emoji prediction for messages a useful task to explore. This project aims to predict emojis for texts in supervised setting and generalize to new emojis in zero-shot setting. We formulate supervised emoji prediction as a multi-class classification problem and fine-tune pretrained BERT and GPT-2 for this task. Further, we adapt instruction tuning that is originally proposed in the FLAN paper to GPT-2 to generalize to zero-shot setting. We find that the fine-tuned models achieve decent results for the supervised task, but instruction tuning fails to help GPT-2 generate accurate predictions for the zero-shot task due to the small scale of the dataset and model compared to those used for FLAN.

## 1 Introduction

The COVID-19 pandemic has led to a dramatic shift in the way people communicate over the last two years. Most of the people now have to turn to virtual communications to help prevent the spread of COVID-19, and text messaging has become a major channel for communication. Many people rely on using emojis to compensate for the lack of emotions in plain texts. As of September 2021, there were in total 3,633 emojis in the Unicode Standard. However, typically people are only familiar with a very small subset of them. And new emojis keep springing out, which makes it challenging for people to keep track of them all and choose the most accurate one. Therefore, accurate prediction of emojis is a very useful task that helps to make online communication richer and more emotive, which is especially important during the pandemic.

In this project, we aim to predict emojis from texts in supervised setting and generalize to zero-shot setting as well. Formally, denote the set of emoji labels by $\mathcal{E}$ and the dataset by $\mathcal{D} = \{(\mathbf{t}_n, e_n), n = 1, \cdots, N\}$, where $\mathbf{t}_n = \{t_1, t_2, \cdots, t_k\}$ represents a text sequence with $k$ tokens and $e_n$ refers to a single emoji in the label set $\mathcal{E}$. Given a piece of text $\mathbf{t}$, the model should out an emoji $e \in \mathcal{E}$ that best captures the emotion and context expressed in $\mathbf{t}$.

We employ the "pre-training and fine-tuning" paradigm for the supervised task using transformer-based language models BERT [1] and GPT-2 [2] as the base model, and we adapt the instruction tuning approach [3] to tune pre-trained language models for zero-shot emoji prediction. We find that fine-tuned BERT and GPT-2 are able to achieve decent performance on the supervised task, whereas the zero-shot emoji prediction is a much more challenging task that requires further investigation. We also give detailed error analysis for the supervised predictions to identify chances for improvements.

## 2 Related Work

### 2.1 Emoji Prediction

There has been limited work on the emoji prediction task so far. Felbo et al. [4] proposed DeepMoji built with a variant of Long Short-Term Memory model that uses emoji occurrences as noisy labels for distant supervision to better detect sentiment, emotion, and sarcasm. Unlike their work, besides the "mood" emojis, we also include other types of emojis such as "objects" and "nature" in our label set because our goal is to help people select the right emoji to use rather than performing sentiment analysis. Ma et al. [5] subsequently used BERT as the base model to make predictions on a richer set of emojis. Our project extends this line of work by fine-tuning both BERT, an encoder-only model, and GPT-2, a decoder-only model, for the supervised task and exploring the emoji prediction task in the zero-shot setting.

### 2.2 Transformers

Transformers [6] have been shown to be effective at a wide range of natural language processing tasks and led to the development of pre-trained language models such as BERT [1] and GPT models [2, 7]. BERT is designed to pre-train deep language representations from unlabeled texts by making use of a multi-layer bidirectional transformer encoder. It can be fine-tuned with just one additional output layer to produce strong results on various tasks such as question answering and language inference. In contrast, the GPT models are decoder-only transformers trained to predict the next token given previous contexts.

The emergence of these pre-trained language models has led to the popularity of the "pre-train and fine-tune paradigm", a simple yet effective way to tackle many downstream tasks. However, the gap between the pre-training objective and downstream tasks can be significant, and new parameters require lots of task-specific examples to be trained. Consequently, prompt-based learning was proposed to address these limitations by putting the downstream task into the same format as the pre-training objective through a template. For example, GPT-3 [7], a large language model with 175 billion parameters, can learn to handle many downstream tasks from a few examples presented through prompts without any gradient updates of the underlying model. Nevertheless, prompting techniques require careful prompt engineering and does not work well on natural language inference (NLI) tasks.

