EMOJI PREDICTION FROM SENTENCE

[1]Madipelly Shashank,[2]Bakki Sumanth,[3]Gopagani Akshitha

[1]Depatment of Electronic & Communication Engineering,SR University,Warangal,Telangana,India

[2]Depatment of Computer Science and Engineering, SR University,Warangal,Telangana,India

[3]Depatment of Computer Science and Engineering, SR University,Warangal,Telangana,India

Abstract

Long short-term memory (LSTM) is an artificial neural network used in the fields of artificial intelligence and deep learning._Long Short-Term Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step. It tackled the problem of long-term dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information. LSTM can by default retain the information for a long period of time. It is used for processing, predicting, and classifying on the basis of time-series data. Emojis are small images that are commonly included in social media text messages. The combination of visual and textual content in the same message builds up a modern way of communication. Despite being widely used in social media, emojis' underlying semantics have received little attention from a Natural Language Processing standpoint. In this project, we investigate the relation between words and emojis, studying the novel task of predicting which emojis are evoked by text-based tweet messages. We experimented variant of word embedding techniques(glove embedding), and trained the models based on LSTMs in this task respectively. Our experimental results show that our model can predict reasonable emoji from sentences.

1.INTRODUCTION

People use emojis every day. Emojis have become a new language that can more effectively express an idea or emotion. This visual language is now a standard for online communication, available not only in Twitter, but also in other large online platform such as Facebook and Instagram. Right now, the keyboard on iOS can predict emojis but only based on certain keywords and tags that are associated with emojis. Emoji prediction is a fun variant of sentiment analysis. When texting your friends, emoji can make your text messages more expressive. It would be nice if the keyboard can predict emojis based on the emotion and meaning of the whole sentence you typed out.

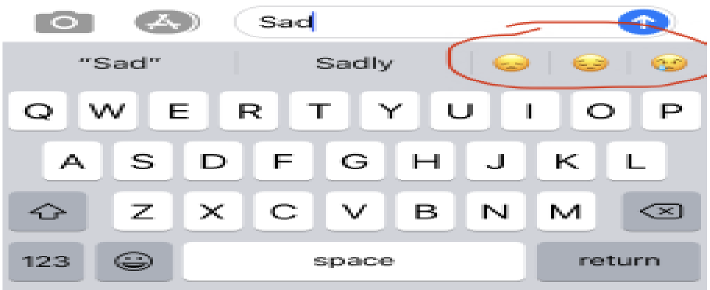
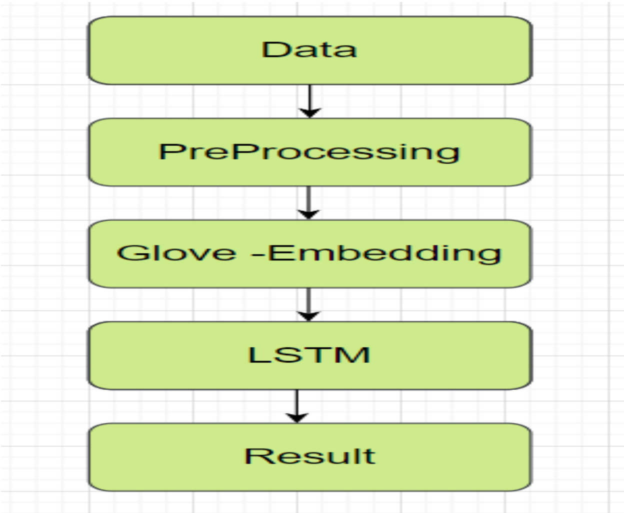


Fig.1 – Example of Emoji Prediction in Keyboard

In this project, we aims to study the relation between words and emojis, studying the problem of predicting which emojis are predicted by textbased messages. We build classifiers that learns to associate emojis with sentences. The models we used here is Long Short-Term Memory LSTM. Pre-trained GLoVe model are used as word embedding, respectively. In this machine learning project, we predict the emoji from the given text. This means we build a text classifier that returns an emoji that suits the given text and we will do this using the Long Short-Term Memory algorithm and glove model for embedding. We train a dataset of sentences with emojis labels aggregated from messages. In the last stage, the trained classifier takes as input a sentence and finds the most appropriate emoji to be used with this sentence.

