COMP4332 Project 1 Sentiment Classification

Chong Li Yen, Hang Shing Hei Tommy, Ng Ka Fong, Wong Man Yung

Data Preprocessing and Feature Extraction

There are many features provided for us in the dataset, but we would first need to select features that we want and preprocess our raw data into a form that would be suitable for our deep learning algorithm to use.

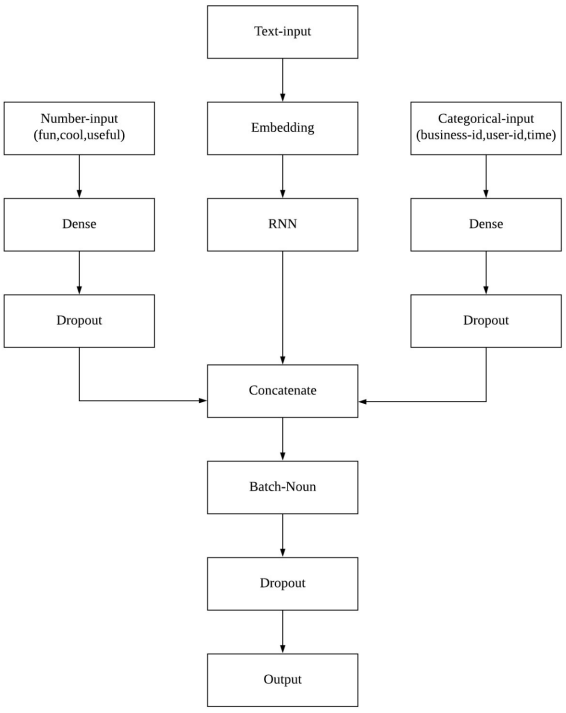
We would consider all features provided into consideration in our deep learning model. Thus, we categorize the features provided into three different categories: Text-Input, Number-Input and Categorical-Input. Data from these three categories would be passed into three different sequential models, and the result of these models would be concatenated and merged into another model to produce an output. Details of our deep learning model would be further explained in the next section.

Text-Input category would be the “text” field in our dataset. For preprocessing of “text” field, we would tokenize the string into words, convert the words into lowercase, remove contraction and adding “not” to indicate negation, filter stopwords and punctuation. We would use the "glove-twitter-50" pretrained model from gensim for creating the embedding weight matrix for an embedding layer. Therefore, we would filter out words that are not in the pretrained model.

The Number-Input category would be the fields “cool”, “funny” and “useful”. Preprocessing would not be needed for these fields as they are already integer values.

Categorical-Input category would be the fields “user_id”, “business_id” and “date”. We would like to consider the frequency of user/business as a parameter. Therefore, we would set users/businesses with less than 5 posts to “Uncommon”, to reduce the effect of uncommon users or businesses. For the “date” field, we would like to consider whether the year, month, weekday and period of the day would affect the user's rating. Thus, we preprocess the “date” field by splitting “date” into four features: “year”, “month”, “weekday” and “time”. “Time” would have four values: 0 for 00:00-06:00, 1 for 06:00-12:00, 2 for 12:00-18:00 and 3 for 18:00-23:59 to indicate a different time period. These data in Categorical-Input would be further processed by passing them into a one hot encoder.

Deep Learning Model



The structure of our deep learning model can be summarized by the flow chart on the left.

Text-Input goes through an embedding layer with the embedding weight matrix obtained in the preprocessing process. The embedding layer is then connected to a Recurrent Neural Network using the layers bidirectional and GRU layers in Keras.

The Number-Input feature first goes through a dense neural network in Keras using the activation function of “Swish”. “Swish” is an activation function $f(x) = x * sigmoid(x)$. Swish activation perform similar to ‘ReLU’ in most cases, with the advantage that its slope $f'(x)$ is continuous at zero. Reports show that this will make the training converge faster and have better performance at deeper networks.

Then, the output of the neural network is passed into the dropout layers to randomly remove 70% of

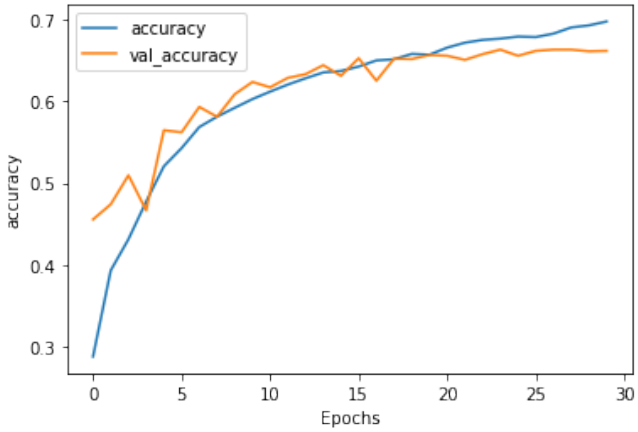
nodes per epoch to prevent overfitting. Apart from its regularization effects, dropouts also force each input to be taken into consideration, hence each node is trained and hopefully provides better performances.

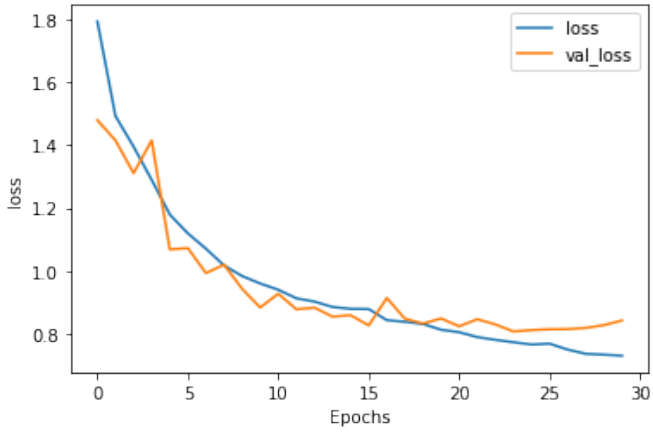
Categorical-Input features goes through similar layers with Number-Input, it goes through a dense neural network with "Swish" as activation function, and passes through a dropout layer but with 75% nodes removed.

The output of these three types of data are concatenated, and pass through a batch normalization layer to adjust and scale the different output from each category. Batch-norm has multiple advantages, including regularization effects, optimizing training speed of the next layer and forcing the layers before or after the batch-norm layer to be trained more independently.

Then, we pass the result into a dense neural network with "Swish", and drop 50% of nodes with a dropout layer. Then, through another dense neural network but using the "softmax" activation function to produce an output of our whole deep learning model.

Results





The validation accuracy starts to level off at epoch 16 and peaks at epoch 24, which is 66.3%. Afterwards, the training accuracy keeps rising while training accuracy oscillates at roughly the same level, loss increases, showing signs of overfitting.

The call-back function is activated at epoch29 and prevents further overfitting. With the help of the Early Stopping function in Keras, the best weight is recovered. Hence, the **resultant weight is restored to epoch24**, providing best validation accuracy and minimal loss. (accuracy: 66.3%, sparse-categorical-cross-entropy: 0.8085)

```
[ngrichard@Ngs-MacBook-Pro Project 1 Sentiment Classification % python3 evaluate.]
py
accuracy: 0.663         precision: 0.5789850829018904   recall: 0.57287515476337
67      f1: 0.5710496159939721
```