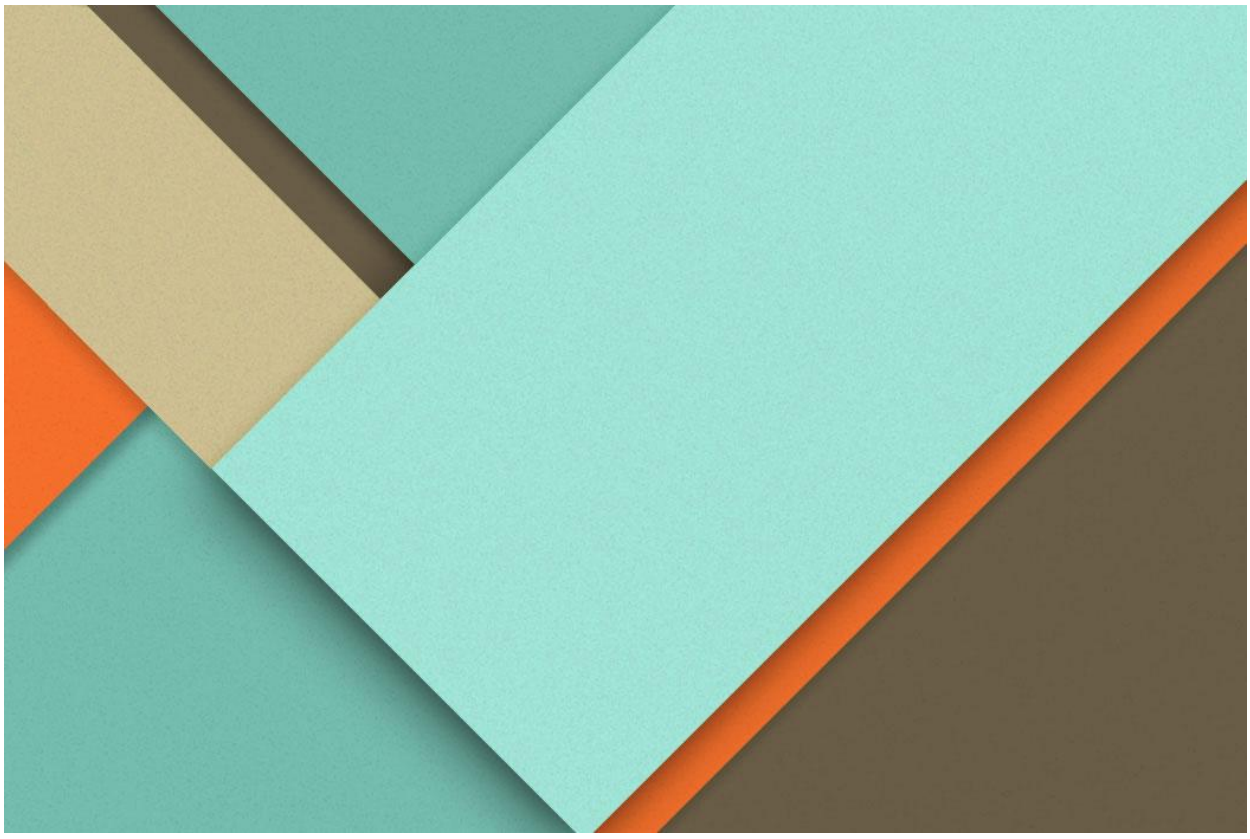


Aspect Based Sentiment Analysis with Gated Convolutional Networks

07.12.2021

Team 47 - Natural Intelligence
Ahana Datta - 2019111007
Pragya Singhal - 2019112001
Sasanka GRS - 2019112017
Tanvi Narsapur - 2019111005

Problem Statement

Sentiment analysis is a Natural Language Processing (**NLP**) approach that aims to determine the emotional tone behind a piece of text, which helps organizations to understand as well as categorize the opinions, reviews and suggestions they receive.

The research paper proposes Aspect Based Sentiment Analysis (**ABSA**) approach which is more efficient and accurate as compared to the general sentiment analysis algorithms. The previous approaches are based on Long short-term memory (**LSTM**) to predict the sentiment. Such mechanisms are complicated and the training phase is time-consuming.

ABSA uses Convolutional Neural Networks and Gating mechanisms which increases its speed as well as accuracy. ABSA can be subdivided into two subtasks, namely Aspect-Category Sentiment Analysis (**ACSA**) and Aspect-Term sentiment analysis (**ATSA**).

