

Emoji Predictor: Introduce emoji embedding to Emoji Semantics Modelling

Lulu Wan
ucaklw0@ucl.ac.uk

Su Shen
ucabs06@ucl.ac.uk

Qiqi Zeng
ucabqze@ucl.ac.uk

Abstract

The widespread use of emoji has spawned a new way of emotional expression and communication. Emoji prediction according to users' text greatly improves the efficiency of passing emotional information. Many predecessors have studies on the emoji prediction and have applied many Natural Language Processing models. To test if we can improve the prediction accuracy based on existing models, we extended their works by adding the pre-trained emoji embedding to neural network models (LSTM model and CNN model in this paper). Instead of using Softmax to pick out the most likely emoji, we predicted emojis according to cosine similarity between the sentence vector and its corresponding emoji vector. The experiment is based on a dataset of 7480 pieces of text messages and it includes 7 emojis. The result shows that this approach can greatly improve the performance of emoji prediction.

Keyword: emoji embedding, emoji prediction, sentiment analysis, cosine similarity.

1 Introduction

The explosion of Social network platforms such as Instagram, Twitter and Facebook has changed way of people communicate. Emojis, as small ideograms depicting emotions, have become one of the main components in such communication and increased exponentially (Cappallo et al., 2015). It enriches the textual conversation, denote the sentiment and act as a substitute for non-verbal

elements like facial expression, pronunciation and intonation (Barbieri et al., 2017).








According to research study for Instagram, emojis are present in up to 40% of social media messages over the world (Dimson, 2015). The emoji “😭”, officially called the ‘Face with Tears of Joy’, even was regarded as the Oxford Dictionaries word of Year in 2015, which became the first pictograph word of the year ever (Oxford Dictionaries, 2015). These years, a number of social media platforms update their emoji library constantly to better user experiences (Li et al., 2017). It was reported that Apple iOS 12.1 introduced more emojis to the keyboard that are supposed to represent global users, including moon cake, red gift envelope and nazar amulet (Apple, 2018).

The boom of emoji usage stimulates the study of semantics emojis from a Natural Language Processing (NLP) standpoint, it quickly become a novel area that attract much attention. To explore the relationships between emojis and text messages and predict which emojis are evoked by text-based messages, researchers exploit a wide range of machine learning architectures such as Deep Neural Networks, Support Vector Machine, Long Short-Term Memory (LSTM). However, it seems that most current Natural Language Processing algorithms that applied to emoji prediction are constructed with pre-trained sets of word embeddings and representation learning (Eisner, Rocktachel et al., 2016). Eisner et al., (2016) produced the Emoji2vec, pre-trained emoji embeddings trained on Unicode descriptions of emojis for improving downstream task of word2vec embeddings.

Greatly inspired by this idea, we intend to introduce emoji embedding to our emoji prediction model and forecast which emoji in Table1 is the emoji most likely to be evoked by a randomly given sentence. The process could be examined in

the demo. For the new built emoji predictor, we first applied two machine learning models (Convolutional Neural Network and Long Short-Term Memory) to generate the embedding of the sentence, and then matched it with the emoji embedding (emoji2vec) through cosine similarity so that we can find the most corresponding emoji. The main objective of this project is to demonstrate that this approach might perform better than traditional classifiers (Baseline models).

Table 1. Emoji display

0	1	2	3	4	5	6
						

The baseline method of this project is to develop Convolutional Neural Network (CNN) Long Short-Term Memory (LSTM) classifiers to get sentence vector and apply the Softmax layer to compute the probability distribution and select the emoji with the maximal probabilities. In terms of the training process, we use negative log likelihood loss function to train the models and assess the performance with several evaluation metrics, namely F1-score, recall, precision, accuracy.

The main outcome: empirical experiments indicate that the new built model with CNN and introduced the pre-train trained emoji2vec is proved to outperform than baseline models.

2 Related Work

There have been many works in emoji prediction and comparison of the performance based on different neural network models. The works of Coman and Zara (2018) explained that the potential of simpler models could achieve a higher accuracy than some complex models through the fine-tuning of hyper-parameters. In their experiment, they applied three different word embedding methods, including randomly initialized word embeddings, GloVe11 embeddings and trainable GloVe embeddings (using Glove Algorithm but can be modified by learning process). Their results showed that the fixed Glove Embedding lead to a poorer performance than random embedding, however, making the Glove embedding trainable (can be adjusted during training process) can lead to a much higher accuracy, even higher than that of random embedding. Therefore, in this paper, we made the pre-trained embedding model trainable, and the parameters can be fine-tuned. In addition,

they compared the performance of two Neural Network strategies, which are Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM), and the result shows that CNN can reach a slightly higher accuracy than LSTM. However, they did not explain the reasons for this result, so we applied both LSTM and CNN in our experiment to prove their result.

