Emoj-AI

Drake Svoboda
drakes@umich.edu

Evan Abrams
evanabra@umich.edu

Denny Yeung
denyeung@umich.edu

Abstract

We train a DistilBERT transformer for the task of emoji prediction. We model emoji prediction as an information retrieval problem at the level of individual tokens. This allows our model to predict context aware emojis at different locations in an input text. We train on a large corpus of tweets. We compare the performance of our model against a base-line logistic regression model. The transformer significantly outperforms the linear model and is able to predict a more diverse set of emojis.

1 Introduction

Emojis are textual characters that are commonly used in casual settings to represent different emotions, people, or things in a pictorial format. Currently, the Unicode Standard (Consortium, 2020), which is a widely adopted character encoding scheme, defines a set of 1,809 emojis. There are over 18,000 including modifiers such as skin tone and gender. It can be somewhat cumbersome for users to navigate all of these emojis, given only basic search functions. Emoj-AI aims to help text messaging users by predicting emojis that best suit the user's needs with the goal of predicting a diverse set of context aware emojis.

In this paper we train a DistilBERT transformer for emoji prediction and compare its performance to a logistic regression baseline. Recent works in emoji prediction focus either on predicting emojis at the level of a complete text or suggesting emojis using RNNs (similar to next-word prediction). We model the task of emoji prediction as an information retrieval problem at the level of individual tokens. This allows our DistilBERT model to recall a rich set of emojis that consider differing contextual information at each token. We imagine our model could be integrated into a custom Android keyboard that decreases the amount of time users spend searching for emojis.

2 Related Work

Ramaswamy et al. use LSTM to suggest emojis to users as they type (Ramaswamy et al., 2019). Their model prompts the user with single emoji suggestions when it determines if an emoji is appropriate.

Barbieri et al. use multimodal input data to predict one or more emojis for an instagram post (Barbieri et al., 2018). Their model takes as input an instagram post containing a picture and description and outputs one or more emoji predictions. This model predicts emojis at the level of complete posts rather than at the token level.

3 Dataset and Problem Definition

Our dataset is a corpus of 1.8 million English tweets from the EmojifyData-EN Twitter corpus available from Kaggle. Each tweet in the corpus contains at least one in-line emoji. We select subsets of 490,000 tweets and 10,000 tweets for training and testing respectively.

The Unicode consortium defines a complete set of 1,809 emojis. We select a subset of these emojis for prediction. We exclude all emojis with gender, skin-tone or hair-color modifiers. These emojis are transformed to their non-modified counterparts. For example, the emojis 👍🏻👍🏼👍🏽👍🏾 and 👍🏿 are each replaced with 👍. We also exclude country flags, letter, and number emojis; tweets containing these emojis are not included in the training or testing data. This leaves a subset of 1,344 emojis that we consider; call this subset E.

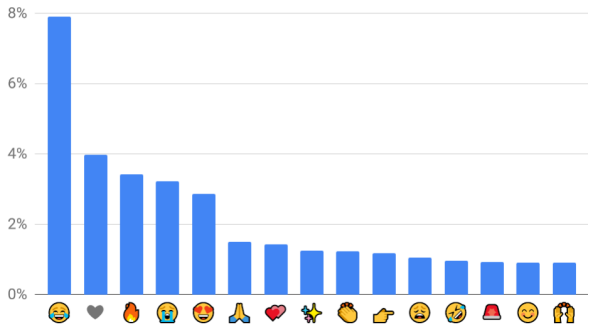

Figure 1: Distribution of the 15 most frequent emojis in the training data.

The distribution of emojis is highly imbalanced. The top 100 most frequent emojis in the training data make up almost 70% of all emoji usage. The most popular emoji (😂) is responsible for almost 8% of usage on its own. Several emojis do not have support in either the training set or the validation set. Figure 1 shows the distribution of the top 15 most used emojis.

Some of the tweets in the corpus are not indicative of normal casual text communication. Many tweets are either spam, advertisements, or obvious bot tweets. This skews the distribution of emojis and hinders our models ability to learn useful relationships between text and relevant emojis.

