

# An Emoji 😊 is Worth 128 Words

Anna Goncharenko  
University of Washington  
angonch@cs.washington.edu

Armin Magness  
University of Washington  
ajm1@cs.washington.edu

## Abstract

*‘A picture is worth a thousand words’ is an incredibly fitting statement when considering the use of emojis for communication today. Emojis clarify nuanced tone and increase the expressiveness of text. A text with sorrowful sentiment can be interpreted as ironic or even humorous with the addition of a single laughing emoji. The use of emojis is so abundant in social media and text messaging that companies have increased their convenience. Apple suggests emojis to replace key words while typing and provides easy access to a user’s most common and recently used ones. Instagram shows a personalized shortcut bar when writing comments, where users can quickly find their 8 most used emojis. Increasing the accessibility to emojis allows users to easily increase the expressiveness of their text. However, recommending the most common emojis does not prove to be helpful for all situations. Different sets of emojis may be useful for different contexts. Therefore, users would benefit from not only increased accessibility, but relevancy of recommendations as well.*

*In this paper we present Text2Emoji - a semantic emoji classification model. Given a sequence no longer than 128 characters, Text2Emoji classifies the English text input with one of the 20 most popular emojis and provides rankings for the top relevant emojis.*

## 1. Introduction

### 1.1. Background & Motivation

Emojis are crucial for communication over text and social media today. They are concise and universally understood, making communication fast and accessible beyond language barriers. Emojis allow the user to express nuanced tone and clarify their emotions. A text message such as ‘This is so sad’ is interpreted as having sorrowful sentiment, but ‘This is so sad 😊’ is ironic and even humorous. A single emoji can add important non-verbal cues that were previously only possible to

detect with face-to-face conversation [7]. In modern internet culture, emojis open up a world of possibilities in terms of expressiveness.

Today’s generation uses combinations of emojis to express an underlying feeling or ‘vibe’ that just can’t be conveyed easily with a collection of words. Like slang, an emoji’s meaning can change over time within internet culture. For example, the sparkles ✨ emoji used to mean ‘sparkles’ or ‘stars’ in the literal sense, but now is used to either convey positive feeling such as admiration and excitement, or is even used sarcastically to lighten negative statements. This is easily seen when looking at beauty advertisements on TikTok, where text is surrounded in sparkle emojis to convey positive emotions that make the user want to buy the product [2].

As emojis are a key component of communication today, it is natural that software companies increase access to them. Providing users with easier access to emojis invites them to augment their text with the symbols. Furthermore, since users are likely to use a similar set of emojis, companies may also increase access to a user’s most frequently used emojis, such as Instagram’s shortcut emoji bar when commenting on a post. While these personalized shortcuts provide a convenient way to use emojis, the recommended emojis are often times not relevant to all contexts. Figure 1 shows that regardless of the post, the recommended emojis will be the user’s most popular emojis, which may be grossly unsuitable for the context of the post. Thus, the shortcuts are ineffective when the recommendations are not appropriate for the context. It is important that the recommended emojis are also relevant to the situation.

In this paper, we provide a solution to recommending relevant emojis. Given text, a deep-learning BERT [3] model infers the most relevant emojis for the text out of the 20 most popular emojis.

### 1.2. Related Work

In 2018, the Semantic Evaluation competition (SemEval) had a MultiLingual Emoji Prediction task for



Figure 1. Example of an Instagram post from the New York Times where the recommended popular emojis are not relevant to the content of the post. The emojis are positive and celebratory, while the post is serious and about war.

emoji prediction in English and Spanish [5]. At the time of the competition, the model that achieved the highest precision for the English text emoji prediction was a support vector machine inspired model [10]. The model that achieved the highest recall was a Bi-LSTM based approach [4]. The dataset for the competition had 500,000 training examples, and 50,000 validation and test examples.