2. METHODOLOGY:



2.1 DATASET

The datset which we used is downloaded from the kaggle which contains all the information regarding the emojis. Our first goal is to extract feature texts from the kaggle dataset, which will help this model to learn. The dataset contains about 360 instances of text categorized into five categories:

1.love

2.Base Ball

3. Funny

4. Upset

5. Dinner

	A	B
1	never talk to me again	3
2	I am proud of your achievements	2
3	It is the worst day in my life	3
4	Miss you so much	0
5	food is life	4
6	I love you mum	0
7	Stop saying bullshit	3
8	congratulations on your acceptance	2
9	The assignment is too long	3
10	I want to go play	1

Fig.2 – Datset Insights

2.2 DATA PRE-PROCESSING

Data pre-processing is a technique that is used to convert raw data into a clean dataset. The data is gathered from Kaggle which is in raw format (i.e., unbalanced data and unrequired columns present in it) which is not feasible for the computer to predict the sentiment of the given text. Preprocessing for this text data is determined below:

2.3 Removing Unnecessary Columns:

The Columns present in the dataset are 'address', 'name', 'online_order', 'book_table', 'rate', 'votes', 'location', 'rest_type', 'cuisines', 'cost', 'reviews_list', 'menu_item', 'type', 'city'. Initially, the columns url, address, dish_liked and Phone are dropped from the dataset.

2.4 Checking for Duplicate Data:

The Values in the dataset may be duplicate so, there is a need to check for data that is repeating and remove it so that it does not affect the model training and the accuracy that it is giving.

2.5 Checking for Empty Data:

The Values in the dataset may be Null so, there is a need to check for data that is null or not, so that it does not affect the model training and the accuracy that it is giving.

2.6 Converting to Lower Case:

Converting the text present in the reviews column into lower case is necessary since it will be helpful while converting the text data into vectors. These vectors will be later used for training the model and validating it using the validation data. All the text should be in same format for this process.

2.7 Removing Punctuations:

It is also necessary to remove punctuations from the text since they are insignificant to train the model

2.8 Removal of Stop Words:

Stop word removal is one of the most used preprocessing steps across different NLP applications. The idea is simply removing the words that occur commonly across all the documents in the corpus. Typically, articles and pronouns are generally classified as stop words. These words have no significance in some of the NLP tasks like information retrieval and classification, which means these words are not very discriminative. On the contrary, in some NLP applications stop word removal will have very little impact

2.9 WORD EMBEDDING (GloVe):

Word embeddings - are basically a form of word representation that bridges the human understanding of language to that of a machine. They have learned representations of text in an n-dimensional space where words that have the same meaning have a similar representation. Meaning that two similar words are represented by almost similar vectors that are very closely placed in a vector space. These are essential for solving most Natural language processing problems.

GloVe (Global Vectors for Word Representation) is an alternate method to create word embeddings. It is based on matrix factorization techniques on the word-context matrix. A large matrix of co-occurrence information is constructed and you count each “word” (the rows), and how frequently we see this word in some “context” (the columns) in a large corpus. Usually, we scan our corpus in the following manner: for each term, we look for context terms within some area defined by a window-size before the term and a window-size after the term. Also, we give less weight for more distant words.

The number of “contexts” is, of course, large, since it is essentially combinatorial in size. So then we factorize this matrix to yield a lower-dimensional matrix, where each row now yields a vector representation for each word. In general, this is done by minimizing a “reconstruction loss”. This loss tries to find the lower-dimensional representations which can

3. MODEL

In this project we used Long Short-Term Memory(LSTM) algorithm to predict the emoji from a given sentence.