Problem Definition We model the task of emoji prediction as a separate information retrieval problem at each individual token. That is, we expect our models to retrieve a set relevant emojis for each token. The in-line emojis in the training data are used as a learning signal to determine which emojis are relevant. If a token is followed by a sequence of emojis (or single emoji), then those emojis are considered relevant for that token and are taken as ground truth. If a token is not followed by any emojis, we do not infer that there are no relevant emojis for that token. Instead, we consider that token as having some hidden set of relevant emojis that may or may not be empty. During training, we do not penalize the model for making predictions at such tokens. Since emojis are sparse in our input, this design decision ensures the model do not regress to predicting no emojis for each token. This encourages our model to over-predict emojis, which is beneficial for our particular use case. We imagine this model could be packaged in a mobile device's keyboard. If while the user is composing or editing a text message, they wish to type an emoji, they would switch to the emoji view where they would be presented with the set of predicted emojis for the token their text cursor is positioned at. If the user has switched to the emoji view, this is an indication that the set of relevant emojis for the position of their cursor is non-empty. That is, in the common use case, our model will not be asked to make predictions for tokens that do not have relevant emojis.

For a token i, our model predicts an $|E|$ dimensional prediction vector p_i where $p_i[j] \in [0, 1]$ is interpreted as the probability emoji j is relevant for token i. We consider the model as predicting emoji j if $p_i[j] > \epsilon$ for some threshold $\epsilon \in [0, 1]$. We can rank the models predictions by sorting $p_i[j]$ and giving the emoji with the largest $p_i[j]$ the lowest rank and so on. A perfect model would have $p_i[j] > \epsilon$ for all relevant emojis and $p_i[j] \leq \epsilon$ for all non-relevant emojis.

4 Methods

We train two different models for the task of emoji prediction: a bag-of-words (BoW) logistic regression model and a DistilBERT transformer model (Sanh et al., 2019).

4.1 DistilBERT Transformer Model

We use PyTorch (Paszke et al., 2019) and Hugging Face library to define and train our model. The Hugging Face library is an NLP library that interfaces with PyTorch and supplies several pre-trained model architectures with their accompanying tokenizers. Transformers were proposed for sequence-to-sequence tasks (Vaswani et al., 2017). Our model transforms a sequence of tokens to a sequence of emoji predictions. That is, our model outputs a set of emoji predictions for each token in the input.

Figure 2 shows our model's architecture. The transformer produces a hidden state a_i for each token t_i in the input. Each a_i is passed through a shared fully-connected classification head followed by the sigmoid activation function. This produces the $|E|$ dimensional prediction vector p_i for each token.

