1. Personal Information:

First name: Tianyi (Lorena)

Last name: Yan

Email address: [tianyiy@usc.edu](mailto:tianyiy@usc.edu)

1. Classification or regression:

I chose to do a classification, because since we are classifying all tweets based on valence values, which include 0, 2, 4 and are all discrete values, so I chose to treat it as a categorical value and thus do a classification task.

1. Data preprocessing:
   1. Use Pandas to load the csv data source file
   2. Use Keras to tokenize the tweets
   3. Use Keras to pad all tweet sequences
   4. Load pretrained word embeddings and get word embeddings for all words in each tweet. GloVe[[1]](#footnote-1) word embeddings are used in this case.
   5. Perform a simple one hot encoding for the valence target value. (Although there’s only 0s and 4s for valence in the data source file, I still converted 0, 2, and 4 to binary vectors and preprocess all valences as the final labels.)
2. Post preprocessing:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column | Domain | Type |
| Feature | Tweet | array of int (word embeddings) | array of int (word embeddings) |
| Target | valence | array of binary int (one-hot encoding) | array of binary int (one-hot encoding) |

1. Machine Learning model:
   1. Type of model: supervised neural network
   2. Input layer:
      1. Embedding layer with size of length of word\_index from the tokenizer (i.e., numbers of unique words that are tokenized), and with output size of embedding dimension of 50.
      2. Reason: Need to convert word indices to GloVe word embeddings before further processing
   3. Hidden layers:
      1. Add several LSTM layers with size of embedding dimension of 50 (as it’s the output size of the embedding layer) with dropout rate of 0.2 and 0.5 and Tanh as the activation function.
      2. Reason:
         1. Use LSTM as we are processing tweets which involve sequencing and timing issue (sequence of the words).
         2. Use dropout rate to avoid overfitting.
         3. Use Tanh to avoid vanishing gradient problem and also to facilitate converging speed to some extent
   4. Output layer:
      1. Use a Dense layer with size of 3 and SoftMax as the activation function
      2. Reason:
         1. Size of 3: because we have 3 categories for valence: 0, 2, and 4
         2. SoftMax: since we have 3 different categories/classes, SoftMax would be useful to standardize the output.
2. Loss and Optimizer:
   1. Loss: categorical\_crossentropy
   2. Optimizer: RMSprop
3. Split data: I used train\_test\_split from sklearn to shuffle and split all data into training and test data, and then split all training data to train and data for validation. The ratio between train, validate, and test data is 3:1:1.
4. Evaluation Metric:
   1. Accuracy metric
   2. Result:

Text

Description automatically generated

Reference:

* GloVe word embedding: <https://nlp.stanford.edu/projects/glove/>
* Libraries used:
  + Pandas
  + Numpy
  + Tensor flow – Keras
  + Sklearn

1. <https://nlp.stanford.edu/projects/glove/> [↑](#footnote-ref-1)