Peaches, Snakes, and Double Meanings: Applying Embeddings to Emojipastas

Aidan Casey
Purdue University
casey39@purdue.edu

Simon Mason
Purdue University
mason142@purdue.edu

1 Introduction

Github Repository link: https://github.com/aarcc530/cs577-emoji-embeddings/
Presentation within repository as pptx and appended to this report as pdf.

1.1 The Problem and Key Challenges

Most NLP projects and papers that analyze social media remove emojis from the text. However, this can be problematic since emojis are beneficial to sentiment analysis, sarcasm detection, and emotion detection tasks. We would like to contribute to the ongoing research of converting emojis into vector representations similar to words. The main difficulty of this problem is the inconsistent use of emojis on social media websites. Emoji definitions are not static and it is possible that they can have change over time. For example, the raised fist emoji used to have a meaning of being excited or pumped up. However, recently it has mostly been used to convey political protests and social resistance. Another potential challenge is that the top 100 emojis are used much more frequently than any other emoji. This can make it very challenging to get accurate embeddings for most emojis.

1.2 Novel Aspects

We will be following a model similar to the work of Barry et al. (2021), where the authors created emoji embeddings on some of the newest emojis added in 2021. However, most emoji embeddings, like those trained in Eisner et al. (2016), are created using the official descriptions from Unicode standard (https://unicode.org/emoji/charts/full-emoji-list.html) or from "Emoji Dictionaries" such as https://emojipedia.org/ and https://emojis.wiki/. Their reasoning for using dictionaries instead of social media was the concern that emoji use was inconsistent for each post. Each emoji on emojis wiki and emojipedia has a short literal description as well as a short metaphorical definition of its use. However, this dictionary source might be biased to how the author perceives the emoji or might be slightly out of date for the current use of the emoji. We plan to address this issue by using a new data source from the subreddit r/emojipasta. The authors claim that emojis used online do not follow any grammatical rules and can simply appear at any point and any time. However, on the reddit page we found that there are strict rules on post effort and emoji placement. The emojis on each post and comment must be related to the words surrounding them, and there must be at least one emoji for every few words. This will create a consistent structure for each sentence needed to accurately create embeddings for each emoji. Our goal is that emoji embeddings generated on this dataset will create more accurate representations. We plan on capturing hidden and metaphorical definitions not represented in online dictionaries.

1.3 Main Research Questions

We ask three main research questions. First, we wish to see if our emoji embeddings clearly capture associations between words and emojis (ex. a smile face should be similar to happy words). This will give insight on if the embeddings exhibit any obvious connections between words that should be contextually similar. If succesfull, we believe our embeddings would boost prexisting models. We may also compare our results with models such as emoji2vec to see if similar connection are made.

Second, we will attempt to create emoji embeddings that capture the hidden meanings emojis often contain. For example, the snake emoji can mean the literal snake or describe someone as backstabbing. We plan on analyzing the cosine distance between popular emojis to see if the hidden meanings are learned. We will attempt to use the cosine distance to compare our models' behavior to previous research which heavily relies on literal descriptions.

Third, we wish to have an embedding objective with a downstream task. As such we will evaluate if the new embeddings improve performance in such a learning task (in particular, n-gram as is discussed later) over randomized embeddings or existing embeddings such as emoji2vec. This will give additional insight into the potential utility of these embeddings in tasks involving similar datasets, and perhaps internet posts in general. Additionally, this gives a more quantitative measurement of performance for our embeddings.

2 Methods

For the purpose of our work, we utilize posts from reddit.com/r/EmojiPasta which is a sub-reddit dedicated to 'EmojiPastas'. 'EmojiPastas' are long strings

of text inter-spaced with frequent use of emoji, usually for humorous effect. Reddit has a well-documented API with plenty of examples online for use in python. This dataset allows for an informal view of emojis, allowing us to represent emojis based on their usage in social media.

For our particular experiments, we start by training new embeddings for emojis both from scratch and on top of another previously generated set utilizing emoji2vec, a system developed by Eisner et al. (2016). These were trained using continuous bag of words, which is fairly typical for training embeddings over a new dataset. Continuous Bag of Words works by training a model to predict what the target word is based on the average of the embeddings of the surrounding window, and allows for training of new embeddings as utilized in works such as word2vec. In particular, we chose a window of the four surrounding words, which we viewed as capturing a sufficient amount of the surrounding context. The performance of this model also gives a measure of how easily this data can extract the relevant information from the existing embeddings.

We then utilized a logistic regression and a few neural networks, each trained as an n-gram language models on these embeddings, allowing for some level of comparison of the utility of embeddings and their performance for a real task. N-gram was chosen due to the nature of the dataset. 'Emojipastas' are, as previously mentioned, built to be linear with emojis appearing after words to be related to the words immediately proceeding them.