ACSA predicts sentiment polarity for a given aspect whereas ATSA determines the sentiment related to target entities appearing in the text (single word or multi-word phrase). Rather than predicting overall sentiment polarity, aspect-based sentiment analysis improves the understanding of the feedback obtained.

Goals and Approach

The Goal of the research paper is to suggest a more efficient and accurate approach for Aspect-Based Sentiment Analysis. The Aspect-Based Sentiment Analysis is considered as a combination of two tasks:

1. Aspect-Category Sentiment Analysis (ACSA)
2. Aspect-Term sentiment analysis (ATSA)

Intuitive Approach

The model is built on convolutional layers and gating units. Each convolutional filter computes n-gram features from the embedding vectors at each position individually. As the gating units on top of the convolutional layers are also independent of each other, the model is more suitable for parallel computing.

The gating units on top of the convolutional layers and the max-pooling layer, both are effective filtering mechanisms that are employed here. Thus, the different layers in the CNN are an embedding layer, a one-dimensional convolutional layer and a max-pooling layer.

- The role of the embedding layer is to take the word indices in the given text and output the corresponding embedding vectors.

- The role of the one-dimensional convolutional filter is to map a set of words in the receptive field to a single feature, along with external bias.
- The features are then passed on to the pair of gates - tanh and ReLU, which constitute the non-linear functions of the neural network. The ReLU Gate receives the given aspect information to control the propagation of sentiment features.
- Element-wise multiplication is performed over the outputs of the GTRU (Gated Tanh-ReLU Unit) and is given to the max-pooling layer above.
- For each of these outputs, the max-pooling layer takes the maximal value to discard less important features.
- Finally, the softmax layer is used to predict the sentiment polarity of the input based on a simple positive, 0 and negative decision rule.

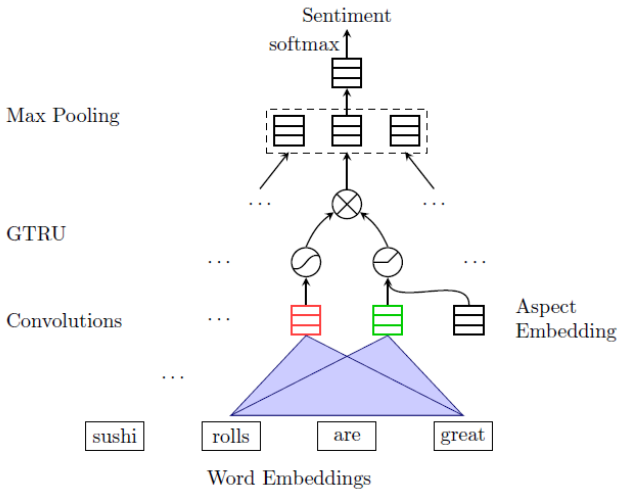
Aspect-Category Sentiment Analysis (ACSA)

The goal of ACSA is to predict sentiment polarity with regard to a given aspect which is one of a few predefined categories.

The model of ACSA comprises two separate convolutional layers on top of the embedding layer, whose outputs are combined by novel gating units which comprise two nonlinear gates, one for each convolutional layer.

GCAE on ACSA

ACSA task uses the Gated Convolutional network with Aspect Embedding (GCAE) model. The figure represents the GCAE model architecture where Gated Tanh-ReLU units with aspect embedding are connected to 2 convolutional neurons at each position t.



The features c_i are computed as follows -

$$a_i = \text{relu}(\mathbf{X}_{i:i+k} * \mathbf{W}_a + \mathbf{V}_a \boldsymbol{v}_a + b_a) \quad (2)$$
$$s_i = \tanh(\mathbf{X}_{i:i+k} * \mathbf{W}_s + b_s) \quad (3)$$
$$c_i = s_i \times a_i \quad , \quad (4)$$

Where v_a represents the embedding vector of a given aspect category in ACSA or the embedding vector of an aspect computed by another CNN over the aspect terms in ATSA. a_i generates the aspect features and s_i is responsible for generating the sentiment features.