Long Short-Term Memory:

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed-forward neural networks, LSTM has feedback connections. It can process not only single data points (such as images) but also entire sequences of data (such as speech or video). LSTM is an application to tasks such as un segmented, connected handwriting recognition, **or** speech recognition**.**

A general **LSTM** unit is composed of a cell, an input gate, an output gate, and a forget gate. The cell remembers values over arbitrary time intervals, and three gates regulate the flow of information into and out of the cell. LSTM is well-suited to classify, process, and predict the time series given of unknown duration.

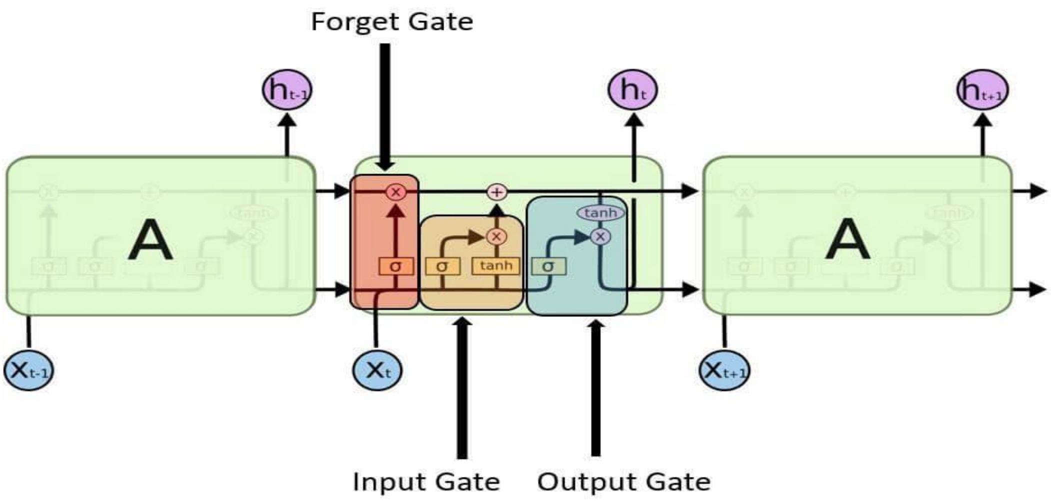
LSTM Architecture:

Long Short- Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory.

ARCHITECTURE OF LSTM:

1. Forget Gate
2. Input Gate
3. Output Gate

Fig.3- Architecture of LSTM



1. Input gate- It discover which value from input should be used to modify the memory. **Sigmoid** function decides which values to let through 0 or 1. And **tanh** function gives weightage to the values which are passed, deciding their level of importance ranging from **-1** to **1**.

$$i_t = \sigma(W_i.[h_t - 1, x_t] + bi)$$
$$C_t = \tanh\,(W_C.[h_t - 1, x_t] + b_C)$$

2. Forget gate- It discover the details to be discarded from the block. A sigmoid function decides it. It looks at the previous state **(ht-1)** and the content input (Xt) and outputs a number between 0(omit this) and 1(keep this) for each number in the cell state **Ct-1**.

$$f_t = \sigma(W_f.[h_t - 1, x_t] + b_f)$$

3. Output gate- The input and the memory of the block are used to decide the output. Sigmoid function decides which values to let through 0 or 1. And tanh function decides which values to let through 0, 1. And tanh function gives weightage to the values which are passed, deciding their level of importance ranging from -1 to 1 and multiplied with an output of **s**igmoid.

$$O_t = \sigma(W_o[h_t - 1, x_t] + b_o$$
$$h_t = o_t * \tanh(C_t)$$

The words are inputted into an embedding lookup. In most cases, when working with a corpus of text data, the size of the vocabulary is unusually large.

This is a multidimensional, distributed representation of words in a vector space. These embeddings can be learned using other deep learning techniques like **GloVe**, we can train the model in an end-to-end fashion to determine the embedding as we teach.

These embeddings are then inputted into our **LSTM layer**, where the output is fed to a sigmoid output layer and the **LSTM cell** for the next word in our sequence.