We specifically chose to use a value of $n = 5$, as many of the groups of words and associated emojis are not particularly longer than this. For evaluation, we will attempt to use our N-Gram model to predict which emojis are present in future documents.

One potential issue is that our dataset is limited to the emojipasta subreddit, which could lead to biased classifers. Additionally, n-gram models and neural networks both require a large volume of data for training. However, the subreddit has only been active for the past two years and is relatively niche, as a result we were only able to find around 500 quality posts. And finally, these models demand a high computational cost. Utilization of a GPU helped alleviate some of the computational load, but the models still take a non-trivial amount of time to train and iterate over.

3 Experiments and Results

3.1 Dataset and Preprocessing

As previously mentioned, we used posts from the Emojipasta subreddit. We first used the reddit API to pull posts from `reddit.com/r/EmojiPasta`. To preprocess the dataset, we deleted all posts shorter than 20 characters, posts with an emoji to word ratio of less than 10 percent or greater than 60 percent, and posts with an excessive amount of swear words, punctuation, and special characters. We started with around 1000 posts, and after preprocessing we ended up with around 550. Then, we used SKLearn's test/train split function to randomly split the data into a ratio of 85 % training data and 15 % test data for the n-gram classifiers.

We then utilized the test and train datasets for creating more specialized datasets within PyTorch. For use in continuous bag of words, these were broken into examples of windows and target words. To ensure our embeddings were emoji focused and to ease computational burden, we did not include any windows that had an emoji in either the window or context. Windows were of size 4, which included the two words before and after the target word. For use in the n-gram model, each dataset was broken into a set of 5 words, the 4 previous words as context words and a target word. This time, to ensure our model captured the entire context, we included all sets of 5 words, even if this did not include an emoji. This ensured that our model could also predict other parts of the dataset. For a fair comparison to previous literature, we limited the dataset to only include emojis that were present in both emoji2vec and our dataset.

3.2 Experiments

First, we generated the embeddings. Throughout our project, we utilized the pretrained GloVe embeddings for normal English words. For emojis, we created a total of four sets of embeddings. The first was a totally random and untrained set of embeddings for each of the words, which was used as a baseline. The second set were embeddings created by the emoji2vec system and GloVe embeddings (Eisner et al., 2016). The third and fourth sets were generated by utilizing randomly generated embeddings (similar to first set) and the emoji2vec generated embeddings (second set) respectively. These last two sets were trained further over the dataset by performing the continuous bag of words model. We will refer to the emoji embeddings that were initialized with emoji2vec as the hot start embedding. This left us with four sets of potential embeddings for emojis to compare.

After finishing the CBoW process, we decided to visualize our results to better understand our model. We used https://projector.tensorflow.org/ to create a Univeral Manifold Approximation of our created emoji embeddings, as well as calculating cosine values for each of the pairs of emojis. We decided to plot each of our embeddings.

Figure 1 displays the nearest points to the smiling with teeth emojis in the emoji2vec embeddings. UMAP was able to succesfully find all the emojis with the nearest cosine distance in the embedding space and place them close in the 2 dimensional space. The closest emojis appear to be mostly smiling faces.

Then, for comparison we decided to investigate some of the major clusters that appeared in our emoji vectors made with a random start.

Figure 2 shows the main major cluster that was formed for the random start CBoW embedding. This

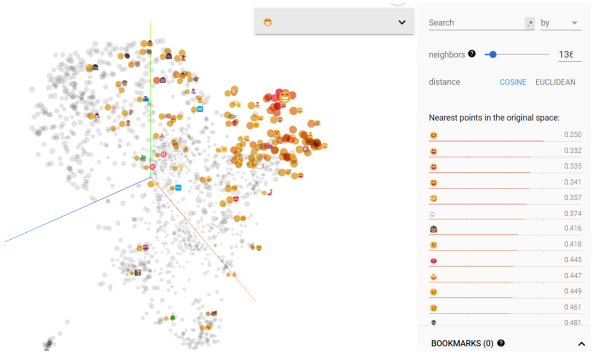

Figure 1: UMAP Cluster of Smile Face Emojis from emoji2vec

cluster appears to be mostly consisting of Keycap emojis, otherwise known as the emojis that just have a number on them. However, the cluster seems noisy since the nearest emojis based on cosine distance include outliers such as :person_golfing:.

