Selected Topics in Computer Science Assignment

By
Alfred William Jacob 2018A7PS0266P
Arpit Kumar 2018A7PS0272P

1. Introduction

Human Speech can convey emotions, which can be thought of as one of the dimensions of a word embedding or a combination of them, further building on this thought we try to derive sentiment from sentences via word embeddings.

2. Dataset

We have used the dataset provided for this task it contained four files:

- Dictionary.txt- contained phrases and their corresponding ID. These phrases were derived from the Movie Review Dataset
    - Format: Phrase | ID
- Movie Review Dataset.txt- Contained movie reviews and their ID.
    - Format: ID (tab) Review
- Senti_scores.txt- Contained phrase ID and sentiment scores, which ranged from 0 to 1, 0 is negative, of phrases.
    - Format: ID | score
- Train-Dev_test split.txt- Contains Sentence ID and a value 1,2,3 denoting which split it would go to
    - Format: ID, val

The file was used to extract sentiment of a movie review and a csv file was prepared containing **movie review and sentiment value**. The file has been attached with the zipfile.

3. Experiment

To determine the sentiment of movie reviews via word embeddings.

4. Procedure

4.1. Preprocessing

We collected all columns from the text files and created dictionaries to link different columns. Finally forming two tables one for movie reviews and one for phrases.

4.2. Word Vectors

We used Glove word vectors(100-dimensional) and word vectors we constructed via CBOW(400 dimensions). We have not used the word embeddings from phase one to use the same code to produce word embeddings as the vocabulary was very different. We reran the CBOW model for this text dataset to generate new vectors to accommodate the new vocabulary.

4.3 Model

We used 2 models, RNN and LSTM. These two were used due to their property of retaining important past values as input to current value outside the context window. LSTM have an added advantage of preventing vanishing gradient.

RNN - the model uses SGD for gradient descent and all the weights are initialized in the range of (-1/root(n), 1/root(n)), where n is number of nodes in RNN are capped at 32. If a review is more than 32 words, it is cut short to 32, else padded element 0 is inserted to make the length 32. 32 is a hyperparameter that was obtained by taking the 90 percentile for word count of all movie reviews . Gradient clipping was used, to clip the problem of gradient explosion along with regularization. Learning rate was adjusted if loss increases, decreased by a factor of 0.5. The hidden embedding used for 128, again a hyperparameter found using testing.

LSTM - this was implemented using the keras library. The words were again capped at 32, and glove vectors with 100 embeddings were used. Also, Adam optimizer was used for the model. The hidden embedding used for 128, again a hyperparameter found using testing. The following is the summary of the model -

```
Model: "sequential_34"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_34 (Embedding)     (None, 32, 400)           6154400
_________________________________________________________________
lstm_28 (LSTM)               (None, 128)               270848
_________________________________________________________________
dense_28 (Dense)             (None, 5)                 645
=================================================================
Total params: 6,425,893
Trainable params: 271,493
Non-trainable params: 6,154,400
_________________________________________________________________
```

CBOW Model

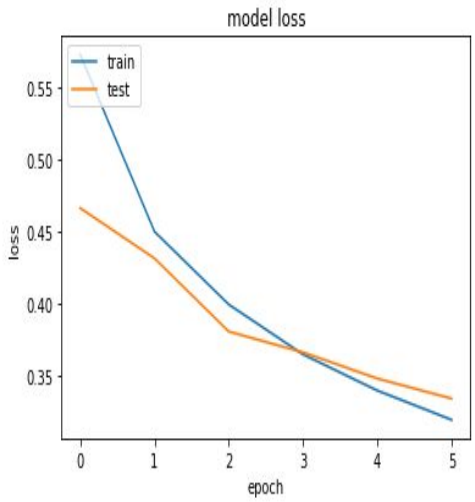
```
Model: "sequential_24"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_24 (Embedding)     (None, 32, 100)           1538600
_________________________________________________________________
lstm_18 (LSTM)               (None, 128)               117248
_________________________________________________________________
dense_18 (Dense)             (None, 5)                 645
=================================================================
Total params: 1,656,493
Trainable params: 117,893
Non-trainable params: 1,538,600
_________________________________________________________________
```

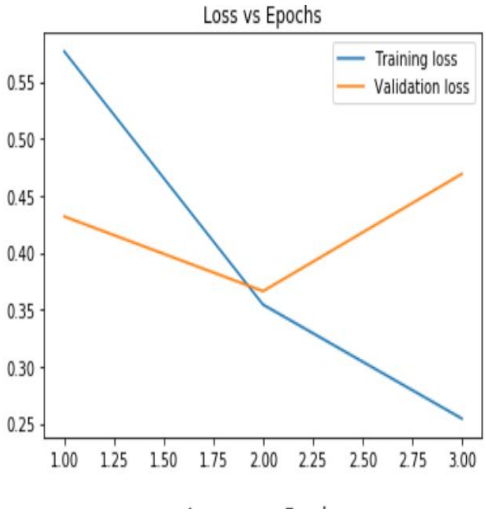
Glove Model

4.3. **Training:** Training was done in batches of 100.