It is interesting to note that for the competition, the SVM approach outperformed the Bi-LSTM approach. It is generally accepted that attention is the state-of-the-art model architecture nowadays, especially for natural language processing tasks. In 2020, Snap developed TweetEval - a model that outperformed the models from the 2018 competition [6]. The model was a transformer based model trained for 9 days with the entire Twitter dataset scraped for 58 million training examples. This model shows that the attention based model can outperform the SVM approach, but at the time of the competition the 500,000 training examples were a severe limitation to the success of the approach.

In our work, we aim to fine-tune a large language model on a small dataset of less than 100,000 examples to produce a system that can effectively predict emojis for text. Inspired by TweetEval, we will create a baseline using BERT architecture [3] and a dataset from the Twitter corpus.

### 1.3. Plan

Our plan is to first acquire a dataset of text with a corresponding emoji. This can be done manually, which takes a significant amount of work. Another approach is to use the Twitter or Reddit API to scrape the social media platforms for this data and then fil-

tering for texts with only emojis. Then we plan to pre-process this data accordingly to be able to use it with large language models we are trying to fine tune. This may involve creating a Dataframe out of it, tokenizing or doing word embedding, and overall cleaning up the data as social media posts often times are much noisier sequences (with lots of special tokens) than traditional text.

We will use a pretrained language model such as BERT to fine-tune with the dataset. With BERT, this will involve adding another fully connected layer to generate scores for the 20 emojis and calibrate the model.

Lastly, we will do hyperparameter tuning to try to increase the accuracy of the baseline model created in the previous step.

### 1.4. Expected Challenges

We expect data collection to come with some challenges. While the Twitter API and Reddit API are a great tool to generate datasets, there are rate limit restrictions preventing the collection of large datasets without paying. We expect there to be issues with gathering a sufficiently sized dataset as well as preprocessing it so that it is compatible with the data structures the model requires. We also expect there to be some challenges integrating the fine-tuning pipeline. It is very common to do transfer learning for tasks - using a pretrained large model and fine-tuning it with a few more examples for a specific task. This involves importing and setting up the right classes, word embedders, optimizers, etc.... While the Hugging Face and PyTorch documentation are good, we still expect roadblocks along the way. Finally, we expect there to be significant challenges with computation. Fine-tuning a large language model on a large dataset may require longer compute time and even the use of GPUs.

### 1.5. Expected Outcome

There are two major axes of evaluation for Text2Emoji.

Firstly, we expect to see high accuracy on the test set. The state-of-the-art model, TweetEval, had an F1-score of 31.6 [6]. This was achieved with a combination of pulling a much larger set of Twitter data 58M tweets and training for 9 days with High GPU. Thus, we expect a significantly lower performance on the test data when using less than 100,000 training examples and training on \$50 worth of Google Cloud Credit. We will also try creating our own test set as the Twitter data is quite noisy, to test whether the expected emojis are correct for more natural text.

The second axes of evaluation will be that the model

is able to infer reasonable emojis given new text. For instance, given that one of the classes of emojis is the emoji for a christmas tree 🎄, we would expect inputs of ‘merry christmas’ or ‘christmas tree’ to yield us this emoji. We will create an easy-to-use pipeline to predict emojis for new sequences.

## 2. Methods

### 2.1. Data

The dataset for this project is sourced from the pre-crawled Twitter dataset of the SemEval2018 Competition [1]. Because utilizing the Twitter API to extract more data is expensive, our corpus comprises of the combined test and validation set from their paper to the data for a total of 100k examples. This is split into a finer grained training, validation, test set. After preprocessing, there was 34,000 training examples and 9,621 validation and test examples. As the Twitter data contains noisy text inputs and only represents a fraction of styles for how humans combine text with emojis, we constructed a custom test data with a few unique texts per emoji.