LSTM LAYER:

We used a function to build the LSTM layers to handle the number of layers and sizes dynamically. The service will take a list of LSTM sizes, which can indicate the number of LSTM layers based on the list's length indicating a two-layered LSTM network. In our model we used a list of length 2, containing the sizes 32 and 16, indicating a two-layered LSTM network where the first layer size 32 and the second layer has hidden layer size 16.

4. RESULTS

For prediction of emoji from a given sentence, we implemented a Long Short-Term Memory (LSTM) model. The LSTM model predicted the emoji with an overall accuracy of 96.88%. Following are the graphs of Accuracy and Loss.

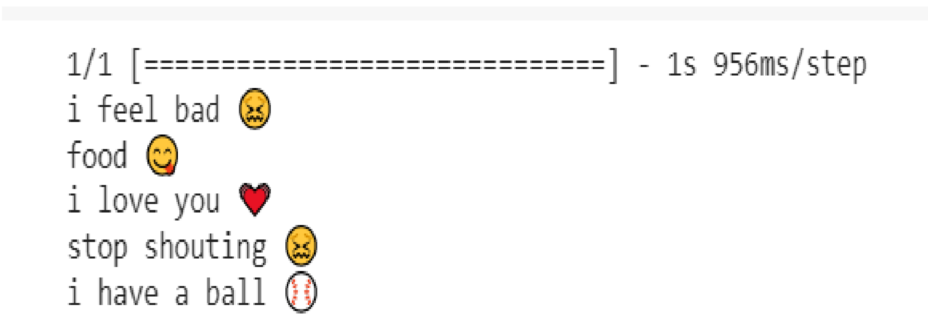


Fig.4- Emoji Prediction

```
Epoch 38/50
9/9 [==============================] - 0s 16ms/step - loss: 0.1087 - accuracy: 0.9894 - val_loss: 0.1822 - val_accuracy: 0.9688
Epoch 39/50
9/9 [==============================] - 0s 15ms/step - loss: 0.0910 - accuracy: 0.9859 - val_loss: 0.1489 - val_accuracy: 0.9688
Epoch 40/50
9/9 [==============================] - 0s 16ms/step - loss: 0.0803 - accuracy: 0.9929 - val_loss: 0.1893 - val_accuracy: 0.9688
Epoch 41/50
9/9 [==============================] - 0s 15ms/step - loss: 0.0750 - accuracy: 0.9929 - val_loss: 0.1490 - val_accuracy: 0.9688
Epoch 42/50
9/9 [==============================] - 0s 15ms/step - loss: 0.0710 - accuracy: 0.9965 - val_loss: 0.1058 - val_accuracy: 0.9688
Epoch 43/50
9/9 [==============================] - 0s 14ms/step - loss: 0.0724 - accuracy: 0.9965 - val_loss: 0.0665 - val_accuracy: 1.0000
Epoch 44/50
9/9 [==============================] - 0s 16ms/step - loss: 0.1168 - accuracy: 0.9823 - val_loss: 0.1789 - val_accuracy: 0.9688
Epoch 45/50
9/9 [==============================] - 0s 16ms/step - loss: 0.0936 - accuracy: 0.9859 - val_loss: 0.2090 - val_accuracy: 0.9688
Epoch 46/50
9/9 [==============================] - 0s 15ms/step - loss: 0.1820 - accuracy: 0.9576 - val_loss: 0.0752 - val_accuracy: 0.9688
Epoch 47/50
9/9 [==============================] - 0s 18ms/step - loss: 0.0819 - accuracy: 0.9894 - val_loss: 0.1207 - val_accuracy: 0.9688
Epoch 48/50
9/9 [==============================] - 0s 16ms/step - loss: 0.0655 - accuracy: 0.9965 - val_loss: 0.0387 - val_accuracy: 1.0000
Epoch 49/50
9/9 [==============================] - 0s 15ms/step - loss: 0.0694 - accuracy: 0.9894 - val_loss: 0.1188 - val_accuracy: 0.9688
Epoch 50/50
9/9 [==============================] - 0s 16ms/step - loss: 0.0640 - accuracy: 0.9965 - val_loss: 0.1599 - val_accuracy: 0.9688
```