Drawing inspirations from both the "pre-train and fine-tune" and "prompting" paradigms, Wei et al. [3] proposed to improve zero-shot learning abilities of language models through *instruction tuning*—fine-tuning language models on a collection of datasets described via instructions. By instruction-tuning a pre-trained model with 137 billion parameters on over 60 NLP datasets and evaluating on unseen tasks, the authors have demonstrated that their instruction-tuned model FLAN (Finetuned LAnguage Net) not only substantially outperforms its untuned counterpart in zero-shot setting, but also surpasses GPT-3 in terms of zero-shot performance on 20 out of 25 datasets that were used for evaluation.

## 3 Approach

### 3.1 Supervised Setting

Following the "pre-train and fine-tune" paradigm (shown in Figure 1(A)), we fine-tune pre-trained BERT [1] and GPT2 [2] for multi-class sequence classification. Since BERT is an encoder-only model, we can use the final hidden state of the [CLS] token that is designed to represent the information of the entire text sequence to perform classification. In contrast, GPT-2 is a decoder transformer in which the last token in the input sequence contains all the information needed for making predictions in a classification task, which requires us to pad to the left. On top of the base models, we stack a simple linear layer with fan-in the same as the model's hidden size and fan-out the number of classes present in the dataset as a simple classifier. By mapping the outputs to a probability distribution through the classifier, our models predict the label with the highest probability.

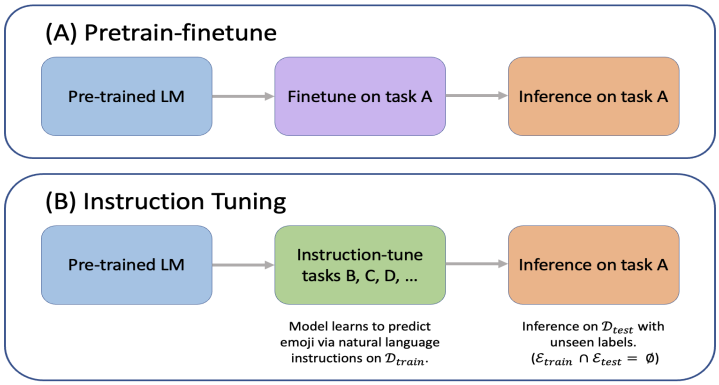


Figure 1: (A): method for supervised setting; (B): method for zero-shot setting.

### 3.2 Zero-shot Setting

For the zero-shot task, we adopt the instruction tuning approach [3] (shown in Figure 1(B)) and use the GPT-2 with a language modelling head as the base model. The intuition behind this method is that by teaching a language model to perform tasks described via instructions using supervision, the model could learn to follow instructions and even generalize to unseen tasks. While the original FLAN paper views zero-shot learning as models' ability to generalize to unseen tasks, we use a more traditional definition to define our problem as classifying instances among a set of unseen categories. Still, the instruction tuning method is applicable in our case if we consider the new emojis as the unseen tasks.

More explicitly, we hold out a subset of emojis for evaluation while using the rest for tuning. This setup ensures that our model has not seen some of the emojis during instruction tuning, so that we can test for its zero-shot ability on the emoji prediction task at inference time. To make our dataset suitable for the instruction tuning approach, we manually compose nine templates that use natural language instructions to describe the emoji prediction task and use them to transform our dataset into an instructional format. Again, since GPT-2 is a decoder-only model designed for free text generation, we need to adapt it to our classification task. Following the FLAN [3] paper, we include an *options* suffix to make the model aware of which choices are desired. An example of a pair of input and output is shown in Figure 2. To help the model to generalize, we randomly choose a template for each input text. In addition, we also randomize the number of options given and the actual options themselves included in the prompt. Meanwhile, we make sure that the true target is among the options given to the model.

We utilize two methods for our models to generate predictions for emojis. The first way exploits the ability of GPT-2 to generate text by giving the model a prompt in the instructional format and asking it to generate the next tokens until an "end-of-text" token appears. We take the generated sequence of tokens as the prediction. However, since the vocabulary size is over 50k and we place no constraints on the generation process, it is not guaranteed that the generated sequences are the desired emojis. Hence, the second way forces the model to generate one of the desired emoji labels by assigning a score to each of the possible labels and select the one with the highest score. Since emoji labels can be tokenized into multiple tokens of different lengths, we use chain rule to calculate the probability of each generated sequence and normalize the probabilities to take into account the effect of varying lengths on the resulting score based on the formula below, where $s, t, \mathbf{e} = \{e_1, e_2, \ldots, e_n\}$ denotes the score, input text, and tokenized emoji label respectively. Without the normalization factor, longer sequences would have lower scores than shorter sequences because probabilities between 0 and 1 are multiplied for a larger number of times.