### 2.2. Preprocessing

#### 2.2.1 Data Cleaning

In the Twitter corpus, almost every string contained special characters such as ‘@’ and ‘#’ that are used for tagging users/locations and hashtags respectively. The BertTokenizer [9] will give each of these characters their own token and these tokens add noise to the training data and don’t add to the emoji semantic meaning [8]. Specifically, because the data was anonymized, any tagging of specific users in a tweet would appear as ‘@user’ in the string. This does not add any value to the emoji sentiment of a string and actively makes the model try and learn what ‘@user’ means when it appears in a string. In addition, when the ‘@’ was referencing a location, the location was removed to prevent the model from learning bias of what emoji’s are used by location instead of by the semantic English. Tweets that were less than 5 characters long were removed to avoid training on any empty or spam tweets [6].

Before	glam on @user for #kcon @ Los Angeles...
After	glam on yesterday for kcon

Figure 2. Example of data cleaning removing anonymous user tag, special tokens, and location.

#### 2.2.2 Data Class Split

The Twitter dataset has a heavy bias of examples that were labeled with the ❤️ or 😊 emoji. When training our initial models, the models were just learning the probabilities of those specific emojis in the dataset, always predicting those emojis over other ones with more semantic relevance to the inference input. To force the model to learn more of the semantic meanings of the words, we clamped the number of training examples to the count of the least represented class in the dataset. This came out to approximately 1700 inputs per class. The final cleaned and balanced dataset consisted of 34000 sequences and emoji labels split amongst the training, validation and test set(80%, 10%, 10%).

In addition to the Twitter data, we created a custom test set with a few unique sentences per emoji. Tweets can be quite noisy, often times containing misspellings, grammar issues, and in general unclear topics. The custom dataset contains clear sentences with unambiguous content that the model should be able to easily generate appropriate emojis for.

In experiments, models were trained on both the balanced dataset of 34k examples and the unbalanced dataset of 76k examples. The models were evaluated on the Twitter validation and test set, as well as on the custom test examples.

### 2.3. BERT

#### 2.3.1 Model Architecture Iteration

The inputs first need to be tokenized and packed into an input that BERT is able to work with. We used the provided BERTTokenizer and Packer [9], setting the maximum sequence length to 128.

**v1:** For our initial model, we attempted to take the readily available BertClassifier, load it with pre-trained weights and train against our uncleaned and unequally distributed dataset. We quickly realized that keras support of this model didn’t allow for it to be easily saved and loaded. This would be a large issue since our training time is not trivial or cheap.

**v2:** We switched over to using the BertModel nn.Module. By adding a Dropout and Classification Linear Layer, the model was modified to generate scores for the 20 emoji classes. The BERTModel was initialized with the provided pre-trained weights. The model was struggling to learn anything about the words and rather was just learning the probability distribution of the emoji classes within the unbalanced dataset. This initialized our attempts to balance the data. After an additional training attempt, we still weren’t seeing any validation performance that was better than random guessing (5%). We tried cleaning the data with

the intention of simplifying the sequences of tokens that we fed into the model. This again did not produce any promising validation results.

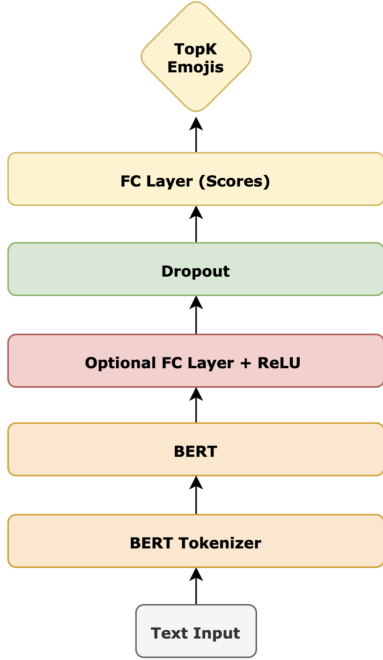


Figure 3. Architecture of finetuned BERT model for emoji classification. The input text is tokenized and encoded with the BERTTokenizer and BERT model. An optional FC layer and non-linearity are included, and results go through a Dropout layer. The final FC layer generates scores for each emoji.