The most interesting result appeared in our embeddings trained from the emoji2vec hot start. In Figure 3 we show which emojis appear close to the :film_frames: emoji. It was able to place the :film_frames: emoji, :popcorn: emoji, and :camera_with_flash: all within reasonable distance to each other. This was not something that happened with the emoji2vec embeddings, as they clustered popcorn with other food items and the camera emoji with other technological items. Although the physical unicode description of each of these clustered emojis are not similar, they are usually used in similar context on social media. After all, people are far more likely to think of popcorn when they are talking about the movies or film than if they are discussing food such as sushi or hamburgers. Therefore, it would make sense that they closer together in the embedding space. Perhaps, with more than 500 posts to train on we would be able to further solidify this cluster and perhaps find new clusters with emojis hidden meanings.

Finally, we trained these different n-gram model architectures on each of the types of embedding sets utilizing a train and test dataset. The models were a logistic regression model, a single layer neural network with hidden size 12, a single layer neural network with hidden size 20, and both of these with a 20% dropout rate after the first layer. The models trained over completely randomized embeddings and emoji2vec based embeddings serve as a baseline, as they represent performance where no additional information is encoded and the results of other previous works respectively.

This leads to our third research question about downstream performance, giving us insight into comparative performance between embeddings tuned by the dataset and base models in at least one task.

3.3 Results

Our exploration of the trained CBOW emoji embeddings had a few promising results, and helped address our first two research questions. Our final remaining research question then turned to our various neural n-gram models. To form our models, we created an input by taking the previous n words and embedding each using the tested embeddings for emojis or GloVe for English words. These were then concatenated to form the input for each of the models. An important thing to note is that the dataset in total contained about 10998 words and emojis, so the final results used the LogSoftmax to form the probabilities of each output word. N-gram is, as previously mentioned, a hard task as well that by definition should not have a particularly high accuracy due to multiple next words all being realistic choices. An alternative interesting metric is loss, which we used negative log likelihood (that is,

$$l(y, c) = -\frac{1}{|y|} \sum_{i \neq c} \log \left(\frac{\exp(y_i)}{\sum_j \exp(y_j)} \right)$$

where c is the index representing the target word, and y is the output of the model) for our use case. A quick summary of some statistics for each model are in Table 1 1

Our logistic regression performed poorly on our data, with accuracy plummeting after just a few epochs and continuing to decrease until the test was over. Similarly, the average loss increased logarithmically. Due to the complex nature of modeling n-grams without a full probability distribution, a less expressive model, such as logistic regression, is likely underfitting this model and attempting to memorize the maximum number of test cases without learning the underlying relationships. Interestingly, after 100 training iterations, fully randomized and eomji2vec embeddings had a significantly lower average loss on the test set (13-14 average loss) than either of their tuned versions (17-18 average loss). This difference suggests the embeddings aided in the underfitting. For more information about this particular case, graphs involving this training are available along with the figures relevant to all the other models on the project's GitHub page [1].

The second two models are single-layer neural networks with a hidden layer of size 12, both with and without dropout. The cases with dropout tended to have a slightly smaller total loss. After 100 iterations, the loss for randomized embeddings was the highest, though not by an overly significant margin, followed by the tuned emoji2vec embeddings, the freshly generated embeddings from scratch, and the original emoji2vec having the best performance. Maximum accuracy held a similar pattern and performed marginally better than the logistic regression. Without dropout, the best performer changed to the embeddings generated from scratch, while the randomized embeddings still performed the worst. Generally, performance was similar between both cases in terms of loss and accuracy. The loss curve

[1] https://github.com/aarcc530/cs577-emoji-embeddings/

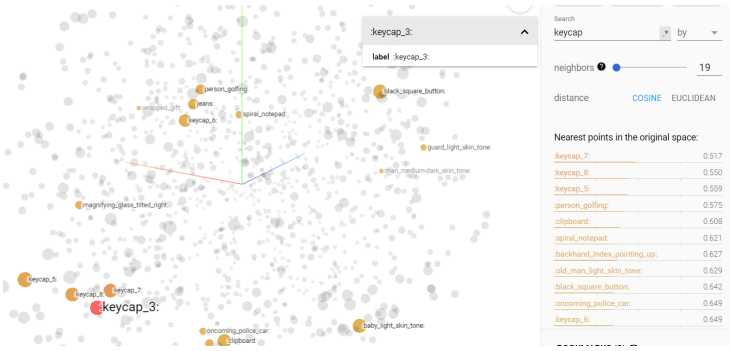

Figure 2: UMAP Cluster of Keycap Emojis from our CBoW Emoji Embeddings

Model	Logistic	NN-12	NN-12D	NN-20	NN-20D
Best Max Acc.	CBOW-0.91%	CBOW-0.99%	CBOW/E2V-0.96%	CBOW-0.98%	CBOW-1.05%
Worst Max Acc.	E2V-0.70%	Rand-0.89%	Rand-0.91%	E2V-0.84%	Rand-0.81%
Best Loss at 100	E2V-13.34	CBOW-10.56	E2V-10.43	CBOW-11.06	CBOW-10.90
Worst Loss at 100	CBOW-18.02	Tuned-10.64	Rand-10.58	Rand-11.14	Rand-11.04

