CSI5386: Natural Language Processing

Assignment 2

Text Entailment and Semantic Relatedness

Submitted by:

Shilpu Srivastava [300133366]

Raunak Mahesh [300115000]

The tasks for this assignment were carried out on the SICK dataset. There were three parts to this dataset: the training dataset(comprising 4500 entries) used for training the model, the development dataset (comprising 500 entries) used to tune the parameters to get the best results and the test dataset (comprising 4927 entries) where the textual entailment and semantic relatedness were predicted.

Task 1: Text entailment

For textual entailment, we used Deep Learning to train our model. We used TensorFlow for this purpose. The first step was to tokenize and use the GloVe dictionary to convert the two sentences into equivalent vectors. We use an LSTM to implement our deep learning model and we use dropout on the features from the words and also on the final output. The final output from the LSTMs are passed into a set of fully connected layers, that yields a single real-valued score which denotes the strength of the entailments that accurately gives our classification results of the two statements being NEUTRAL, ENTAILMENT or CONTRADICTION. For each of our training iterations, we calculate the accuracy and the training loss for our model.

Sentence A: Considered as Premise
Sentence B: Considered as Hypothesis

Code Module Name: Textual Entailment.py

Code Module Explained:

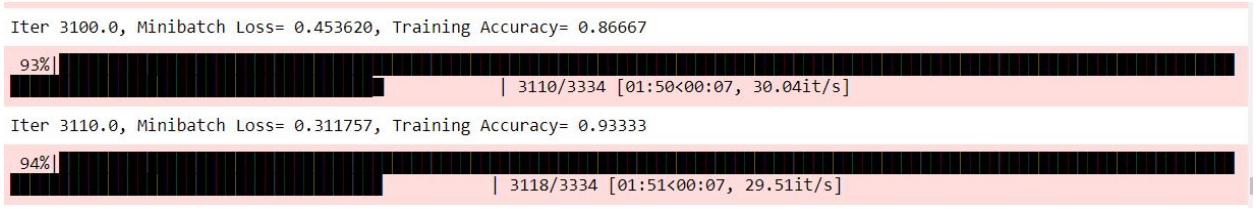
1. Sentences A (Premise) and Sentence B (Hypothesis) are tokenized and converted to vectors. These tokens are combined together by a function sentence_to_sequence that arranges these vectors in order so that they actually represent the sentence.
2. From our training data, we calculate the entailment score in the form of a 1D array with three elements [entailment neutral contradiction]. Based on the entailment labels provided in our training dataset, the array value changes to 1. For example, if the value of entailment is neutral, the array becomes [0 1 0].
3. We define the hyperparameters for our LSTM model as well as the accuracy and the loss functions for training the model.
4. Then we train our model based on the two sentences and their corresponding entailment scores from the arrays we obtained in Step 2.
5. Later we predict the type of entailment for our sentences from this model that we trained.

Parameters and their values that led to the best results:

Maximum Premise Length = 15
Maximum Hypothesis Length = 15
Batch Size = 30
Vector Size = 50
Hidden Size = 100
LSTM Size = 100
Weight Decay = 0.0001
Learning Rate = 0.5
Training Iterations Count = 100000
The probability that inputs to the LSTM will be retained at each dropout iteration = 0.5
The probability that outputs to the LSTM will be retained at each dropout iteration = 0.5

Training:

We achieved the highest training accuracy of 93% and the minibatch loss as low as 31%



Accuracy of Results:

The Prediction Accuracy on the Dev dataset:

```
print('Test accuracy is {}'.format(accuracy_score(y_true,predictions)))
print('Test Precision is {}'.format(f1_score(y_true, predictions,average='weighted')))
print('Test Recall is {}'.format(recall_score(y_true, predictions,average='weighted')))
print('F1 score is {}'.format(f1_score(y_true, predictions,average='weighted')))
print('Classification Report is {}\n'.format(classification_report(y_true, predictions)))
print('Confusion Matrix is {}'.format(confusion_matrix(y_true, predictions)))
```

```
Test accuracy is 0.622
Test Precision is 0.6202450384111766
Test Recall is 0.622
F1 score is 0.6202450384111766
Classification Report is               precision    recall  f1-score   support

CONTRADICTION       0.66      0.53      0.59        74
   ENTAILMENT       0.52      0.51      0.52       144
      NEUTRAL       0.66      0.70      0.68       282

     accuracy                           0.62       500
    macro avg       0.61      0.58      0.60       500
 weighted avg       0.62      0.62      0.62       500


Confusion Matrix is [[ 39   4  31]
 [  0  74  70]
 [ 20  64 198]]
```