**v3:** Because the model wasn’t learning anything about the dataset, we removed the pre-trained weights of the BertModel, leaving just the BertModel architecture with randomly initialized weights. This is the model that we report results for as our baseline. The architecture for this baseline model is shown in Figure 3. The architecture also includes an optional extra fully-connected layer and non-linearity before the Dropout layer, which was used in experiments.

## 3. Experiments

### 3.1. Model Architecture

During experimentation, it was revealed that adding an extra Fully Connected (FC) Layer and ReLU activation could have positive impact on the model performance.

The extra layers were able to correct overfitting and increase performance on the full validation and test set, however it performs worse on the custom set (Figure 4). Given the custom test set only contains 60 examples,

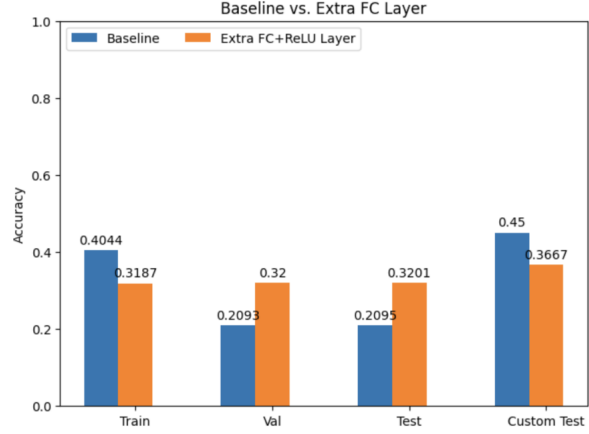


Figure 4. Training, validation, Twitter test and custom test set accuracies of baseline model vs. model with extra FC Layer + ReLU activation.

each individual correct/incorrect classification can significantly skew the resulting accuracies. Thus, we can conclude that adding an extra layer and linearity is better for model performance based on the accuracy for the full test set.

### 3.2. Hyperparameter Tuning

*Default:* Learning Rate =  $2e-5$ , Batch Size = 128, Epochs = 5, Optimizer = AdamW.

*Weight Decay:* HuggingFace does not provide an SGD implementation so we experimented with adding weight decay to AdamW (defaults weight decay to 0). Added weight decay of 0.01 to two versions of the model: v10, v11. Both resulted in significant drops in performance, seeing specifically in model v11 that we over-regularize as our accuracies crash (Table 1).

*Learning Rate  $\lambda$ :* Due to compute limitations we were not able to complete a random search over the learning rate space. However, we did experiment with the LR in model v6, and noticed a significant drop in performance (Table 1).

*Dropout:* The dropout layer is a critical part of the model architecture and changes in the dropout had significant impact on the overall performance. Figure 5 shows that dropout can act as a strong regularizer, allowing small changes in dropout percentage to drastically alter the amount of overfitting.

### 3.3. Training Data

After developing a model that was reaching good performance on the balanced dataset, the baseline model was trained on the full unbalanced training set.

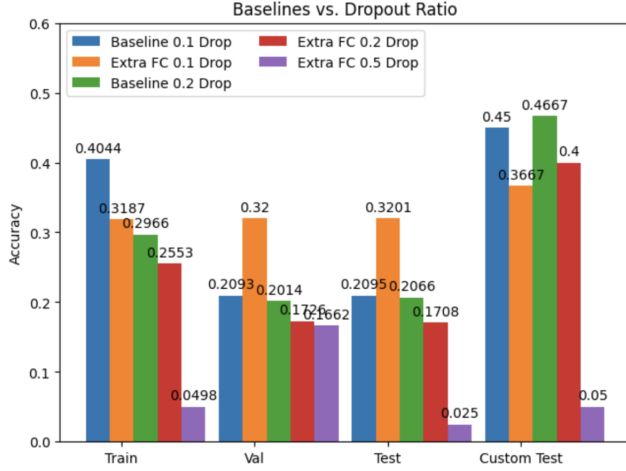


Figure 5. Varying model architectures trained with different degrees of dropout.

