Emojify sentences, comparing a simple neural network to Recurrent neural networks

Mohammadmahdi Ghahramani, University of Padova
muhammadmahdi0007@gmail.com
https://github.com/MohammadMahdiGhahramani7/AI_Project

Abstract

Attempting to analyze and understand the sentiment of a sentence has always been useful. Its application ranges from document classification to customer comment classification for companies who are providing a service and would like to automatically get informed about their customers' feelings about their services. This study reviews how to assign an emoji to a given sentence which is the best representation of that sentence. The first contribution of this research is to use a pre-trained word embedder and take advantage of averaging method on sentences' words and consider a simple neural network to classify the sentences. However, in the second contribution, it reveals the fact that by using Recurrent neural networks (RNNs), the performance of the model increases. The dataset used to conduct this research consists of approximately 130 training samples and 60 test samples. Five different emojis are considered to label these samples.

Introduction

Wide applications of sentiment analysis have resulted in the enormous contributions by researchers in the field of Natural language processing (NLP). In this light, this study implements and compares two different approaches to classifying sentences by assigning an emoji to a given sentence. It uses a pre-trained word embedding dataset, Glove 50, to avoid representing the words by one-hot encoding. The reason is that the words in a given comment are more likely to be correlated with each other and it is not a true assumption to consider their representations orthogonal to each other. This word embedding block maps each word to a vector of size 50.

In the first approach, we simply call the vector representation of each word in a sentence and then average them together to represent the whole sentence. By feeding this 50-dimension representation to a simple one-layer neural network, we select the best emoji for the sentence. Although this model is easy to implement and is not computationally expensive, it is not able to detect the dependencies between words. For example, it cannot understand that "*NOT*" in the sentence "*The food was NOT delicious, well-prepared and lovely*" is negating the positive aspects of the rest of the sentences and more likely labels this sentence as a positive sentence. The same intuition is the main reason why RNNs are introduced. They are able to preserve information in sequences to avoid such dysfunctionalities. We use simple RNNs to perform such a task because the sentences are not so long that the danger of gradient vanishing threat the model.

The dataset of 130 training samples and 60 testing samples is used to train and evaluate the model performance. Changing the approach from the first method to the second one improves the accuracy of the model on the test set from accuracy of 77% to 82%. Note that the small part of the dataset contains those exceptional samples that a simple neural network is not able to work well. That is why we see only a 5% improvement in accuracy. The code associated with this research is available on my GitHub of https://github.com/MohammadMahdiGhahramani7/AI_Project.

1. Dataset

The dataset includes a balanced number of sentences regarding the labels. There are five emoji labels, including *Heart*, *Baseball*, *Smile*, *Disappointed* and *Fork and Knife*. Although the similarity between these labels is not zero, this study considers one-hot encoding for these five labels.

2. Simple neural network classifier

One approach to classify the sentences is to use neural networks. However, this question may arise that how to feed this network when the length of sentences might be different. This study to answer this question introduces the averaging method.

Each sentence is made of several words, w_1, w_2, ..., w_n. Using Glove 50, each word is represented by a 50-

dimension vector. By computing the average vector of this representation, we encode sentences. In this way, regardless of the sentence length, all sentences are encoded to a 50-dimension vector which can be fed to the network. Eq1 shows how each sentence is encoded,

$$\forall i \in [1, m] : E(S_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} GR(w_{ij}), \tag{1}$$

where m is the length of the dataset, i.e. the number of all sentences, S_i is the i-th sentence of the dataset, n_i is the number of words in the i-th sentence, GR is a function whose output is the glove-embedded version of its input and w_{ij} stands for the j-th word of the i-th sentence.

Considering a shallow neural network, this study attempts to perform multi-class classification. Figure1 illustrates how this network is constructed.