Fig.5 – Testing and Training Accuracy

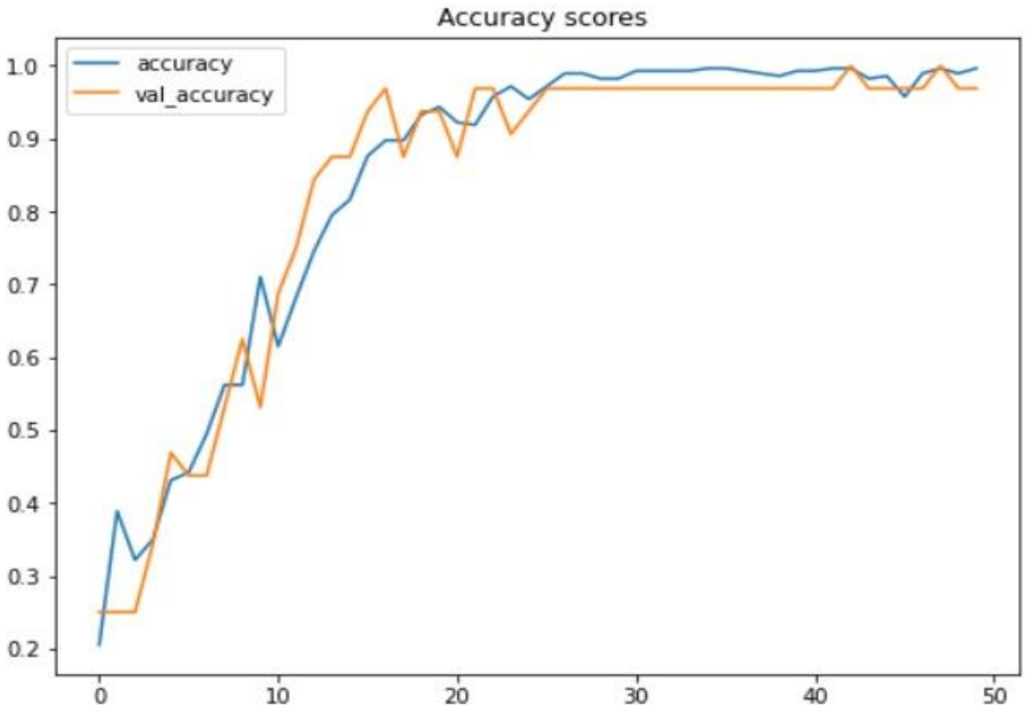


Fig.6 – Accuracy & val_accuray

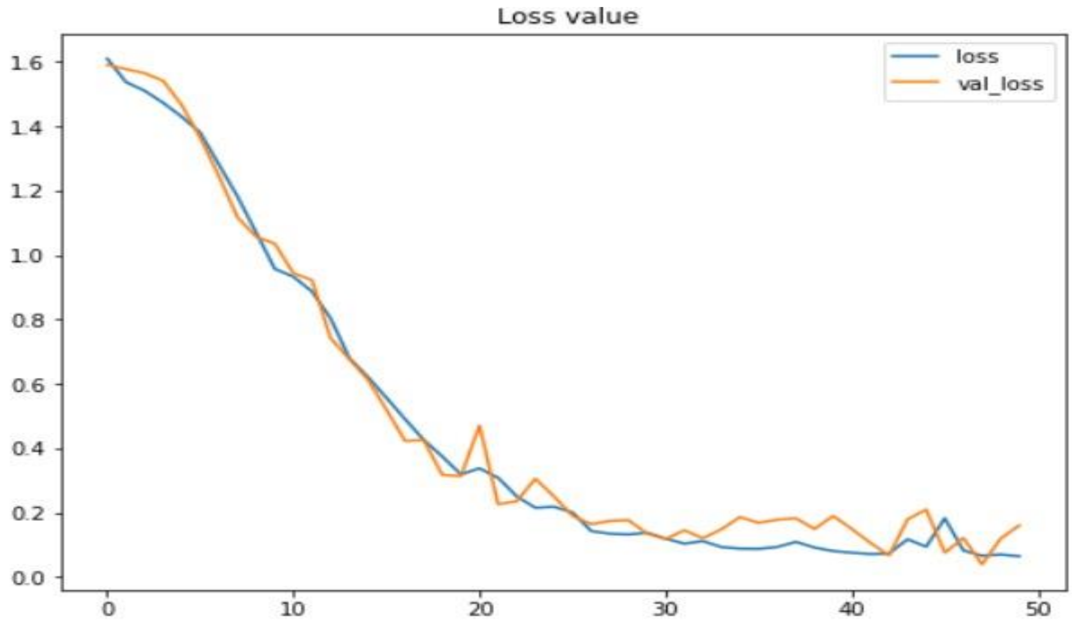


Fig.7 – Loss & val_loss

5. CONCLUSION

Emojis have become a new language that can more effectively express an idea or emotion. There are large amount of datsets and many different machine learning algorithms applied to predict the emoji from a sentence accurately.

In this project, we had successfully built Emoji Prediction project that learns to associate emojis with sentences using Long Short-Term Memory model. We started with a good amount of sentences that contain emojis collected from messages, then looked at features from those sentences, embedded them, created our classifier and trained it to associate certain features with their (known) smileys. Finally, by using our LSTM model users can predict the emojis from a sentence accurately.

6.REFERENCES

[1] Chunhua Zhang, Xiaojian Shao, Dewei Li, "Knowledge-based Support Vector Classification Based on C-SVC", Procedia Computer Science 17:1083-1090, December 2013.

[2] Daniel Dichiu, Irina Rancea, "Using Machine Learning Algorithms for Author Profiling In Social Media Notebook", PAN at CLEF 2016.

[3] S. R. Safavian, D. Landgrebe, "A survey of decision tree classifier methodology," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no. 3, pp. 660-674, doi: 10.1109/21.97458, 1991.