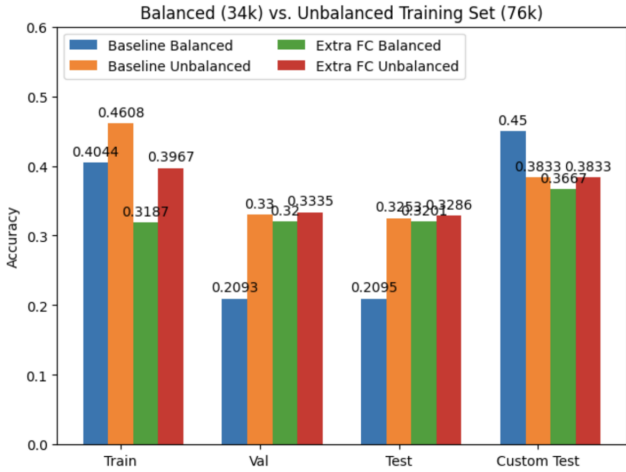
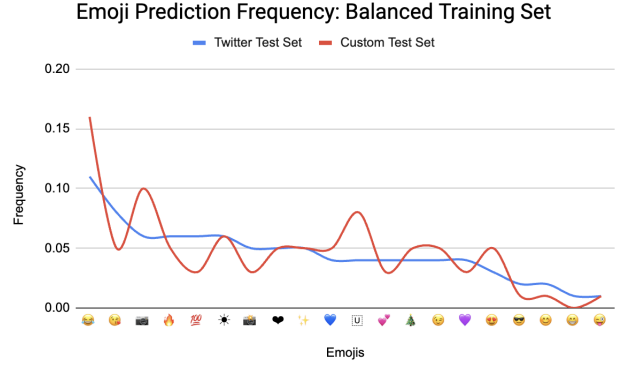


Figure 6. Varying model architectures trained on the balanced vs. unbalanced training set.

Model v12 contains the extra fully-connected layer and was trained on the full unbalanced dataset, achieving the highest test accuracy amongst prior experiments (Figure 6). However, it is important to note that the Twitter test set is built from the unbalanced dataset, so it was hypothesized that the model might be learning class distribution probabilities.

To explore this, Figure 7 and Figure 8 look at the class probability distribution of emojis predicted during inference on the Twitter test set and custom test set by the best model trained on balanced data (v9) and unbalanced data (v12).

Figure 8 shows that the class predictions for the model on the custom dataset follow closely with the probabilities of class predictions in the Twitter test set.





Model	Params	Training Data	Train Acc.	Val Acc.	Test Acc.	Custom Test Acc.
v2: Pretrained init	default	Balanced	0.05	0.0465	0.0378	0.05
v3 (Base): Random init	default	Balanced	0.4044	0.2093	0.2095	0.45
v4: Base	default	Unbalanced	0.4608	0.33	0.3253	0.3833
v5: Extra FC layer	default	Balanced	0.3187	0.32	0.3201	0.3667
v6: Extra FC layer	LR=2e-4	Balanced	0.05	0.0266	0.0314	0.05
v7: Extra FC layer	Dropout=0.5	Balanced	0.0498	0.1662	0.025	0.05
v8: Extra FC layer	Dropout=0.2	Balanced	0.2553	0.1726	0.1708	0.4
v9: Base	Dropout=0.2	Balanced	0.2966	0.2014	0.2066	<b>0.4667</b>
v10: Extra FC layer	Weight Decay=0.01	Balanced	0.2526	0.18	0.1855	0.3333
v11: Base	Weight Decay=0.01	Balanced	0.0573	0.057	0.0591	0.057
v12: Extra FC layer	default	Unbalanced	0.3967	0.3335	<b>0.3286</b>	0.3833

Table 1. Results for experiments testing two different model architectures, hyperparameters, the full unbalanced training dataset against the balanced training dataset. The results show that when looking at the accuracy for the Twitter test set of 9k examples, v12 performs the best, but when looking at the accuracy for the custom test set, v9 performs the best.

veloped key insights into the task of emoji prediction, such as the challenge of predicting emojis when subsets of them are interchangeable for the same input.