Zhao and Zeng (2017) explained Long-Short Term Memory (LSTM) has been proven to be very effective in language modelling and many other sentimental analysis tasks. However, specific to this emoji prediction task, they compared the performance of Long-Short Term Memory (LSTM) model and Convolutional Neural Network (CNN), and the result showed that CNN achieved a much better accuracy (40%) and F1 scores (46%). Again, they did not give a full explanation to that result. Additionally, they also compared the performance of using random embedding and using pre-trained embedding model (Glove). It indicated that using pre-trained embedding model can lead to a slightly more accurate result. Moreover, in their report, they mentioned that a team at Dango used an RNN to convert sentences and emoji to embeddings, and they choose the best matching emoji by calculating the cosine similarity. This give us a lot of inspiration for our project. In the traditional training model for emoji prediction, emojis are treated as labels, which cannot show the relationship among emojis and how emojis distributed. With the emoji embedding, emojis can be converted to vectors and can be visualized by projecting them into the same 2-D dimensional space. Thus, in this project, we decide to calculate the similarity of sentence vector and emoji vectors in order to decide the best matching emoji, and this method is expected to better distinguish expressions with similar meanings.

The works of Eisner, Rocktachel and Augenstein et al. (2016) pointed out that currently existing pre-trained set of word embedding only contain few emoji representations. Since the usage of emojis in social media has increased, they released a pre-trained embedding for emojis: *emoji2vec*. It maps emoji symbols into 300-dimensional vectors, which are the same space as Google News *word2vec* embeddings. This package provides an important source for our project. Using the *emoji2vec*, we can convert emojis into vectors and then compared them with sentences vectors.

3 Background

The history of emojis date back to 1997, when the first set of emojis was released by J-Phone in Japanese (Luongova, 2017). The word ‘emoji’ comes from the Japanese words for ‘picture’ (e) and ‘character’ (moji). It is not until 2010 when 722 emoji characters were finally standardized by Unicode and Apple iOS5 iPhone operation with keyboard support such character in 2011, emoji became a global phenomenon and increased exponentially (Lucas, 2016). Emoji is a kind of small visual icon that used to convey semantics on Internet. Nowadays, people are so keen on using emojis in Social Media and Instant Messaging Platforms. For example, they use ‘😊’ to represent happy and joy, ‘😕’ for confused, ‘😞’ for sad, ‘😱’ for fearful, ‘😡’ for angry etc. In normal communication, people can convey their emotions through pronunciation, facial expressions, and body language while this cannot be realized on the electronic communication. The popularity of emoji is mainly because it makes up for the lack of sentiment expression in text messages.

Despite the widespread of emoji characters, Researches and works on the relationships between text and related emojis are still scarce (Groot., 2018). Currently, most studies usually research and predict emoji usages based on its corresponding textual information.

However, several main points make this problem different from traditional sentiment analysis and can be considered as a challenging problem. 1) ambiguity of emojis, which can also be regarded as user demographics issue. Miller et al., (2017) indicated that different people have different habits on assigning emojis and may not always have the same interpretations for the meaning of emojis. Because of such ambiguity, it seems that emojis can sometimes mislead humans. 2) similarity of emojis. Since the sentence vector connects to non-computable labels directly, the usages of emojis can be complicated and subtle and it is difficult to distinguish the emojis with similar meanings (Zhang et al., 2018). Based on this fact, our motivation is to replace the traditional classification with a matching layer that match the utterance embedding with the emoji embedding (*emoji2vec*) and produce the degree that helps to find most likely emoji and address two problems mentioned above.

This emoji predictor model is considered to be functioned as an emoji recommender during the online communication in the social media. It can also support for an automatic human-computer conversation system, the machine may act more

like human and looks more attractive and proactive if the dialogue is followed by emojis (Li et al., 2017). Figure 1 is the demo of such application on ‘Siri’, an intelligent assistant on Apple.