[4] Gerard Biau, “Analysis of a Random Forests Model”, Journal of Machine Learning Research 13, 1063-1095 Submitted 10/10, Published 4/12, 2012.

[5] Nurendra Choudhary, Rajat Singh, Vijjini Anvesh Rao, Manish Shrivastava, Twitter corpus of Resource-Scarce Languages for Sentiment Analysis and Multilingual Emoji Prediction, 2018.

[6] Ronen Feldman, Techniques and applications for sentiment analysis, Communications of the ACM, April 2013.

[7] Jin Wang, Liang-Chih Yu, K. Robert Lai, Xuejie Zhang, Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 225–230, Berlin, Germany, August 7-12, 2016.

[8] M. Thamarai, S P. Malarvizhi, “House Price Prediction Modeling Using Machine Learning", International Journal of Information Engineering and Electronic Business (IJIEEB), Vol.12, No.2, pp. 15-20, 2020. DOI: 10.5815/ijieeb.2020.02.03

[9] Mingrui “Ray” Zhang, Ruolin Wang, Xuhai Xu, Qisheng Li, Ather Sharif, Jacob O. Wobbrock, "Voicemoji: Emoji Entry Using Voice for Visually Impaired People", Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, May 2021, Article No.: 37Mingrui “Ray” Zhang, Ruolin Wang, Xuhai Xu, Qisheng Li, Ather Sharif, Jacob O. Wobbrock, "Voicemoji: Emoji Entry Using Voice for Visually Impaired People", Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, May 2021, Article No.: 37.

[10] Kotaro Oomori, Akihisa Shitara, Tatsuya Minagawa, Sayan Sarcar, Yoichi Ochiai, "A Preliminary Study on Understanding Voice-only Online Meetings Using Emoji-based Captioning for Deaf or Hard of Hearing Users", The 22nd International ACM