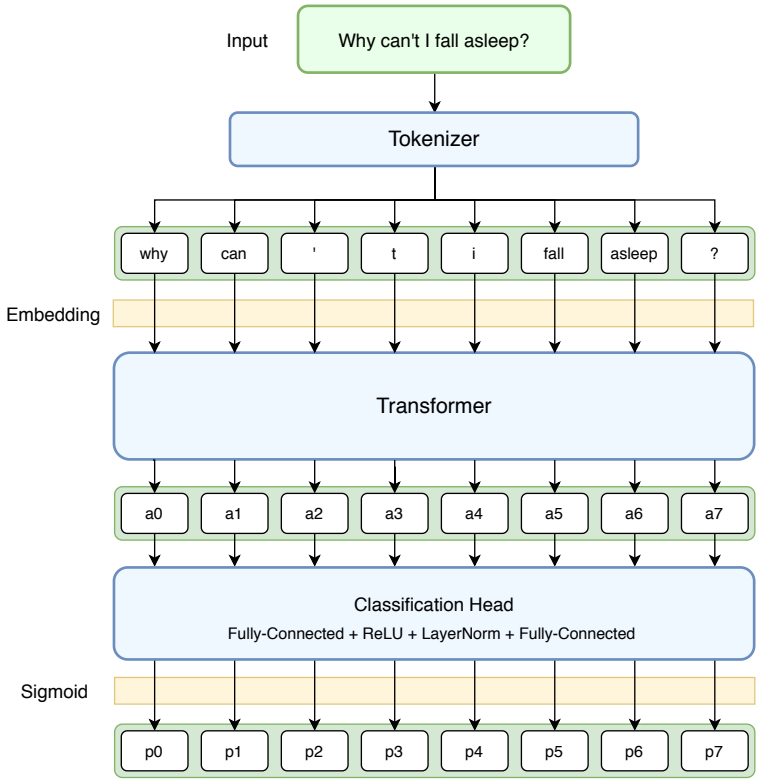

Figure 2: Transformer for Emoji Prediction.

Input	Predictions
Why can't I fall asleep?	😔😴😞😕💤
I love you	❤️💕💖💛💓
Who wants to buy me a pizza?	🍕😋😍🙈🤗
I left the stove on...	🧐🤣😰🤦😅

Table 1: Sample transformer predictions. Note that the model makes a set of predictions for each token in the input, however, we only present predictions for the final token. Predictions for entire sequences can be found in table 3.

We choose DistilBERT (Sanh et al., 2019) (a smaller variant of the popular BERT transformer (Devlin et al., 2018)) as the transformer in our model. Before training, DistilBERT is initialized with pre-trained weights. The Hugging Face library supplies a DistilBERT model pre-trained on a large corpus of English text for the task of masked language modeling (MLM). MLM is a self-supervised NLP task where the model is trained to predict tokens masked from the input i.e. fill in the blanks. The pre-trained MLM model does not have emojis in its vocabulary. We add each emoji in E to the vocabulary and randomly initiating each emoji's embedding. We then fine-tune this model for the MLM task on our corpus of tweets. We fine-tune for 20 epochs with a learning rate of $5e-6$ using the Adam optimizer (Kingma and Ba, 2014). We extract the embedding layer and transformer from the MLM model and attach a randomly initialized classification head to create the emoji prediction model in figure 2.

We optimize our model for emojis prediction using negative-log likelihood loss and stochastic gradient descent. Each tweet in the training data is parsed into input tokens and labels. Each input token that was followed by one or more emojis is assigned those emojis as labels. For example, consider the following tweet: "Oh no 😳 I left the stove on 😮🤦." This tweet is parsed into the following list of tokens: [Oh, no, i, left, the stove, on]. The token "no" is assigned an $|E|$ dimensional multi-hot vector as its label with a one in the position corresponding to the "😳" emoji and zeros elsewhere. The token "on" is assigned a different label with ones in the positions corresponding to "😮" and "🤦" and zeros elsewhere. The remaining tokens are not assigned labels and are not considered when computing the loss. In this scheme, the emojis in the original text are not passed as input into the model and instead are

removed and only used as labels. This prevents the model from learning relationships between emojis; that is, how an emoji in the input can influence the set of relevant emojis at each token. To correct this, each emoji in the original text has a small probability of being retained in the input and not selected as a label. We set this probability to 0.1 during training. This improves the performance of the model when emojis are in the input, however, we do not include emojis in the input when evaluating the transformer on the test set.

The model is trained on 490,000 tweets for 70 epochs using the with learning rates of $5e-6$ for the transformer and $5e-5$ for the classification head. Again we use the Adam optimizer (Kingma and Ba, 2014). Since the transformer is pre-trained, we expect it to produce predictive hidden representations even before optimizing for emoji prediction. For this reason, we choose a smaller learning rate for the transformer than for the randomly initialize classification head. Figure 1 shows a set of sample predictions. Section 5 gives an in depth evaluation of the model's performance.

4.2 Logistic Regression

The logistic regression model serves as a baseline to compare the performance of our transformer model. The model predicts an $|E|$ dimensional probability vector p from a bag of words representation of the input. We interpret p as the emoji predictions for the final token in the input. For example, if the the text "Oh no I left the stove on" was used as input, the output p is the set of predictions for the token "on". We can get predictions for the token "no" by only using the sub-string "Oh no" as input. Using this framework, the logistic regression model is able to make a prediction at each token and can be directly compared to the transformer.