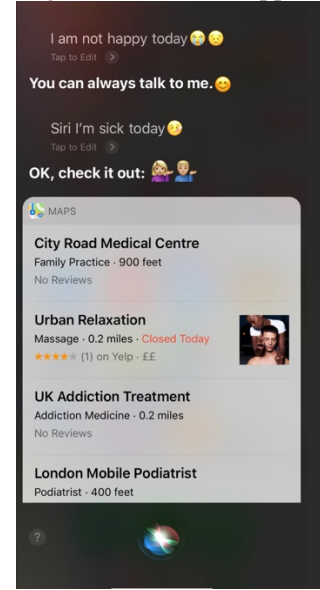


Figure 1: Example of Siri

4 Methods

4.1 Task definition

Given a sentence set $Y = \{y_1, y_2, \dots, y_n\}$ and an emoji set $X = \{x_1, x_2, \dots, x_k\}$, our aim is to train a classification model which could predict the correct emoji $g(y) \in X$ for a sentence y by using the pre-trained word embeddings and emoji embeddings, matching the emoji embedding with the embedding of the text from a series of layers using cosine similarity formula. We would like to verify whether this approach performs better than traditional classifiers (our baseline models).

4.2 Model Structure

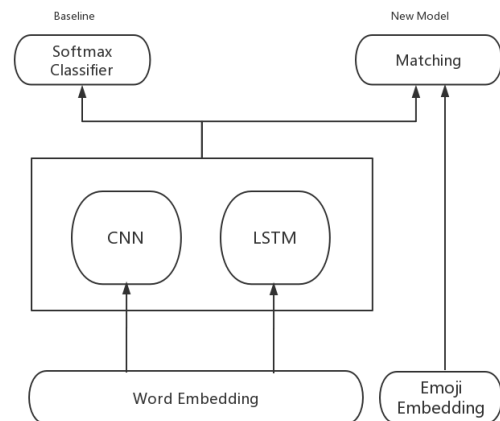


Figure 2: Model Structure

The structure of model is shown in Figure 2. In the baseline routine, this project will firstly convert words into vectors so that the machine can understand them. Then we use the CNN model or LSTM model to get a sentence vector (its weights of each element can be adjusted by thousands of trainings). Then Softmax Function will be applied to calculate the probabilities corresponding to each emoji label, and directly pick out the emoji with the maximal probabilities. However, since the sentence vector connects to non-computable labels directly, it is hard to distinguish the emojis with similar meaning. Therefore, in the New Model routine, to try to improve the performance, and get a more accurate prediction, we added cosine similarity layer to the matching process. In this new model, the emojis are also converted to vectors to take part in the training process. By computing the distance between output vector in the previous layer and emoji embedding we can select the most likely emoji.

4.3 Baseline Algorithm

We set baselines to compare with our proposed approach. If our emoji prediction models outperform the baselines in terms of those metrics, we would suppose the model is acceptable.

LSTM: Recurrent Neural Networks (RNN) is proven to be a powerful model for processing sequential data. With the application of Long-short Term Memory layer, it becomes more powerful since it has different gates to deal with the memory and mitigate the effect caused by gradient vanishing problem. To be specific, the forget gate removes the information that is irrelevant, and the input gate manages updating values (Hochreiter and Schmidhuber, 1997).

CNN: Compared with other neural network architectures, where the number of parameters to be trained grows really fast when it contains lots of hidden layers, the convolutional neural networks only manage particular numbers of connections from the last layer by means of a sliding window of a fixed size. After the convolution, a pooling layer is applied, which includes three different types Max Pooling, Mean Pooling and Sum Pooling, in order to reserve most significant features.

4.4 Layers of Best-performed model

Word embedding Layer: This layer aims to get the distributed representation of each word. Since we want to introduce pre-trained emoji vectors which based on the Google News word2vec embeddings, to be consistent with that, in this layer we use pre-trained Google News word embeddings as well. Firstly, we build our vocabulary according to training data. Secondly, we obtained the embedding $e_1(w_i)$ of each word w_i in the vocabulary from pre-trained embedding model, composing the weight matrix $W_1 \in R^{V \times D}$, where V is the length of the vocabulary and D is the dimension of word embeddings, i.e. 300 in this case. This matrix is trained as the training proceeds.

Convolutional Layer: The convolutional structure of neural network is believed suitable to be implemented on the work of sentiment analysis. It can extract n-gram feature of a sequence, especially for a short text. Different from fully connected layers, CNN uses the concept of sliding windows, which is like a local feature extractor, to catch the important information from word embeddings. Assume the kernel size is k , and the corresponding word embeddings are $e_1(w_1), e_1(w_2), \dots, e_1(w_t)$, then we have:

$$y_1 = f(W_1[e_1(w_1); e_1(w_2); \dots e_1(w_t)] + b_1) \quad (1)$$

where f is the non-linear activation function, W_1 is the weight matrix and b_1 is the bias vector which will be trained during the training process. In practice, we choose a kernel size of 2, which performs in a better way than a kernel size of 3 in our project.