$$s(\mathbf{e}|t) = [p(e_1|t) \cdot p(e_2|t, e_1) \cdot p(e_k|t, e_1, \ldots, e_{k-1})]^{\frac{1}{n}} \quad (1)$$

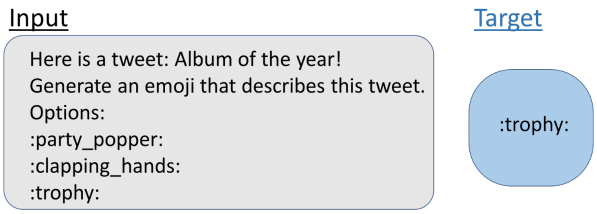


Figure 2: An example of input and corresponding target.

# 4 Experiments

## 4.1 Data

We use a dataset of English tweets with emojis extracted from the *EmojifyData-En* Dataset [8] downloaded from Kaggle. The original dataset is in BIO format and contains over 6 million tweets, each with at least one emoji from 49 classes in total. Since each tweet in the *EmojifyData-En* Dataset can contain more than one emoji, we filtered out those tweets with multiple emojis and obtain the *EmojifyData-En-Single* dataset with around 5.2 million tweets. In other words, each tweet in *EmojifyData-En-Single* contains only one single emoji from the same 49 classes as the original dataset. Hashtags, urls, and mentions were removed from the tweets, and we chose not to convert all letters to lowercase since capitalization can encode users' emotions. Note that the distribution over emojis is not balanced in that the most frequent one (😂) has 1 million occurrences while the least frequent one (🤦) only has 1579 occurrences.

### 4.1.1 Supervised

We randomly sample 1 million tweets from *EmojifyData-En-Single* to form *Emoji-1m-49*. In order to examine the effect of sample size on the fine-tuning results, we also form another dataset *Emoji-100k-49* in the same way except that it has a smaller size of 100 thousands tweets. Further, we obtain *Emoji-1m-20* and *Emoji-100k-20* from the two datasets respectively by picking those tweets with labels being one of the top 20 most frequently used emojis out of the 49 classes, each containing 749,570 and 75,087 tweets. We divide each dataset into train/dev/test sets with 80%/10%10% using random seed 2022.

### 4.1.2 Zero-shot

We split the *Emoji-1m-49* dataset based on labels to ensure that the labels predicted at test time were not seen in training. We randomly select 39 emojis for training and use the rest 10 emojis for testing, and we call the resulting dataset *Emoji-1m-zs*. To account for the imbalanced distribution, we also form another zero-shot dataset *Emoji-balanced* with a more even distribution by randomly sampling 3000 tweets from all emoji classes except for two, 🤷 and 🤦, which has only 2570 and 1579 occurrences respectively and thus all the associated tweets are included.

## 4.2 Evaluation method

We use top 1 accuracy, top 3 accuracy, and F1-score to evaluate the performance of our models, where top n accuracy is defined as the accuracy where the true class matches with any one of the n most probable classes predicted by our models. Top 3 accuracy is computed because there could be more than one reasonable emojis that can be used for each tweet. F1-score is also used here because there are imbalanced classes.

### 4.3 Experimental details

#### 4.3.1 Supervised

We use the pre-trained 'bert-base-cased' and 'gpt2' models provided in the HuggingFace transformers library [9] as the base models for fine-tuning. For each model, we stack a single linear layer with fan-in 768 and fan-out being the number of classes in the respective dataset as the classification head and fine-tune all the layers. We train with cross entropy loss for 5 epochs. The models predict the label with the highest probability score for each input sequence. We set the maximum sequence length to be 128 and batch size to be 64. We use the AdamW optimizer with learning rate 1e-4 and stopping threshold 10e-8.

#### 4.3.2 Zero-shot

We use the pre-trained 'gpt2' model with a language modelling head [9] as the base model. For each input tweet, a template is randomly selected out of the nine to turn it into instructional format; an integer $n$ is randomly sampled between 5 and 10 as the number of options to give; $n$ emojis are randomly selected as the list of options given in the prompt. If the true label is not among the randomly selected options, we randomly replace one of them by the true label. We prepend "<|startoftext|>" and append "<|endoftext|>" tokens to the formatted input to enable the model be aware of where a piece of text starts and end. During training, we use teacher forcing by including the true label in the formatted input; at test time, we do not give the true label and let the model generate until it reaches an "<|endoftext|>" token. We train for 3 epochs with batch size 8 and max sequence length 512. We use the AdamW optimizer with learning rate of 5e-5, gradient accumulation steps of 4, warm up steps 500, and weight decay 0.01.