The max-pooling layer generates a fixed size vector *e* of dimension d_k where this vector contains the most important sentiment features of the sentence. The layer with the softmax function takes the vector *e* as input and predicts the sentiment polarity denoted by $\widehat{y}$. The training aims to minimize the cross-entropy loss between the ground truth and the predicted value, the equation for the same is given by -

$$\mathcal{L} = -\sum_i \sum_j y_i^j \log \hat{y}_i^j$$

i represents the index of the data sample and *j* is the index of a sentiment class.

Aspect-Target Sentiment Analysis (ATSA)

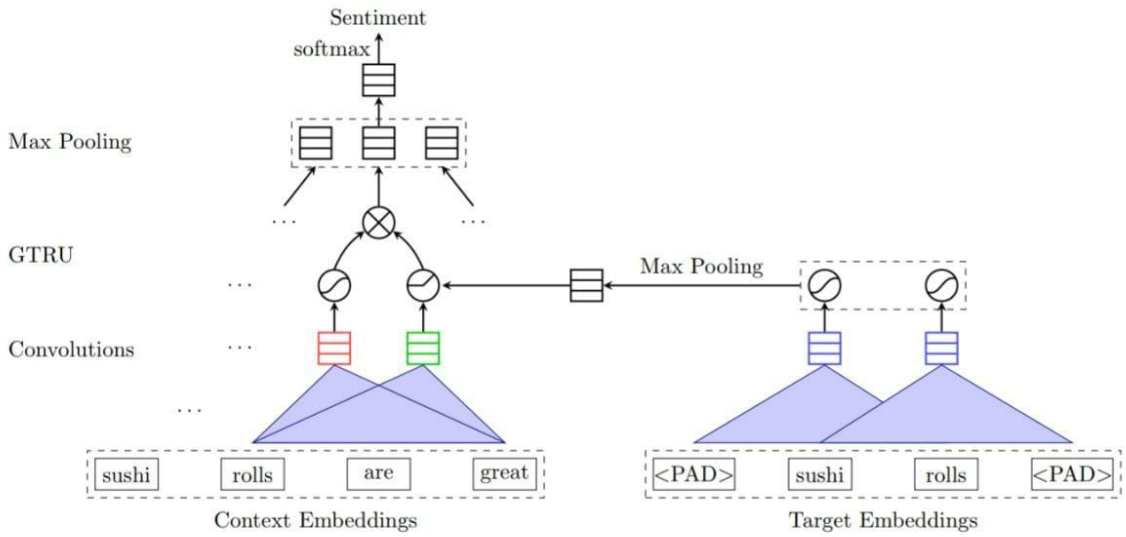
The goal of ATSA is to identify sentiment polarity concerning the target entities that appear in the text instead, which could be a multi-word phrase or a single word.

The model of ATSA comprises another added convolutional layer for the multiple words in the aspect terms.

The algorithm is to utilise the relation or position between the target words and the surrounding context words either using tree structure (dependency) or by counting the number of words between them.

GCAE on ATSA

For the ATSA task, we extend the GCAE architecture used for the ACSA task by adding a small convolutional layer on the aspect terms. The diagrammatic representation for the same is given by the figure below. The aspect information controlling the flow of sentiment features in GTRU is provided by CNN on aspect terms. The CNN added to the ACSA architecture extracts important features from multiple words in the text body.



Experiment

Choice of Sentences

The sentences which have different sentiment labels for different aspects or targets in the sentence are more common in review data than in standard sentiment classification benchmarks. The sentences of interest are similar to:

"Average to good Thai food, but terrible delivery"

This shows the reviewer's different attitude towards multiple aspects, here: food and delivery. Therefore, to assess how the models perform on review sentences more accurately, small but difficult datasets are created, which are made up of the sentences having opposite or different sentiments on different aspects/targets. If a sentence has *n* aspect targets, this sentence would have *n* copies in the data set, each of which is associated with different target and sentiment labels.

ACSA Task and Decision Rule

For the ACSA task, experiments are conducted on restaurant review data of SemEval 2014 Task 4. There are 5 aspects: food, price, service, ambience, and misc to be classified into 4 sentiment polarities: positive, negative, neutral, and conflict. For each sentence, let *p* denote the number of positive labels minus the number of negative labels. The decision rule is to assign a positive label to the sentence if $p > 0$, a negative label if $p < 0$, or a neutral label if $p = 0$. The resulting dataset has 8 aspects: restaurant, food, drinks, ambience, service, price, misc and location.

ATSA Task

For the ATSA task, we use restaurant reviews and laptop reviews from SemEval 2014 Task 4. On each dataset, we duplicate each sentence n_a timmax-poolinges, which is equal to the number of associated aspect categories (ACSA) or aspect terms (ATSA).