The Prediction Accuracy on the Train dataset:

```
print('Test accuracy is {}'.format(accuracy_score(y_true,predictions)))
print('Test Precision is {}'.format(f1_score(y_true, predictions,average='weighted')))
print('Test Recall is {}'.format(recall_score(y_true, predictions,average='weighted')))
print('F1 score is {}'.format(f1_score(y_true, predictions,average='weighted')))
print('Classification Report is {}\n'.format(classification_report(y_true, predictions)))
print('Confusion Matrix is {}'.format(confusion_matrix(y_true, predictions)))
```

```
Test accuracy is 0.7262222222222222
Test Precision is 0.7244559658395192
Test Recall is 0.7262222222222222
F1 score is 0.7244559658395192
Classification Report is               precision    recall  f1-score   support

CONTRADICTION       0.84      0.67      0.74       665
   ENTAILMENT       0.65      0.61      0.63      1299
      NEUTRAL       0.74      0.80      0.77      2536

     accuracy                           0.73      4500
    macro avg       0.74      0.69      0.71      4500
 weighted avg       0.73      0.73      0.72      4500


Confusion Matrix is [[ 444   13  208]
 [   2  787  510]
 [  85  414 2037]]
```

Predictions Made on the Test Dataset:
The entailment predictions based on the test dataset are saved in the file "results.txt"
For each pair id, the entailment predictions are made as shown below.

```
pair_ID Entailment_Judgement Relatedness_score
6 NEUTRAL 2.44
7 ENTAILMENT 2.72
8 NEUTRAL 2.43
10 ENTAILMENT 4.39
11 ENTAILMENT 4.02
13 CONTRADICTION 2.36
15 NEUTRAL 2.44
16 NEUTRAL 2.64
17 NEUTRAL 2.64
19 ENTAILMENT 4.26
20 CONTRADICTION 3.54
21 ENTAILMENT 4.34
22 ENTAILMENT 4.3
23 CONTRADICTION 4.19
```

Task 2: Semantic Relatedness

For semantic relatedness, the task was to determine the degree of relatedness between two sentences. The degree is calculated on the scale on a scale 1-5 where a value close to 5 denotes a very high degree of similarity. For this task also we used a Deep Learning classifier and implemented it using Keras. To calculate our similarity scores, the sentence similarity is evaluated in WordNet. We used the Sequential model from Keras to predict the similarity scores between the two sentences. Since this is not a classification task and rather a regression problem, we use the Mean Absolute Error and Mean Squared Error to determine the loss while training the model.

Sentence A: Considered as Premise
Sentence B: Considered as Hypothesis

Code Module Name: Semantic Relatedness.py

Code Module Explained:

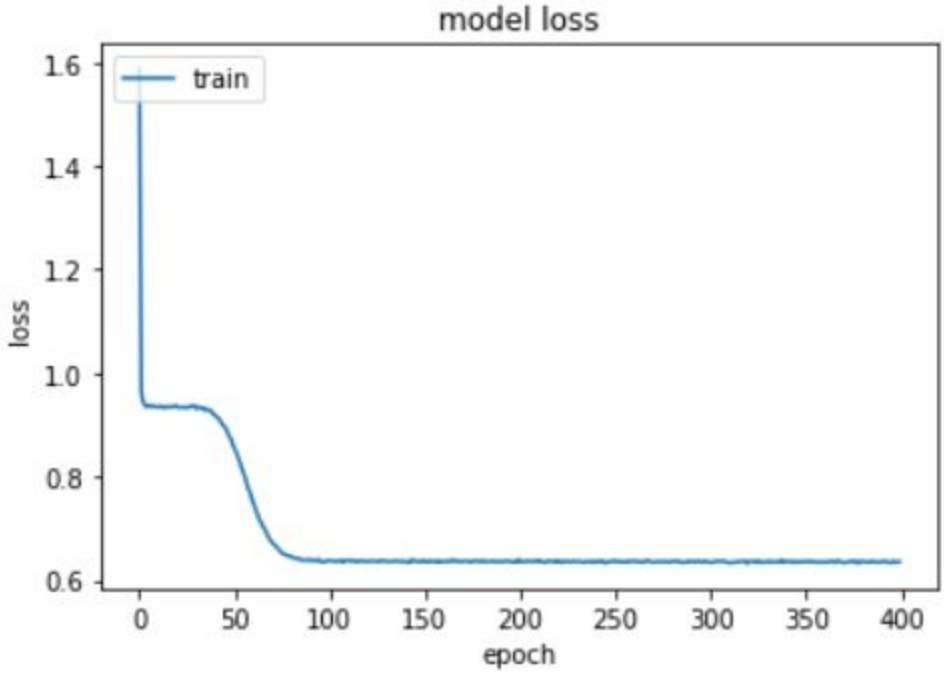
1. Sentences A (Premise) and Sentence B (Hypothesis) are tokenized and converted to vectors. These tokens are combined together by a function sentence_to_sequence that arranges these vectors in order so that they actually represent the sentence.
2. From our training data, we calculate the similarity scores based on the sentence similarity computed using Wordnet.
   ⇒ First, the sentences are tokenized.
   ⇒ Second, we try to look for a contradiction between the premise-hypothesis pair. This is done by looking up for words like “no”, “not”, “none” etc. in the pair of sentences.
   ⇒ Next, the sentences are tagged based on Part of Speech Tagging
   ⇒ Now we get the synsets for these tagged words. Synsets are the grouping of synonymous words that express the same concept. We try to look for these synsets in the other sentence of the premise-hypothesis pair.
   ⇒ Based on all the above calculations, a relatedness score is computed in the file relatedness.py and is returned to the main caller function.
3. After calculating these scores, our Keras Model is provided with these inputs: Sentence_A, Sentence_B, given relatedness scores and the calculated relatedness scores as in step 2.
4. We define the hyperparameters for our Sequential Keras model as well as the accuracy and the loss functions for training the model.
5. Later we predict the semantic relatedness for our premise-hypothesis pair sentences from the test dataset based on this model that we trained.

Parameters and their values that led to the best results:

Model Name: Sequential
Number of Hidden Layers = 60
Activation Function used in the hidden layers = Sigmoid
Activation Function used in the final output layer = Linear
Learning Rate = 0.03
Decay = 1e-6
Momentum = 0.9
Number of Epochs = 400
Batch Size = 10
Loss function = MSE
Training evaluation metrics = MAE, MSE
Optimizer = Adam

Training:

The model loss that was evaluated based on the MAE, MSE showed the below graph when plotted vs the number of epochs.



Accuracy of Results:

The Prediction Accuracy on the Dev dataset:

```
print("*****************************************")
print ("Evaluation Measures")
from scipy import stats
correlation, p_value = stats.pearsonr(y_true2, predict_relatedness2)
print("Pearson Correlation")
print(correlation)
scorr, p = stats.spearmanr(y_true2, predict_relatedness2)
print("Spearman Correlation")
print(scorr)
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(y_true2, predict_relatedness2)
print("Mean Squared Error")
print(mse)
print("*****************************************")
```

```
*****************************************
Evaluation Measures
Pearson Correlation
0.6218503438542828
Spearman Correlation
0.6255549333527446
Mean Squared Error
0.62282805
*****************************************
```

	pair_ID	Calculated Relatedness_score	Given Relatedness Score
0	4	3.25	3.6
1	24	2.53	3.4
2	105	3.51	3.8
3	116	2.53	2.9
4	119	3.88	4.2
...	...	...	...
495	9880	2.82	1.2
496	9931	4.02	3.0
497	9963	2.57	1.0
498	9975	2.13	1.0
499	9988	2.29	1.0