### 4.4 Results

#### 4.4.1 Supervised

The training curves for GPT-2 and BERT are displayed in Figure 3 and 4 respectively. As demonstrated in the accuracy and loss curves, both of the models start to overfit after the fourth epoch. In addition, the training accuracy is initially lower than the validation accuracy, possibly because of the random dropout that brings a better generalization ability.

The results for models' performance on the supervised emoji prediction task are shown in Table 1. The top 3 accuracies are consistently higher than the top 1 accuracy since they are less strict. The performance of the pre-trained GPT-2 model without fine-tuning is close to random, suggesting that emoji prediction is a rather difficult task, and as expected, fine-tuning greatly boosts the performance of our models. Comparing the results across datasets with the same number of classes but different sizes, we see that increasing the size of training data helps with learning. Moreover, with fewer classes, our models are able to achieve a higher accuracy. Comparing the performance of fine-tuned BERT and GPT-2, we find that their performance are quite similar, though GPT-2 are slightly better in terms of top 1 accuracy and F-1 score on all four datasets.

#### 4.4.2 Zero-shot

Unfortunately, the instruction-tuned GPT-2 is not able to produce predictions better than random. On both zero-shot datasets, *Emoji-1m-zs* and *Emoji-balanced*, the training loss quickly drops to be very small at the beginning of the training process, whereas the validation loss gradually starts to increase (example training curves shown in Appendix A). For example, using the *Emoji-balanced* dataset, the training loss drops from 6.1 at step 100 to 0.39 at step 200, and keeps decreasing after that. To save training time, we evaluate on the validation set every 1000 steps. The validation loss at step 1000 is 0.66 and slowly increase to 0.75 at step 10,000. Even if we train the model for a longer period of time, the validation loss keeps increasing. We originally plan to train for 5 epochs, but stop after 3 epochs as the continuously decreasing training loss and increasing validation loss indicate that the model is clearly overfitting. When loading the best model for evaluating the model's performance

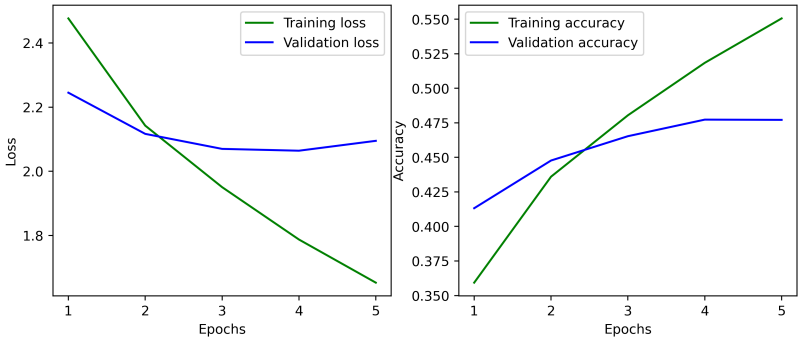


Figure 3: The training and validation loss (left) and accuracy (right) for fine-tuned GPT-2 on emoji-1m-49.

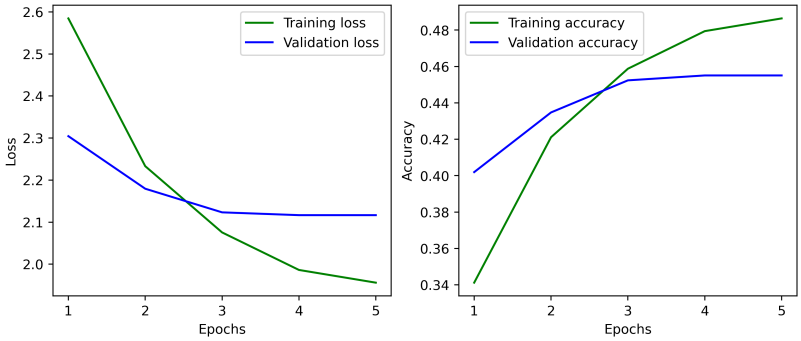


Figure 4: The training and validation loss (left) and accuracy (right) for fine-tuned Bert on emoji-1m-49.