#### 4.1. Evaluation Results

The model with the best accuracy on the Twitter test and validation dataset was model v12, which had an extra fully connected layer and non-linearity and was trained on the full 76k unbalanced training set. This model was able to beat our baseline model (v3) accuracy by 13%. This model also achieved a macro-average F1 score of 21, which is lower than the state-of-the-art model’s [6] F1 score of 31.6. Given that we trained on 76k examples while the state-of-the-art model trained on millions of examples, this was expected. Thus, the final model passed our first axis of evaluation as it was able to beat the baseline’s model test and validation accuracy.

Custom Test Text	Top Emojis
merry christmas!!	🎄, ❤️, 😊
this volleyball team is so fire	🔥, 🏐, ❤️
i’m dead hahaha	😂, 🏐, 😊
Happy Fourth of July!	🇺🇸, ❤️, 🇺🇸
Selfie with my new camera!	📷, 📷, 😊

Figure 9. Sample texts tested on model v12, with top 3 predicted emojis in order of prediction score.

Figure 9 shows that the final model is able to recommend relevant emojis for various texts from our custom test set, which passes our second axis of evaluation.

#### 4.2. Insights

It is important to note that while the accuracy and F1 score are relatively low, even for the state-of-the-art model, this does not mean the models have poor performance. Through experimentation and analysis of inference results, it was learned that it is very hard to correctly classify a specific emoji. This is because many emojis are often times interchangeably used for the same context. For example, any positive smiling or laughing face emoji may be used for a text with happy or funny sentiment. Similarly, the various heart emojis are often times interchangeably used for the same text. The important takeaway is that even if the model didn’t recommend the same ‘ground truth’ emoji, it will recommend a similar *relevant* emoji.

After analyzing which emojis were correctly or incorrectly assigned in the custom test set, we found which emojis were ‘easy’ and ‘difficult’ to classify for particular texts. ‘Easy’ emojis were found by seeing which emojis had all texts correctly classified. Similarly, ‘difficult’ emojis were found by seeing which emojis had all texts misclassified as other emojis.

Results show that the easiest emojis to assign to text are 🎄, 😊, 🔥, 🌞, 🇺🇸. These emojis are associated with specific words like ‘Christmas’/‘holiday’, ‘funny’, ‘fire’/‘lit’, ‘sunny’/‘sky’, and ‘fourth of july’/‘fireworks’. When emojis are associated with specific scenarios and words, they are much easier to learn how to predict.

The hardest emojis to assign to text are shown in Figure 10, along with which emoji they were commonly misclassified as. The predicted emojis are incredibly similar to the ground truth and would often be used interchangeably in the associated context. For example, a camera with or without a flash could be associated

with the same text. Similarly, any positive facial expression, such as grinning or laughing or winking, are often times used interchangeably as well. Finally, there are numerous different heart emojis that would all be appropriate for the same context. These emojis are all variations of the same object or facial expression. It is important to note that even though the prediction does not exactly match the ground truth, it is still an incredibly similar emoji, such as the same object or a facial expression with similar sentiment.













Ground Truth	Prediction
	
	 , 
	 , 
	 ,  , 

Figure 10. Emojis that were most commonly misclassified and what they were misclassified with.

Analyzing ‘easy’ and ‘difficult’ examples revealed a key insight about how the model is good at learning associations between specific words and emojis, but has trouble when there may be multiple emojis appropriate for the same context. Emojis that are variations of the same object or facial expression could really be interchangeable in the same context, thus we don’t think it is incorrect that the model misclassified these examples. These are hard emojis to distinguish because they are all used in similar contexts. Instead, we conclude that the model achieves the desired task as it does well at predicting relevant emojis for different scenarios. For example, it does well predicting event-specific emojis (like Christmas and the Fourth of July) as well as appropriate emojis for a particular feeling (such as laughing emojis for funny scenarios and heart for more heartwarming scenarios, even if the actual emoji does not perfectly match the ground truth).