Pooling Layer: After the convolutional layer, we get a series of local features. Since we need to synthesize these local embeddings into one vector as the distributed representation of the whole sentence, a pooling layer is applied here. Here, we choose max pooling. Let $y_1^1, y_1^2, \dots, y_1^h$ be the output vectors from the convolutional layers, where h is the hidden dimension of convolutional layer. and

$$y_2[i] = \max(y_1^i). \quad (2)$$

After this layer, y_2 as the sentence embedding is obtained and the dimension of the embedding is decided by hidden dimension of convolutional layer, i.e. the numbers of convolutions we have applied, which is 200 in our project. Thus, it can be seen that a sentence embedding is also a vector to

represent the utterance in a low-dimension space, indicating latent information.

Hidden Layer: We use a hidden layer of full connection to apply a linear transformation to the incoming data, which is calculated by:

$$y_3 = W_2 y_2 + b_2 \quad (3)$$

where b_2 is the bias vector. Then we could output another 300-dimensional vector from this layer to denote the input sentence.

Emoji Embedding Layer: As described in the part of related work, we took advantage of the pre-trained model: *emoji2vec*. The pre-trained emoji vectors are meant to be used in conjunction with *word2vec*, and are therefore 300-dimensionsal. The emojis we used in our project are 😊 (joy), 😨 (fear), 😡 (anger), 😞 (sadness), 😤 (disgust), 😳 (shame), 😇 (guilt). Hence, what we are supposed to do firstly is to use “*emoji2vec*” to find the corresponding vectors of these seven emojis. Each element is one parameter of the neural network, which gets trained during the training process. This in turn helps update the vectors representing emojis in a more concrete way.

Matching Layer: In this layer, we matched the embedding of the text y_3 from the linear hidden layer with the emoji embedding $e(x_i)$, $i=1, 2, \dots, 7$, and then obtained the scores corresponding to each emoji which could indicate their matching degrees. We utilized the cosine similarity as the measurement of matching and then have:

$$score(y_3, e(x_i)) = \frac{\langle y_3, e(x_i) \rangle}{\|y_3\| * \|e(x_i)\|} \quad (4)$$

Since a higher score indicates a smaller distance between these two vectors, we choose $\arg\max_{x_i}(score(y_3, e(x_i)))$ as the final emoji representing the input sentence.

Algorithm 1: Process of training one sample

Input: One sentence, its correct emoji(label)

Output: Updating model parameters

Implementation:

For each emoji x_i do:

1. Forward propagation to calculate the matching scores between the vector of the given sentence and each emoji vector
2. Calculate the loss using the CrossEntropyLoss function
3. Backward propagation to update model parameters using SGD method.

4.5 Training

In the training process, we use the CrossEntropyLoss as our loss function, which is useful to train a classification problem with multi classes. Not surprisingly, we exert stochastic gradient decent method as our optimizer to train our proposed neural networks, and adjust the learning rate to get the best performance on the dataset. In our project, after comparing the performances of models of different hyperparameters, (here we only talk about CNN classifier using pre-trained emoji embeddings and word embeddings), the best one is with the hyperparameters learning rate:0.01, best_epoch:5.

5 Experiment

Datasets: The dataset that we used is collected from the DeepEmoji GitHub repository. The data consists of a text message with an emoji as its label. There are 7 kinds of emojis included in this dataset, which respectively represent joy, fear, anger, sadness, disgust, shame and guilt. This dataset was applied to our model to train the parameters of all the NLP model in this project.

Open-Source project: we used the Google News word2vec and emoji2vec to get the word embeddings and emoji embeddings.

Language: we used pytorch to implement our project.

Metrics: In our project, we evaluate the performance of the model via Accuracy, Precision, Recall and F1 score metrics. The specific results are revealed in table3 which will be discussed in detail in next section.

Accuracy: It is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

Precision: It is the ratio of correctly predicted observation to the total predicted positive observations.

Recall(sensitivity): It is the ratio of correctly predicted positive observations to the all observations in actual class.

F1 score: It is the weighted average of Precision and Recall. Intuitively it is not as easy to understand as accuracy, but F1 is actually more useful.