| Dataset | Model | ACC | ACC@3 | F-1 |
|---|---|---|---|---|
| Emoji-100k-49 | GPT-2* | 0.01 | 0.06 | <0.01 |
| | BERT | 0.32 | 0.52 | 0.25 |
| | GPT-2 | **0.36** | **0.55** | **0.32** |
| Emoji-100k-20 | GPT-2* | 0.07 | 0.12 | 0.01 |
| | BERT | 0.42 | **0.67** | 0.37 |
| | GPT-2 | **0.43** | 0.67 | **0.41** |
| Emoji-1m-49 | GPT-2* | 0.01 | 0.03 | <0.01 |
| | BERT | 0.45 | 0.63 | 0.42 |
| | GPT-2 | **0.47** | **0.64** | **0.45** |
| Emoji-1m-20 | GPT-2* | 0.10 | 0.16 | 0.02 |
| | BERT | 0.55 | **0.76** | 0.53 |
| | GPT-2 | **0.56** | 0.75 | **0.55** |

Table 1: Results on supervised emoji prediction for pre-trained GPT-2 (denoted by *), fine-tuned BERT and GPT-2. We report top 1 accuracy (ACC), top 3 accuracy (ACC@3), and F-1 score for all models.

on the language modelling task, we discover that the first checkpoint saved at step 100 has the best accuracy, which is consistent with our speculation.

As it turns out, the evaluation accuracies are extremely low using either of the two ways for prediction. On one hand, when we provide the tuned model with a prompt and let it generate the next tokens, we find that the model does learn to generate an emoji, but it often ignores the list of options given in the prompt and produces labels seen in the training data. When it does look at the option list, the model tends to output three specific emojis most of the time, including ":back_hand_index_pointing_right:", ":smiling_face_with_sunglasses:" and ":white_heavy_check_mark:". On the other hand, the model does generate a more diverse set of emojis all belonging to the test label set when we force the model to predict a label from the test classes using scores from equation 1, but it still gives out "back_hand_index_pointing_right" and "smiling_face_with_sunglasses" most of the time, consistent with the free text generation case. As a result, the accuracies are below 10% for test sets, which only contain tweets with emojis from 10 classes.

## 5 Analysis

### 5.1 Supervised

Overall, fine-tuned GPT-2 achieves slightly better performance than BERT, possibly because BERT is pre-trained on the English Wikipedia and Brown Corpus while GPT-2 is pre-trained on a more diverse dataset, which might be more helpful for the emoji prediction task. A sample of the predictions produced by the fine-tuned GPT-2 is shown in Table 2. We can see that for both of the first two tweets presented in the table, the top 3 predictions contain the true label. In particular, the top 1 prediction exactly matches the true emojis that are used. Note that the model predicts 💀 for the first tweet, which is actually not a typical emoji to use, suggesting that our model is able to capture the meaning carried in less frequently used emojis. As for the third example, the true emoji is 💜, which is not among the top 3 predictions produced. Admittedly, our model has some stereotypes. Nevertheless, the top prediction ❤️ is closely related to the true label, and it is even hard for human to identify the true emoji in this case without additional contexts.

By inspecting at the evaluation report for all the classes, we observe a correlation between F-1 scores and the number of times the emojis appear in the dataset. Figure 5 displays two scatter plots for F1-score versus log(number of occurrences). We apply a log transformation to the number of occurrences to make the distribution of number of occurrences less skewed, and the plot shows a moderately strong positive relationship between the F1-score and the frequency of emojis as higher frequency is associated with higher scores.

| Tweet | Predictions (↓probability) | True emoji |
|---|---|---|
| Lmao my brother is so dramatic. | 💀 😂 😭 | 💀 |
| Done. Good luck everyone! | 😊 💕 ❤️ | 😊 |
| Its Yoongis birthday. We wish him all the best. | ❤️ 🎉 💕 | 💜 |

Table 2: A sample of predictions made by the fine-tuned GPT-2 model.