## 5. Conclusion

This project develops a model that is able to recommend relevant emojis for different text inputs. Through experimentation, we developed two competing models that beat our baseline model and achieve the highest accuracy for the Twitter test set and our custom test set amongst other experiments. The first best performing model is model v12, which includes an extra fully-connected layer and non-linearity, and was trained on a larger but unbalanced dataset containing an uneven distribution of examples per emoji. The other best performing model is model v9, which did not have extra layers, was trained on a smaller but balanced dataset

containing an even distribution of examples per emoji, and had a higher dropout than the baseline model.

While model v12 achieved the highest accuracy on the Twitter test set, its lower accuracy on the custom set and further analysis showed that the model was actually heavily biased by emoji probabilities. The model was often times predicting emojis based off of their frequency in the training set rather than by semantic associations. Similarly, while model v9 achieved less accuracy compared to v12, it achieved higher accuracy on the custom set and analysis showed that it was able to predict different emojis equally and was unbiased by their frequencies.

Since our goal was to develop a model that recommends emojis based on their relevancy to the task, rather than their popularity, model v9 is the best fit. Inference on various texts show that the model successfully predicts relevant emojis for very different contexts.

### 5.1. Limitations

One major limitation to this project was access to training data. To obtain the full Twitter dataset of over 5 million tweets, as the state-of-the-art model [6] does, we would need to pay for an expensive subscription. Since emojis are often times used in texting and social media, it would be impossible to get more data without paying for a subscription to scrape Twitter or Instagram comments. Furthermore, scraping the internet and other websites for examples with emojis would not only be a time-consuming task, but may not produce lots of examples, as large bodies of text on most websites don’t have emojis. Thus, we used the existing Twitter dataset that already had emoji labels, but were limited by the size.

Another limitation of this project is that the Twitter dataset only included examples for 20 emojis. If the goal is to be able to recommend various emojis, then one would need a large dataset with many examples for all of those emojis. This dataset creation would be an entire project in and of itself, likely consisting of scraping various social media platforms and filtering comments for comments with emojis.

The last major limitation was access to GPU memory based on the limited hardware we had access to. Because we were training on smaller GPUs we couldn’t construct as large of models. When we tried to increase the model size further by an additional FC Layer, we would receive an “out of memory” error. While state-of-the-art large language models train with hundreds of billions of parameters on multiple GPUs, while our model’s BERT component had 110M parameters [9] and we were training on a single GPU.

## References

- [1] Francesco Barbieri. Semeval2018 task2 - multimodal emoji prediction, 2018. [3](#)
- [2] Juwhan Choi. The effect of the smiley emoji and sparkles emoji in tiktok beauty product ads on female genz’s emotional valence and purchase intention, 2023. [1](#)
- [3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. [1](#), [2](#)
- [4] Christos Baziotis et al. Ntua-slp at semeval-2018 task 2: Predicting emojis using rnns with context-aware attention, 2018. [2](#)
- [5] Francesco Barbieri et al. Semeval 2018 task 2: Multilingual emoji prediction, 2018. [2](#)
- [6] Francesco Barbieri et al. Tweeteval: Unified benchmark and comparative evaluation for tweet classification, 2020. [2](#), [3](#), [6](#), [7](#)
- [7] Isabelle Boutet et al. Emojis influence emotional communication, social attributions, and information processing, 2021. [1](#)
- [8] Didier Henry, Erick Stattner, and Martine Collard. Filter hashtag context through an original data cleaning method, 2018. [3](#)
- [9] HuggingFace. Bert. [3](#), [7](#)
- [10] Dennis Asamoah Owusu and Jonathan Beaulieu. Umduluth-cs8761 at semeval-2018 task 2: Emojis: Too many choices?, 2018. [2](#)