6 Result and Discussion

Table2: Experimental performance of Emoji predictors on the testing dataset (1486)

Metrics	Accuracy	MacroF1	Recall	Precision
LSTM Baseline	0.35	0.33	0.33	0.33
CNN Baseline	0.43	0.40	0.41	0.42
LSTM emoji embedding	0.40	0.32	0.34	0.40
CNN emoji embedding	0.60	0.58	0.61	0.63

Table3: CNN emoji embedding

Emoji	Macro F1	Recall	Precision
😊 joy	0.69	0.73	0.67
😨 fear	0.67	0.62	0.74
😡 anger	0.45	0.37	0.57
😞 sadness	0.65	0.70	0.61
😤 disgust	0.55	0.66	0.48
😳 shame	0.52	0.51	0.53
😓 guilt	0.52	0.52	0.53

As it shown in the evaluation metrics, the new built model of CNN classifier with pre-trained emoji embedding achieved the best results out of all the models, with the accuracy of 0.60 and F1 score of 0.58. In terms of the F1 score of each individual emoji out of the 7 emojis, the emoji ‘😊 joy’, which is the only emoji expressing positive emotion, got the highest F1 score at 0.69. The emoji ‘😤 disgust’, ‘😳 shame’ and ‘😓 guilt’ got a relatively low score (0.55, 0.52 and 0.52 respectively). That might be because these three emojis express similar emotions, and therefore they are harder to be distinguished.

From the experiment results, it can be concluded that CNN performs more effective than LSTM for both baseline models and new built emoji predictor

model. The accuracy of CNN baseline is 0.43, which is 0.08 higher than LSTM baseline and for CNN with *emoji2vec*, it is 0.20 higher than LSTM with *emoji2vec*. Some other studies in emoji prediction seem to also obtain the same conclusion. Coman et al., (2018) indicated between the two Neural Network algorithms CNN model is proved to outperform than the LSTM. Zhang and Zeng (2017) also found that CNN model works better than LSTM in their model of predicting emoji usage from Twitter Data.

Another finding is that the application of pre-trained word embedding and pre-trained emoji embedding can improve the performance of neural networks models. In this experiment, based on both LSTM model and CNN model, we applied *word2vec* and *emoji2vec*, and the parameters of these two embedding models are adjustable. After training, we compared the results of our new models, which applied the pre-trained emoji embedding models, with our baseline. In terms of the accuracy, this pre-trained emoji embedding increased the testing accuracy from 0.35 to 0.40 based on LSTM model and from 0.43 to 0.60 on CNN model, and it also improve the F1 score from 0.40 to 0.58 based on CNN model. In terms of the training speed, the accuracy of models with pre-trained emoji embedding rises much quicker than that with random embedding.

Table4: Emoji prediction examples on the model CNN with *emoji2vec*

Sentence	Emoji
I got an offer from University College London.	😊
A Boeing 737 Max8 airplane crashed in Ethiopian.	😨
I heard someone expressed discriminating opinions.	😤
I struggle with countless deadlines these days.	😓
A couple abused animal after drunk.	😡
I said something bad behind my classmate's back which is totally wrong.	😞
He has wasted too much time on watching soap opera.	😓

Seven emoji prediction examples are shown in the Table5 obtained through our most outstanding model CNN with pre-trained emoji embedding. As seen, the model provides appropriate emojis for every input sentence. Such predictions are all linked to the meaning of the message, such as ‘got an offer from University College London’ results in its associated emoji of joy(😊) or a negative polarity (😡) would be evoked in terms of ‘I struggle with countless deadlines these days’. Such

highly-subjective automated emoji predictor will help to better understand human natural languages.

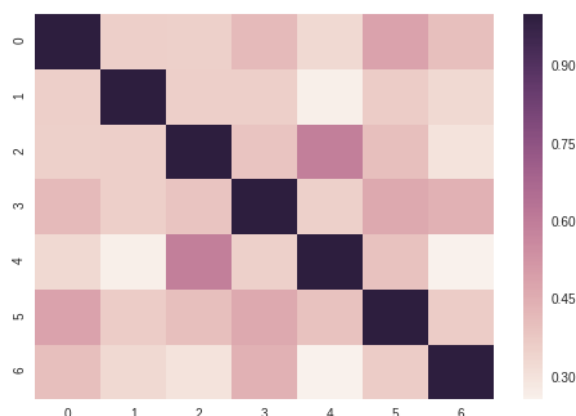


Figure3: The relationship between emojis

After training, we plot the correlation map of the updated emoji vectors. Actually, our emoji set consists of one positive and six negative emojis. However, from this map, “😊” (0) seems to be kind of connected with “😬” (5). This actually reveals some problems we have overlooked and simultaneously not an easy problem to overcome. This is because different people use these emojis from different perspectives and certain emoji does not defined clearly, even indicate completely different polarities. Take “😬” for example, in the official definition, it can not only represent “shame”, but also “blush”. In this case, it could bring a nuisance factor to our experiment.