### 5.2 Zero-shot

Since the predictions made by the instruction-tuned model through free text generation are emojis present in the dataset, the model indeed learns from the data to some extent. However, the fact that it outputs emojis seen in training data suggests that the model starts to memorize the training labels and cannot generalize to new data. Moreover, as mentioned in section 4.4.2, the model tends to generate a few fixed emoji labels including ":back_hand_index_pointing_right:" and ":smiling_face_with_sunglasses:". We initially think the reason might be the imbalanced distribution of emojis in *Emoji-1m-zs*, but these

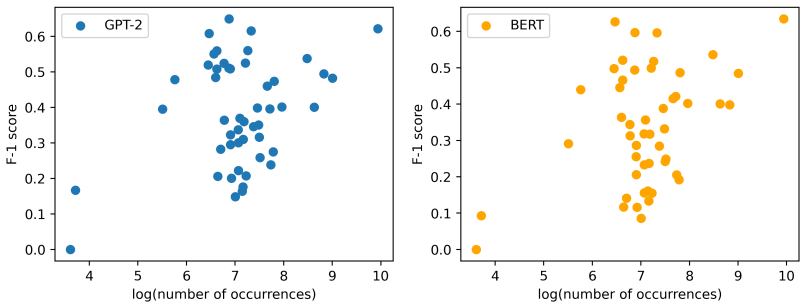


Figure 5: Scatter plot on F-1 score versus log(number of occurrences).

two emojis are respectively the 19th and 28th most frequent emoji in the dataset, which makes this explanation not plausible. Still, we construct the *Emoji-balanced* to see if a difference could be made, but the pattern persists, which rules out this explanation. We further notice that these emojis have longer lengths than other emojis. The model might deem generating more tokens as a shortcut that can lead to a higher chance of being correct. Even though the model predictions are mostly inaccurate, it does not mix up different emoji labels or make up new labels. The reason could be that the total number of emoji labels is small for a model like GPT-2, so it might be able to memorize the token combinations given in the prompt even at test time.

Besides the two zero-shot datasets mentioned in section 4.1, we also tried other ways of spliting the label sets for zero-shot setting, and all of them led to the same patterns of training and validation losses indicating that the model severly overfits the data. It is worth noting that the FLAN [3] model is instruction-tuned on over 60 datasets and its base model has 137 billion parameters. In contrast, we use a single dataset for both training and testing, and our base model has only 1.5 billion parameters. It has been demonstrated that the effect of instruction tuning on the performance of unseen tasks scales with model size. Hence, the smaller dataset and model used in our case might explain why instruction tuning does not help the model to generalize to unseen emojis.

## 6 Conclusion

In this project, we explored the emoji prediction task in both supervised and zero-shot settings. To achieve our goals, we fine-tuned two transformer-based models, BERT and GPT-2, with supervision and experimented with the novel instruction-tuning approach to perform prediction on unseen emojis. We are able to achieve F-1 scores of 45% and 55% on *Emoji-1m-49* and *Emoji-1m-20* datasets respectively, which is a substantial improvement from the pre-trained GPT-2 without fine-tuning, again demonstrating the efficacy of the "pre-train and fine-tune" paradigm in supervised setting. Though the instruction-tuned GPT-2 was not able to produce accurate predictions for unseen emojis, we observe some interesting phenomena through the predictions it make and show that a single task and a model of similar size as GPT-2 might not be sufficient for instruction tuning to work.

Due to the limited time and resources we have, we are not able to gather more datasets to further experiment with this approach. Also, it is difficult for us to gain access to models at scale like GPT-3 to match the size of the base model for FLAN. Nevertheless, zero-shot emoji prediction task is an important task that is worth further exploration. One possible future direction is to use images of emojis along with texts to develop multi-modal embeddings that might help achieve better zero-shot emoji predictions.

## References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*

*arXiv:1810.04805*, 2018.

[2] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[3] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. *arXiv:2109.01652*, 2021.

[4] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[5] Weicheng Ma, Ruibo Liu, Lili Wang, and Soroush Vosoughi. Emoji prediction: Extensions and benchmarking. *arXiv preprint arXiv:2007.07389*, 2020.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv:2005.14165*, 2020.

[8] https://www.kaggle.com/rexhaif/emojifydata-en.

[9] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

## A Zero-shot Training Curve

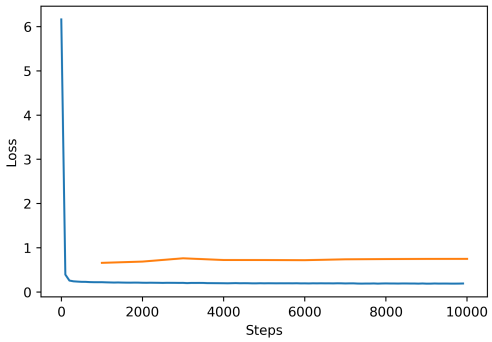


Figure 6: The training and validation loss for zero-shot on *Emoji-balanced* dataset.