5. **Result -** the model performed better on LSTM using the Glove embeddings.

LSTM - loss (both for GloVe) RNN - loss

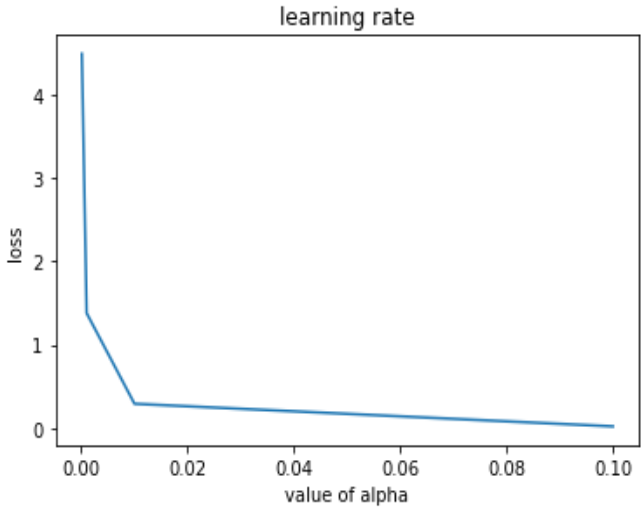


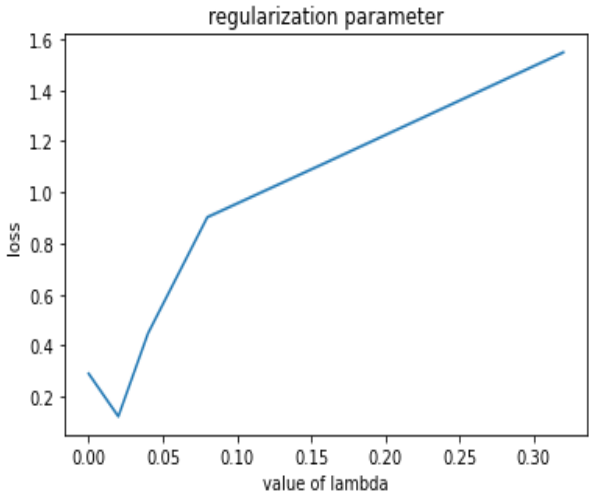


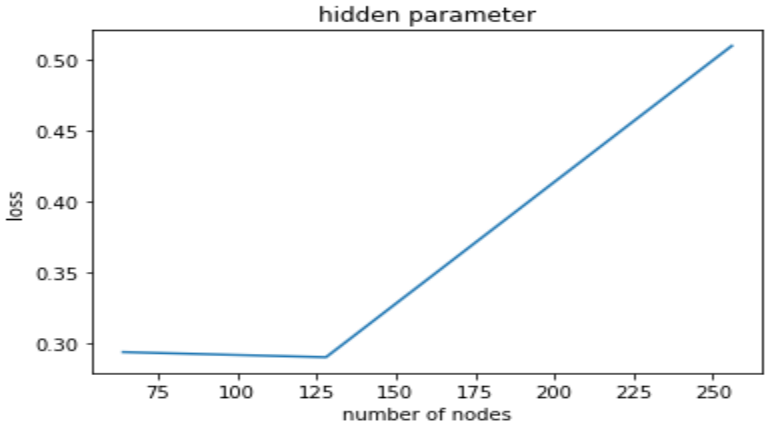
Value of hyperparameter -

	Word Count
count	11856.000000
mean	19.164727
std	9.208945
min	1.000000
50%	18.000000
90%	32.000000
max	56.000000

For most of the reviews(90%), the word count is <= 32. Therefore, the words were capped at a size of 32.







Hyperparameter of RNN	**Optimal Value**
Learning rate	0.1
Regularization Constant	0.02

Number of hidden nodes	128
Size of each input (number of words/review)	32

CBOW vs GloVe - number of epochs = 5, batch size = 100

For RNN
Loss for CBOW - 0.4386944555755576
Loss for GloVe - 0.38405012948739947

For LSTM
Loss for CBOW - 1.6858
Loss for GloVe - 0.2679

6. Conclusion

So, LSTM was found to be better on glove pre-trained embeddings, the accuracy for that came to be around 70% while for RNN, it capped at 62%. The optimal length of word embedding was found to be 32 using statistics while for the hidden dimension of RNN, learning rate, regularization constant and size of input, hyperparameter tuning found the optimal value at 128, 0.1, 0.02 and 32 respectively.