The test set is designed to measure whether a model can detect multiple different sentiment polarities in one sentence toward different entities. Without such sentences, a classifier for overall sentiment classification might be good enough for the sentences associated with only one sentiment label.

Training and Testing

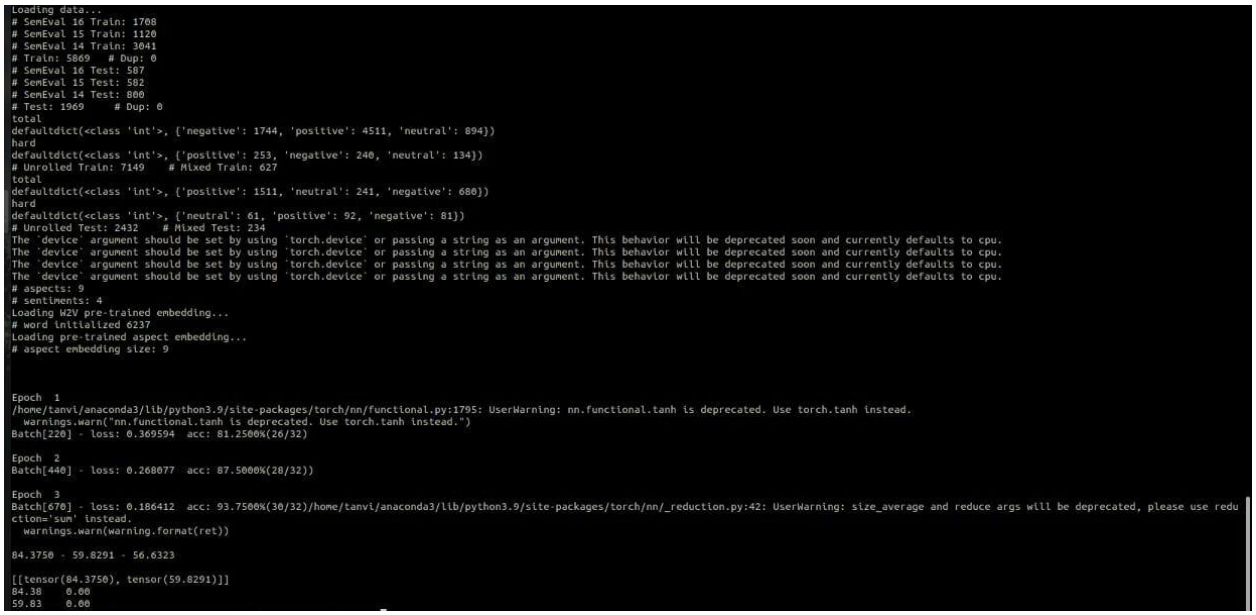
In the above experiments, word embedding vectors are initialized with 300-dimension GloVe vectors which are pre-trained on unlabeled data of 840 billion tokens. Words out of the vocabulary of GloVe are randomly initialized with a uniform distribution $U[-0.25, 0.25]$. Adagrad with a batch size of 32 instances is used, with a default learning rate of 10^{-2}, and maximal epochs of 30. Only early stopping is fine-tuned with 5-fold cross-validation on training datasets. All neural models are implemented in PyTorch/MATLAB.

Dataset

The experiment is performed using a public dataset from SemEval workshops consisting of customer reviews about restaurants and laptops. The dataset is available on the given GitHub repository - https://github.com/Sasanka-GRS/SMAI-Project - in the directory - "DATASETS".

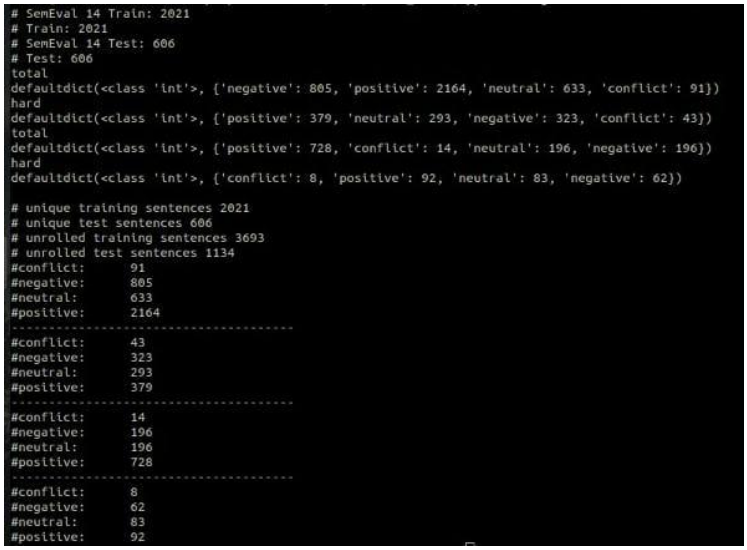
GCAE Model:

Implementing Gated Convolution for 3 epochs and a batch size of 32. The pre-trained embeddings used are - GoogleNews-vectors-negative300.bin.gz. The dataset used for this task is SemEval 14, 15 and 16 Task5.

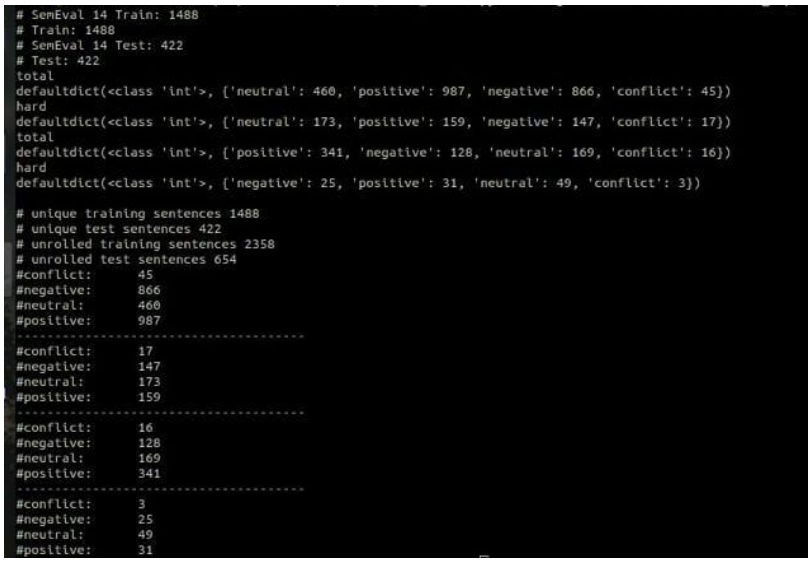


ATSA Model:

Performing ATSA over the restaurant reviews -



Performing ATSA over the laptop reviews -

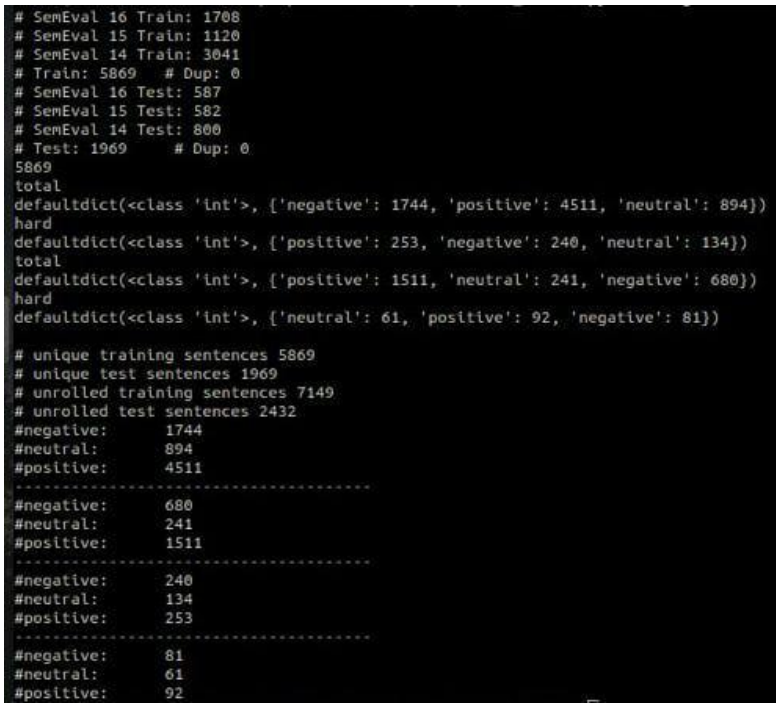


```
# SemEval 14 Train: 1488
# Train: 1488
# SemEval 14 Test: 422
# Test: 422
total
defaultdict(<class 'int'>, {'neutral': 460, 'positive': 987, 'negative': 866, 'conflict': 45})
hard
defaultdict(<class 'int'>, {'neutral': 173, 'positive': 159, 'negative': 147, 'conflict': 17})
total
defaultdict(<class 'int'>, {'positive': 341, 'negative': 128, 'neutral': 169, 'conflict': 16})
hard
defaultdict(<class 'int'>, {'negative': 25, 'positive': 31, 'neutral': 49, 'conflict': 3})

# unique training sentences 1488
# unique test sentences 422
# unrolled training sentences 2358
# unrolled test sentences 654
#conflict:      45
#negative:      866
#neutral:       460
#positive:      987
-------------------------------------
#conflict:      17
#negative:      147
#neutral:       173
#positive:      159
-------------------------------------
#conflict:      16
#negative:      128
#neutral:       169
#positive:      341
-------------------------------------
#conflict:      3
#negative:      25
#neutral:       49
#positive:      31
```