7 Conclusion and Future Work

In this project, we mainly focus on improving the accuracy of emoji prediction, which translates given text into emojis to help describe users’ emotions. Based on the traditional neural network models, we proposed an “emoji embedding layer” where we use pre-trained emoji embeddings and get them updated during the training process. Empirical results demonstrate that our approach significantly improves both the accuracy and efficiency compared to traditional classifiers.

For the future work, one direction is to establish the project on a dataset of abundant sample size to better construct the correlation map among a large number of emojis and consider contextual information, prompting the machine learn and distinguish various emotions, thus telling the

difference between them. Additionally, we intend to establish a brand-new loss function of cosine similarity for multi classification problems, in which case we may improve the current work to a higher level. As in our work, to some extent, we have not completely fulfilled substituting Softmax layer for cosine similarity due to lack of the corresponding loss function for multi-class. Potential future work can also be implemented with the LSTM model, where an attention mechanism is supposed to be applied to condition the weight of each individual word so that we can focus on most pertinent piece of information, rather than managing all available information.

References

- Andrei Catalin Coman¹, Giacomo Zara¹, Yaroslav Nechaev², Gianni Barlacchi², and Alessandro Moschitti¹. 2018 *Exploiting Deep Neural Networks for Tweet-based Emoji Prediction*. University of Trento, Trento, Italy
- Apple Newsroom. 2018. *Apple brings more than 70 new emoji to iPhone with iOS 12.1*. Available from: <https://www.apple.com/uk/newsroom/2018/10/apple-brings-more-than-70-new-emoji-to-iphone-with-ios-12-1/>. (Accessed: 8 Feb 2018)
- Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. *emoji2vec: Learning Emoji Representations from their Description*. In *Proc. of the 4th Intl. Workshop on NLP for Social Media at EMNLP (SocialNLP)*.
- Catalin, Andrei, Giacomo Zara, Yaroslav Nechaev, Gianni Barlacchi and Alessandro Moschitti. 2018. “*Exploiting Deep Neural Networks for Tweet-based Emoji Prediction*.”. *NL4AI@AI*IA*
- Daphne Groot, Rémon Kruizinga, Hennie Veldthuis Simon de Wit. 2018. *PickleTeam! at SemEval-2018 Task 2: English and Spanish Emoji Prediction from Tweets*. *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 454–458 New Orleans, Louisiana, June 5–6, 2018. ©2018 Association for Computational Linguistics
- English Oxford Living Dictionary. 2015. *Word of the Year 2015*. Available from:

<https://en.oxforddictionaries.com/word-of-the-year/word-of-the-year-2015> (Accessed: 6 Feb 2018)

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. *Are emojis predictable?* In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Volume 2, Short Papers, pages 105–111, Valencia, Spain. Association for Computational Linguistics.

Gavin Lucas. 2016. *The Story of Emoji. Evolution of Emoticons/Emoji and their Functions in Digital Communications*. Gebundenes Buch, Pappband, ISBN: 978-3-7913-8150-3.

Hannah Miller, Daniel Kluver, Jacob Thebault-Spieker, Loren Terveen, and Brent Hecht. 2017. *Understanding emoji ambiguity in context: The role of text in emoji-related miscommunication*. In 11th International Conference on Web and Social Media, ICWSM 2017. AAAI Press.

Luda Zhao and Connie Zeng. 2017. *Using neural networks to predict emoji usage from Twitter data*.

Peijun Zhao, Jia Jia, Yongshen An, Jie Liang, Lexing Xie and Jiebo Luo. 2018. “*Analyzing and Predicting Emoji Usages in Social Media*.” *Companion of the The Web Conference 2018 on The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 2018: 327–334.

Spencer Cappallo, Thomas Mensink, and Cees GM Snoek, 2015. *Image2emoji: Zero-shot emoji prediction for visual media*. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1311–1314. ACM.

Thomas Dimson. 2015. *Emojineering Part 1: Machine Learning for Emoji Trends*. <http://instagram-engineering.tumblr.com/post/117889701472/emojineering-part-1-machine-learning-for-emoji>. (2015).