To train the logistic regression model, we break each tweet in the training set into multiple input-label pairs. We do so by splitting the input at every occurrence of a sequence of emojis. The text preceding the emojis is taken as input and the set of emojis are taken as the label.

For example, consider the tweet from before: Oh no 😳 I left the stove on 😮😭

This tweet is split into two separate feature/label pairs. The first pair consists of the input "Oh no" and the label set [😳]. The second pair consists of the input "Oh no 😳 I left the stove on" and the label set [😮,😭]. The inputs are represented as bag-of-words embeddings that are constructed by passing each input text into a CountVectorizer from sklearn (Pedregosa et al., 2012) with a maximum vocabulary size of 5,000. A total of 150,000 training tweets are parsed into 205,508 feature-label pairs. As in the transformer, the labels are represented as a $|E|$ dimensional multi-hot vector. We use the one-vs-rest strategy for multi-label prediction. A separate binary logistic regression model is fitted for each emoji. To get the prediction vector p, for each emoji j, the corresponding logistic regression model outputs the probability $p[j]$ that j is relevant.

The model is trained on 205,508 training examples. Section 5 gives an in depth evaluation of the model's performance.

5 Results & Discussion

We compare the performance of our models using the following metrics: precision at $\epsilon = 0.5$, recall at $\epsilon = 0.5$, average precision, one-error, and ranking loss (Sorower, 2010). We only compute these metrics for positions in the text where ground truth emojis are present. That is, we only consider the models performance at tokens where there are known relevant emojis. Predictions for tokens with hidden relevant emojis are not considered when evaluating. To compute precision and recall, we count a true positive when $p_i[j] > \epsilon$ and emoji j is relevant for token i; a false positive when $p_i[j] > \epsilon$ and emoji j is not relevant for token i; and a true negative when $p_i[j] \leq \epsilon$ and emoji j is not relevant for token i. We compute precision and recall for each class and report both a micro and macro average. Recall is undefined for classes with 0 support and precision is undefined for classes where $p_i[j] \leq \epsilon$ for all i. We exclude such classes when computing the macro average. We also report the average precision (AP) for all values of ϵ. We do not report accuracy as it is overwhelmingly dominated by true negatives.

We also evaluate our models using common ranking metrics. At each token, we rank the

models predictions in decreasing order using prediction vector p_i. One-error (OE) computes the percentage of tokens where the top ranked emoji does not belong to the set of ground truth relevant emojis. Ranking loss (RL) computes the average number of emoji prediction pairs that are incorrectly ordered in the ranking (emojis present in the ground truth should be ranked before emojis not in the ground truth).

Table 2 compares the results of our models. The transformer model outperforms the logistic regression model in every metric besides precision. Both models achieve much higher precision than recall. This is caused by the significant class imbalance in the training data. The top 15 most frequent emojis make up nearly 35% of all emojis in the training set. The logistic regression model is able to achieve higher precision by only predicting a small-subset of emojis. Overall, the transformer is a much better performing model and is able to predict a much more diverse set of emojis. This is seen clearly in the transformer's superior recall. Even still, there is a set of infrequent emojis that the transformer never predicts. Of the 756 emojis with support in the validation set, the transformer only assigns positive predictions to 539 of them.

Table 3 shows predictions for a few example token sequences. Evaluating the relevance of the predicted emojis is subjective. It is likely that for many of the tokens the "true" set of relevant emojis is empty. However, for tokens where it is plausible that a person might write an emoji, the predictions often seem relevant. The transformer is particularly good at predicting emojis when a single noun is input on its own. For example, the highest probability prediction for "clown" is 🤡. Some more interesting predictions are 🐰 for "rabbit", 😍 for "love", 🚿 for "shower", 💔 for "agony" and 💡 for "enlighten". A larger list of single noun predictions can be found in section A.