ACSA Model:

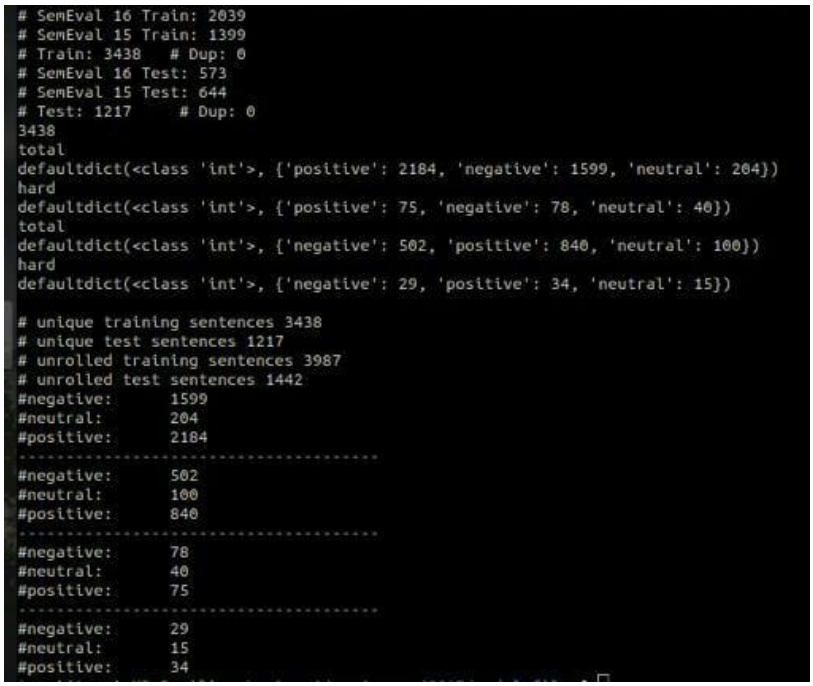
Performing ACSA over the restaurant reviews -



```
# SemEval 16 Train: 1708
# SemEval 15 Train: 1120
# SemEval 14 Train: 3041
# Train: 5869   # Dup: 0
# SemEval 16 Test: 587
# SemEval 15 Test: 582
# SemEval 14 Test: 800
# Test: 1969     # Dup: 0
5869
total
defaultdict(<class 'int'>, {'negative': 1744, 'positive': 4511, 'neutral': 894})
hard
defaultdict(<class 'int'>, {'positive': 253, 'negative': 240, 'neutral': 134})
total
defaultdict(<class 'int'>, {'positive': 1511, 'neutral': 241, 'negative': 680})
hard
defaultdict(<class 'int'>, {'neutral': 61, 'positive': 92, 'negative': 81})

# unique training sentences 5869
# unique test sentences 1969
# unrolled training sentences 7149
# unrolled test sentences 2432
#negative:      1744
#neutral:       894
#positive:      4511
-------------------------------------
#negative:      680
#neutral:       241
#positive:      1511
-------------------------------------
#negative:      240
#neutral:       134
#positive:      253
-------------------------------------
#negative:      81
#neutral:       61
#positive:      92
```

Performing ACSA over the laptop reviews -



Results:

For GCAE, the mean and standard deviation of accuracies over the testing dataset and mixed dataset. Along with this output, a file named time_stamps is created which contains the timestamps of the execution.

