

COMP90049 Project 2 Report: Which emoji is missing?

1 Introduction

The project is to analyse the effectiveness of Machine Learning methods on the problem that predicting which emoji should be used in a plain text. Our system is built to output the predictions.

The Machine Learning methods that we are going to analyse are Naive Bayes, Decision Stump (one-level Decision Tree), and TensorFlow Deep Neural Network (DNN). This report is mainly focused on the TensorFlow DNN in detail, whilst other two methods are only used for comparison and critical analysis.

We are building the system in Python and using TensorFlow and some relevant Machine Learning libraries such as Pandas and Numpy. The evaluation of the DNN Classifier will be done accordingly by TensorFlow.

2 Problem

Which emoji is missing? Nowadays, on the Internet, most people are texting and tweeting with emojis. It gives a text an emotional feature and people are able to express their feeling in a text.

We are going to build a system that to train a Machine Learning model and by which to predict an emoji from a plain text in a tweet. This emoji problem is similar to sentiment analysis, but in a discrete way; instead of output a continuous value which represents how positive or negative of a text, we categories a text to a specific emoji class e.g. Happy, Sad etc. In Machine Learning perspective, we defined such problem as a text classification problem.

In such classification problem, text is the only dependency. In order to train our Machine Learning model effectively, we should

carefully define features and do a lot of works on data preprocessing to obtain as much as information we can.

3 Dataset

The dataset contains three set of data: training, development and test sets. Each set of data contains a list of tweets that harvested over several days in April 2018 by sending queries to Twitter API. The training and development data are both contained a column which labels the emoji, whilst the test data is not. There are 10 emojis defined in the class set: Clap, Cry, Disappoint, Explode, FacePalm, Hands, Neutral, Shrug, Think, and Upside.

The training data is used to train our Machine Learning model. After the model is trained, we used the test data as input data and then to output the predictions from our model. The development data is used to evaluate our trained model. In our case of deep learning model, we are using raw data only and will be discussed later.

In our system, all dataset will be converted into TensorFlow-friendly data frame - Pandas DataFrame.

4 System

TensorFlow is an open-source deep learning library which allows to build high performance numerical computation on most types of neural network architecture.

In the system, we are going to use the TensorFlow DNN Classifier with a feature column that generated by a text embedding module called 'nnlm-en-dim128'. The prediction and evaluation are also done in TensorFlow accordingly.

4.1 TensorFlow-Hub

TensorFlow-Hub library provides a feature column that applies a module on the given text feature. 'nnlm-en-dim128' is one of the trained module in TensorFlow-Hub library.

'nnlm-en-dim128' is a token based text embedding trained on English Google News 200B corpus, is based on feed-forward Neural-Net Language Models [1] with two hidden layers. The module takes a batch of sentences as input and maps each sentence into 128-dimensional embedding vectors.

The TensorFlow DNN Classifier is going to initialize with feature columns. In our case, the feature columns only contain one entry which is the sentence. We used TensorFlow-Hub library with module 'nnlm-en-dim128' to generate such feature column for the classifier initialization.

4.2 TensorFlow DNN Classifier

TensorFlow DNN Classifier has a well-constructed neural network and is useful for solving classification problem. The training process is used back propagation method with labelled training data.

A neural network has three layers: input, hidden and output layers. Each layer contains neurons that outputs a continuous value 0 to 1 which represents how likely a corresponding feature exposed in a particular input. The input data will start from the input layer and go through the pre-defined hidden layer and then to arrive the output layer.

In our case, the input layer will be the feature column that generated by TensorFlow-Hub with module 'nnlm-en-dim128' as mentioned earlier. Generally speaking, the feature column is just a neural network compatible representation of the input text. We do not need to manually choose and define features since we are applying a deep learning mechanism which has a well-defined feature representation of input. The output layer will have 10 neurons as we have 10 classes (emojis), and each neuron represents how likely the input should be categorized as that class (should be using that emoji). For each input text, we should pick the neuron that has the highest value in the output layer as our prediction to the input.

5 Evaluation

.....

6 Analysis

.....

7 Conclusion

.....

References

- [1] Bengio, Yoshua; Ducharme, Rjean; Vincent, Pascal; Jauvin, Christian *A Neural Probabilistic Language Model* 2003: Journal of Machine Learning Research, 3:1137-1155.