Report

**Task 1**

**Method Description: Explain your text cleaning and pre-processing steps, as well as your approach for constructing the distributional semantic representations.**

For the pre-processing steps (function “process\_document”), I have removed all numbers, abbreviations, internal hyphens words, special characters, and punctuation from the reviews, then convert all of them into lowercase. After that, I removed the stop words including “u” and “p”. Finally, I used lemmatization, which only removes affixes if the result is in its dictionary to maintain the meaning of a word. To get the top 50 most frequently occurred words, I merged all the words in the reviews in a list and then do the pre-processing steps (use function “process\_reviews\_str”). For constructing the distributional semantic representations, I have conducted a term-context matrix and used latent semantic indexing (LSI) as the singular value decomposition (SVD) for constructing a low-dimensional dense representation.

**Result Analysis: Analyse and discuss the obtained clustering results.**

Without using SVD and normalization, the feature dimensionality d is 4584, the obtained mean of probabilities after testing 10 times was at 57.6% and the standard deviation was at 0.023323807579381173, then I improved my model by using SVD and normalizing the term-context matrix M, the new feature dimensionality d is 100, then I obtained the mean of probabilities at 93.8% and the standard deviation at 0.02599999999999998. The novelty improvement here is the normalization, which highly improves the performance and training stability of the model, without the normalization, the performance is still around 57.6%, and after normalization, the performance reached 93.8%.

**Task 2**

**Method Description: Explain your classification model design and training.**

I have used the LSTM model, which consists of multiple layers, each taking input from the previous one and advancing output to the next. The first layer takes the numerical sequences as input, and the last layer gives the prediction label as the output. I have defined a sequential () model to embed the layers of LSTM. The layers are as follows:

1. Encode: This is not a layer for the LSTM network but a mandatory step of encoding our words.
2. The Embedding Layer converts our encoded words (integers) into embedding of a specific size.
3. Bidirectional () layer and the LSTM layer with a specified unit size in the LSTM layer: defined by lstm\_out.
4. Fully Connected Layer: maps the output of the LSTM layer to the desired output size.
5. Sigmoid Activation Layer: that turns all output values in a value between 0 and 1
6. Output: Sigmoid output from the last timestep is considered the final output of this network.

For training and testing, I have split the data into 75 % training data and 25 % testing data, I specified the callback for EarlyStopping (), which will halt the model training after the model fails to minimize the validation loss value after the stated no. of epochs in the callback parameters. After that, I fit the model on the training data for a maximum of 70 epochs.

**Experiment and Result Analysis: Describe your experiment and evaluation approach. You should discuss hyperparameter relevant issues if your approach requires any hyperparameter setting. Report and analyze classification accuracy.**

It was observed that the model training was halted after 13 epochs because the validation loss did not improve after the first epoch. We can visualize the training/testing data loss and accuracy, the training loss decreased from 0.679950 to 0.548396, the training accuracy increased from 0.628872 to 0.713618, the testing loss decreased from 0.662402 to 0.626448and the testing accuracy increased from 0.625293 to 0.639344.

文本

描述已自动生成

I have tested the model using 5-fold cross-validation and used model.evaluate () to calculate the score and accuracy. I have achieved a testing accuracy of 64%.

By training the model and modifying some hyperparameters, I have found out that by increasing the density of the fully connected layer and increasing the epochs and batch size, we can train a more accurate model.