Ethical Considerations Several emojis defined by the Unicode consortium are either gendered or have skin tone modifiers. We explicitly remove skin tone and gender modifiers to help prevent the model from learning racial or gender biases that may exist in the data. However, our model is still susceptible to learning more subtle biases. What makes machine learning models powerful is their ability to learn biases and relationships. Unfortunately, some of these biases may be contrary to a more sophisticated moral understanding. As with all language, emojis are not exempt from being used in hateful and non-egalitarian ways. With that said, it is indeed likely that our models have learned morally-incorrect emoji usages.

6 Conclusion

In this paper, we train a both a DistilBERT transformer and a logistic regression model for the task of emoji prediction. We model emoji prediction as information retrieval at the level of individual tokens. That is, we expect our model to recall the set of relevant emojis for each token in an input text. This formulation lends itself to sequence-to-sequence models. Given a sequence of input tokens, our transformer outputs a sequence of predicted relevant tokens. Despite a large class imbalance, our transformer model is able to recall a diverse set of emojis. We imagine our model could be integrated into a mobile device's keyboard to aid users by suggesting relevant emojis upon request.

Future Work It would be interesting to explore how pre-training DistilBERT affects the overall performance of the model. We take an additional pre-training step of adding emojis to DistilBERT's vocabulary and fine-tuning for the MLM task. In the future we would like to conduct an ablation study to determine what parts of our training formulation are most important to the performance of the model.

We would also like to train on a better corpus of text. Tweets are not indicative of normal text conversation. Additionally, our data set contained a large number of spam, advertisement, and bot tweets. We expect that the model would be more useful if trained on a more representative corpus.

Given more time, we would research and implement other models such as OpenAI's GPT-2. The implementation of the model would include the necessary training and testing, similar to the BERT model. Subsequently, we would compare the GPT-2 model to the Logistic Regression and BERT model based on evaluation metrics considered earlier.

We imagine our model could be packaged in a

Model	$\text{Precision}_{\epsilon=.5}$		$\text{Recall}_{\epsilon=.5}$		AP	RL	OE
	Micro	Macro	Micro	Macro			
Transformer	0.8184	0.8439	**0.4098**	**0.3397**	**0.5305**	**0.0375**	**0.5149**
Logistic Regression	**0.8940**	**0.9157**	0.0870	0.0208	0.4051	0.0528	0.7675

Table 2: Results in terms of precision, recall, average precision, ranking-loss, and one-error.

I	would	lock	hi
love	you	the	how
my	like	door	was
new	fries	when	your
dress	with	you	day
	that	leave	

Table 3: Sample transformer sequence predictions. The top 5 predicted emojis are presented for each token.

web-service and Android keyboard. If the user would like to type an emoji while composing a text, they would open an emoji view. Upon opening the view, the model would compute a set relevant emojis and a portion of the view would display the predicted relevant emojis for the token nearest the user's text-cursor.

References

Francesco Barbieri, Miguel Ballesteros, Francesco Ronzano, and Horacio Saggion. 2018. Multimodal emoji prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 679–686, New Orleans, Louisiana. Association for Computational Linguistics.

Unicode Consortium. 2020. *The Unicode Standard*, volume 13.0.0. The Unicode Consortium.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2012. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490.

Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. 2019. Federated learning for emoji prediction in a mobile keyboard. *CoRR*, abs/1906.04329.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Mohammad S Sorower. 2010. A literature survey on algorithms for multi-label learning. Technical report.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

A Supplemental Material

The code to train our models is publicly available on google colab [1].

You can demo our trained DistilBERT model in google colab[2]. Execute the entire notebook; it will download the trained weights and the final cell will output an interactive text-box.

[1] `Training notebook in google colab`
[2] `Demo notebook in google colab`

cat	😻
dog	🐶
lion	🦁
tiger	🐯
frog	😂
turtle	😂
tree	❕
flower	🌸
baseball	⚾
football	⚽
happy	😃
sad	💔
elated	😍
ecstatic	😍
devoid	⭐
angry	💢
hostile	🎧
agony	💔
love	😍
despair	😢
philosophy	👉
math	😗
immaculate	🙏
machine	😂
learning	💥

Table 4: DistilBERT single word predictions