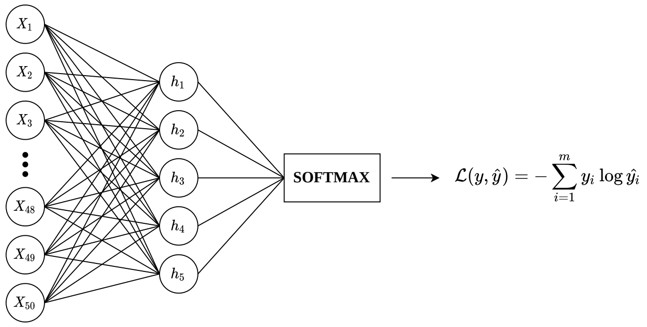


Figure 1: **Approach1.** This network considers a 5-dim hidden layer and connects it to a *Softmax* unit and then computes the loss function. The weights are updated using the back propagation step.

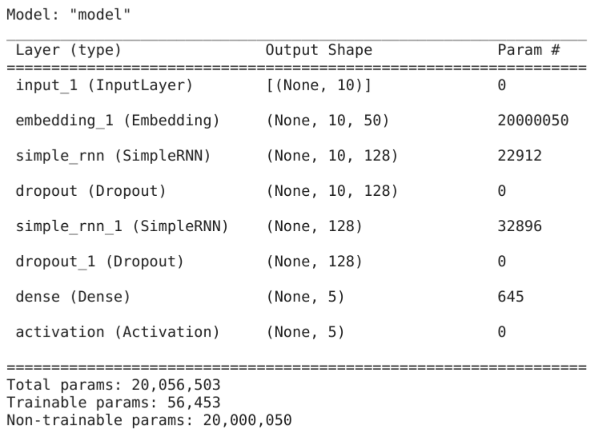
3. RNN-based classifier

The previously mentioned model is considered as a base model for our task, because it does not consider the dependencies between words in sentences and naively sum all words' representation and then divides on the representation vector's length.

In this part, we propose a network that is able to bring the information from the past elements in a sequence. The maximum number of words in sentences is 10. Hence, the gradient vanishing strnegth is not significant. That is why, instead of LSTM or GRU units, this study considers Simple RNN units. Defining one Embedding layer, 2 RNN block, 2 Dropout layers and a Dense layer, this part classifies the data. Figure2 illustrates how this network is constructed.

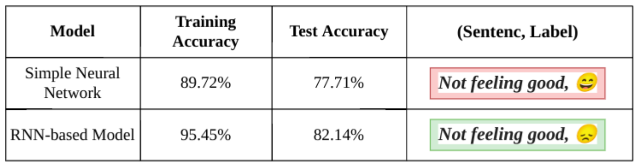
Results

Fig3 reports the results obtained by this study. It shows, the second approach is capables of preserve memory so that it is now able to detect dependency between words.



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 10)]              0

 embedding_1 (Embedding)     (None, 10, 50)            20000050

 simple_rnn (SimpleRNN)      (None, 10, 128)           22912

 dropout (Dropout)           (None, 10, 128)           0

 simple_rnn_1 (SimpleRNN)    (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense (Dense)               (None, 5)                 645

 activation (Activation)     (None, 5)                 0

=================================================================
Total params: 20,056,503
Trainable params: 56,453
Non-trainable params: 20,000,050
```

Figure 2: **Approach2.** The network uses a many-to-one kind in RNNs. That is why the dimension of the data in the second RNN layer is (None, 128) and not (None, 10, 128).



Model	**Training Accuracy**	**Test Accuracy**	**(Sentenc, Label)**
Simple Neural Network	89.72%	77.71%	***Not feeling good,*** 😄
RNN-based Model	95.45%	82.14%	***Not feeling good,*** 😞

Figure 3: **Results.** The first model is trained on 400 epochs, while the second one is trained on 50 epochs.

Conclusion

In this study, we proposed two approaches to emojify sentences. We first used Glove 50 word embedder to encode sentences to 50-dimension vectors. By constructing a simple shallow network we could obtain a good accuracy on test data. However, the model is not able to take word dependencies into account in a sentence. Hence, the study suggests using RNNs to address this issue. By doing so, not only does the accuracy of the model improves, but the model is also capable of detecting dependencies between words.

Despite what we did in this study, labels are not orthogonal to each other and considering better labelling methods might improve the functionality of the models. Furthermore, by taking advantage of BiDirectional RNNs and/or attention units, there is the possibility of improving the accuracy index. Providing a higher number of data samples for both training and test sets as well as fine-tuning, can be considered as another contribution that could be done in future research.