For ATSA and ACSA, we obtain 4 sets of output. These predictions correspond to -

1. Simple training dataset
2. Simple testing dataset
3. Hard training dataset
4. Hard testing dataset

The output represents the number of predictions for each of the sentiments namely - negative, neutral and positive. Along with these predictions, the output data also represents statistics about the dataset. The total number of sentences and the number of unique sentences are printed along with the predictions.

LSTM Model:

For comparison with the proposed model, we build an LSTM Model, with parameters:

```
model.add(SpatialDropout1D(0.25))

model.add(LSTM(50, dropout=0.5, recurrent_dropout=0.5))

model.add(Dropout(0.2))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam',
metrics=['accuracy'])
```

We observe that the validation accuracy of this model is **0.9194.**

We observe that the training time of this model is **43sec.**

```
#Training the model
time1 = time.time()
history = model.fit(padded_sequence,sentiment_label[0],validation_split=0.2, epochs=5, batch_size=32)
time2 = time.time()

print("Training time:",time2-time1)
```

```
Epoch 1/5
45/45 [==============================] - 9s 194ms/step - loss: 0.0064 - accuracy: 0.9979 - val_loss: 0.4394 - val_accuracy: 0.9111
Epoch 2/5
45/45 [==============================] - 9s 195ms/step - loss: 0.0061 - accuracy: 0.9986 - val_loss: 0.4392 - val_accuracy: 0.9056
Epoch 3/5
45/45 [==============================] - 9s 194ms/step - loss: 0.0100 - accuracy: 0.9986 - val_loss: 0.4288 - val_accuracy: 0.9111
Epoch 4/5
45/45 [==============================] - 9s 192ms/step - loss: 0.0320 - accuracy: 0.9889 - val_loss: 0.3849 - val_accuracy: 0.8972
Epoch 5/5
45/45 [==============================] - 9s 192ms/step - loss: 0.0073 - accuracy: 0.9993 - val_loss: 0.3782 - val_accuracy: 0.9194
Training time: 43.51257109642029
```

SVM Model:

For comparison with the proposed model, we build an SVM Model, with parameters:

```
SVM = svm.SVC(kernel='linear')
```

We observe that the validation accuracy of this model is **0.9212.**

We observe that the training time of this model is **9.72sec.**

```
# Perform classification with SVM, kernel=linear
SVM = svm.SVC(kernel='linear')
t_start = time.time()
SVM.fit(train_vectors, trainData['Label'])
t_train = time.time()
prediction_linear = SVM.predict(test_vectors)
t_pred = time.time()
train_time = t_train-t_start
pred_time = t_pred-t_train
# results
print("Training time: %fs; Prediction time: %fs" % (train_time, pred_time))

report = classification_report(testData['Label'], prediction_linear, output_dict=True)
print('positive: ', report['pos'])
print('negative: ', report['neg'])
```

```
Training time: 9.810706s; Prediction time: 0.953786s
positive:  {'precision': 0.9191919191919192, 'recall': 0.91, 'f1-score': 0.914572864321
negative:  {'precision': 0.9108910891089109, 'recall': 0.92, 'f1-score': 0.915422885572
```

Comparison of Accuracy with LSTM and STM:

From the obtained accuracies, we can clearly see that the proposed GCAE Model has a higher accuracy as compared to SVM and LSTM, which can be tabulated as below.

Model	GCAE	SVM	LSTM
Accuracy	0.9301	0.9191	0.9194

Comparison of Training time with LSTM and STM:

From the obtained time values, we can clearly see that the proposed GCAE Model has a longer training time as compared to SVM and LSTM, which can be tabulated as below.

Model	GCAE	SVM	LSTM
Accuracy	68.57sec	9.72sec	43.51sec

GitHub Repository link

The link for the GitHub repository is as below:

https://github.com/Sasanka-GRS/SMAI-Project

The above repository contains datasets for training and testing in the DATASETS Folder, the codes for ATSA, ACSA and GCAE along with the comparison codes of SVM and LSTM in the CODES Folder. Along with the codes and datasets, some images of example outputs for all the above mentioned have been included in the RESULTS folder.

Work distribution

Ahana Datta	ATSA, Training
Pragya Singhal	Performance Analysis and Testing
Sasanka GRS	Final Joint Model and Performance Analysis
Tanvi Narsapur	ACSA, Training