500 rows × 3 columns

The Prediction Accuracy on the Train dataset:

```
print("*****************************************")
print ("Evaluation Measures")
from scipy import stats
correlation, p_value = stats.pearsonr(y_true, predict_relatedness)
print("Pearson Correlation")
print(correlation)
scorr, p = stats.spearmanr(y_true, predict_relatedness)
print("Spearman Correlation")
print(scorr)
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(y_true, predict_relatedness)
print("Mean Squared Error")
print(mse)
print("*****************************************")
```

```
*****************************************
Evaluation Measures
Pearson Correlation
0.6323454184332175
Spearman Correlation
0.6251107784248787
Mean Squared Error
0.6307423166666667
*****************************************
```

	pair_ID	Calculated Relatedness_score	Given Relatedness Score
0	1	4.25	4.5
1	2	3.48	3.2
2	3	3.66	4.7
3	5	3.44	3.4
4	9	3.23	3.7
...	...	...	...
4495	9993	3.44	1.1
4496	9997	2.53	1.0
4497	9998	2.31	1.0
4498	9999	2.15	1.2
4499	10000	2.64	1.0

4500 rows × 3 columns

Predictions Made on the Test Dataset:

```
pair_ID Entailment_Judgement Relatedness_score
6 NEUTRAL 2.44
7 ENTAILMENT 2.72
8 NEUTRAL 2.43
10 ENTAILMENT 4.39
11 ENTAILMENT 4.02
13 CONTRADICTION 2.36
15 NEUTRAL 2.44
16 NEUTRAL 2.64
17 NEUTRAL 2.64
19 ENTAILMENT 4.26
20 CONTRADICTION 3.54
21 ENTAILMENT 4.34
22 ENTAILMENT 4.3
23 CONTRADICTION 4.19
27 NEUTRAL 2.51
29 NEUTRAL 2.92
31 NEUTRAL 3.37
```

Comparison to Baseline Model:

Task Name	Parameters	Baseline	Train Dataset	Dev Dataset
Textual Entailment	Accuracy	54.8%	72.6%	62.2%
Semantic Relatedness	Pearson Correlation	0.618	0.632	0.621
Semantic Relatedness	Spearman Correlation	0.596	0.625	0.625
Semantic Relatedness	Mean Squared Error	10.384	0.631	0.622

Code Modules:

Dataset Description:
File: sick_dataset_train (Used to train the model)
File: sick_dataset_dev (Used to tune the hyperparameters)
File: sick_dataset_test (Used to predict the textual entailment and semantic similarity

Steps to run:
1. Execute file "Textual Entailment.ipynb"
2. Execute file "Semantic Relatedness.ipynb"
3. Final results are obtained in the file "results.txt"

Our Observations:

- For textual entailment, our model's accuracy was quite good around 72% (baseline is 58%)
- For semantic relatedness, our model performed(63% correlation) equally well as the baseline(61%), while the Mean Squared Error was significantly reduced in our model(0.63) while in the baseline it was 10.38.
- The Pearson Correlation definitely has room for improvement and can be achieved by adding some more conditions while computing the similarity using WordNet. Some more hyperparameter tuning and deep learning techniques can help increase the correlation. Additionally, some other semantic relatedness evaluation procedures can also be considered to see if the results get better.
- The dev dataset had quite a few entries(500) when compared to the train (4500) and the test (4927). Hence, the results in terms of accuracy or losses didn't seem adequate. Therefore we evaluated our predicted values both on the train and the dev dataset to ensure good accuracy before predicting the values on the test dataset.

Work Distribution:

	Shilpu Srivastava	**Raunak Mahesh**
Question 1	● Feeding the 1D entailment array to the TensorFlow model and tuning the hyperparameters. ● Predicting on the test dataset and storing the results in file "results.txt"	● Evaluation of the entailment scores based on the entailment label. ● Prediction on the Dev and Train dataset ● Model evaluation for accuracy, precision, recall.
Question 2	● Evaluation of Sentence Similarity scores based on Wordnet. ● Prediction on the Dev and the Train dataset. ● Evaluate the model for the best results (Pearson correlation, Spearman correlation, and MSE).	● Feeding the scores to the Sequential Keras Model and tuning the hyperparameters. ● Predicting on the test dataset and storing the results in file "results.txt"
Documentation	● Documented Steps and Results for Semantic Similarity.	● Documented Steps and Results for Textual Entailment.