Table 1: Results from Each of the n-gram models. Rand are the randomized embeddings, CBOW are the trained from scratch emeddings, E2V are the emoji2vec embeddings, and Tuned are the embeddings trained over emoji2vec using CBOW

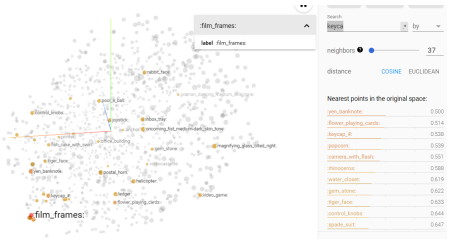

Figure 3: UMAP Cluster of Movie Related Emojis from our CBoW Emoji Embeddings emoji2vec start

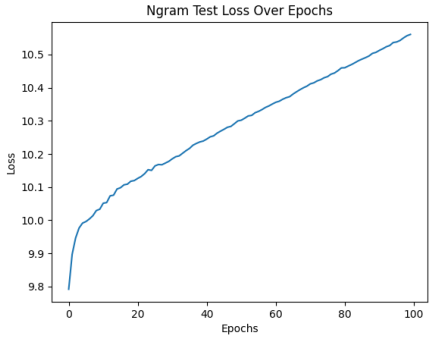

Figure 4: Loss Curve from n-gram on our CBoW Emoji Embeddings randomized start for a Linear-12 Neural Network.

for the trained-from-scratch embeddings for this case is shown in Figure 4. All loss graphs have more or less the same shape but scale at different rates. The final two models are single-layer neural networks with a hidden layer of size 20, again both with and without dropout. A pattern similar to the previous two neural networks emerged, where our from-scratch embeddings performed best and randomized performed the worst.

Overall, while these results show that every model is likely not learning the dependencies of the words within the dataset but rather some basic heuristics, the results also suggest that in the more expressive models, our embeddings did help to some extent. However, significantly more testing should be performed in the future for a more complete judgment.

3.4 Future Variants

N-gram is a complex problem to model effectively and can vary greatly depending on the dataset. Using a different architecture to form n-gram models could also cause a significant performance increase. We focused on embedding training and evaluation methods such as cluster identification and visualization rather than n-gram model optimization.

The most significant variant of our methods that require investigation is the effectiveness of our embeddings for other learning tasks. Other use cases exist wherever emojis might appear. We planned to perform a sentiment analysis test, but the overhead required to annotate a new dataset prohibited such an analysis from being performed within this project's scope.

4 Discussion

4.1 What We Learned

This project taught us that embedding emojis can be a difficult task. Previous papers claimed to have difficulties training emoji embeddings on data from social media, citing issues with bias and inconsistent use of emojis. We believe these issues played a significant role

in our project, as we did not have enough data to accurately predict how future Redditors would use emojis in their post. Another issue may be because emojis can have multiple disjoint hidden meanings apart from the literal. For example, the "eyes" emoji represent someone's eyeballs, but there are two hidden meanings. The first hidden meaning is when eyes represent admiration of someone, and the second is when someone is anticipating or reacting to drama. Neither our data nor our model was large enough to capture these complex relationships. Since simple word embeddings have difficulty representing words with multiple meanings, perhaps a context-based embedding such as ELMo would be preferable.

4.2 Thoughts on Extension

Given an extra month, we would likely try to find new data sources for our emoji embeddings. Although the quality of our data set was good, the quantity was not large enough to create meaningful representations.

We might also try other models to test our embeddings, such as a sentiment analysis task, though this would require some annotation. We might also try to create alternative models to perform the n-gram task, as the models we trained were simplistic when compared to other potential models.

Given an extra year, we would probably try to find more elaborate embedding techniques and investigate the application of the embeddings on other natural language problems. Emoji use is widespread in many contexts, and we want a better representation of those contexts and related changes in meaning. An investigation into adapting something like ELMo to emoji contexts might be a good idea.

In 5 years, we may be able to create an app that recommends emojis to the user based on what they have typed. For example, if somebody makes a sarcastic remark, we could recommend the emoji with the eyes rolling up into its head. We might also be able to create a generative language model or something similar.

References

Elena Barry, Shoaib Jameel, and Haider Raza. 2021. Emojional: Emoji embeddings. *Advances in Intelligent Systems and Computing*, page 312–324.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bosnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *CoRR*, abs/1609.08359.