Project 3

Task-

To use traditional machine learning approach with Logistic Regression and Deep Learning Approach using RNN/LSTM for the Textual Entailment where a premise and hypothesis are given as input and the prediction has to be made whether the hypothesis is neutral, contradiction, entailment of the premise.

1)Logistic Regression –

The SNLI dataset is divided into the training and test sets with 0.5 million training sentences and 10k sentences for the testing. There are some sentences in the dataset which are not labelled anything and marked with a '-'. They have not been taken into the consideration. Each input consists of two parts the premise and hypothesis and the output consists of one of the three types out of Neutral ,Contradiction, Entailment.

The dataset is first converted into 3 csv files with the same names each for train ,test and dev thereby reducing the memory consumed by the dataset

The preprocessing of the text is done using the standard NLP techniques. First of all each of the sentence is converted to a lowercase. All the punctuation marks were removed from the sentence.

This input sentence is processed using tf idf vectorizer which is used to convert the input sentence to a n dimensional vector. Ngrams of words were used with grams of sizes 1-2 which creates features from the sentences using unigrams and bigrams. Both the premise and hypothesis are independently converted using the tf idf vectorizer and the model is fit using two independent training set separately for the premise and hypothesis.

Logistic Regression was used as a classifier for classifying the input sentence into one of the three categories contradiction, neutral, positive entailment. LBFGS is used as the method for optimization problem. One vs rest Classifier was used to obtain the probabilistic classification of the classifier.

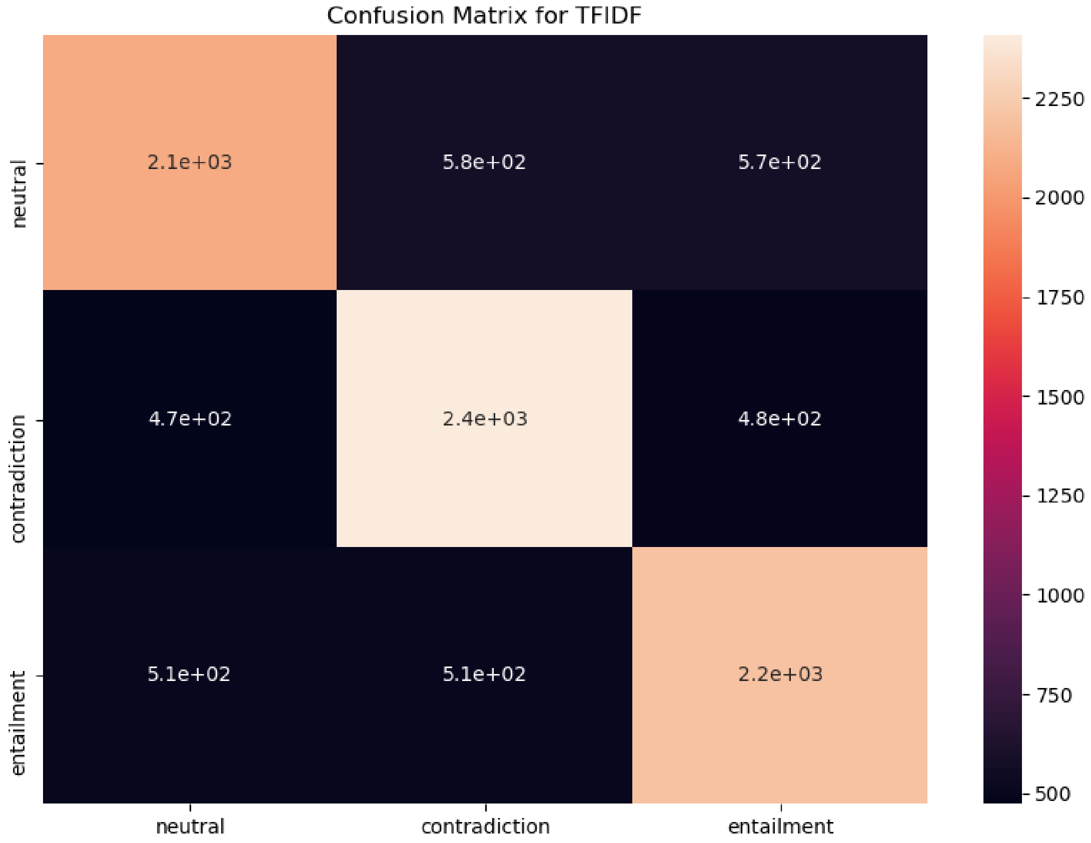
The training data was first converted to the tf idf vectorized form which was fitted using the Logistic regression model. Then test data was converted to the tf idf vectorized form which is predicted using the earlier obtained classifier separately both for the premise and hypothesis.

Observations-

Accuracy of 68% was obtained for this task.

Confusion Matrix –

- On the test set of 9824 the following confusion matrix is plotted. Thus it can be seen that the accuracy is highest for the contradiction sentences i.e 71% and least for the neutral ones being 64%.
- The contradiction class has the highest precision score of 68.81% and thus out of all the sentences classified as contradiction 68.81% are actually contradictory.


Confusion Matrix for TFIDF
neutral
contradiction
entailment
neutral
contradiction
entailment
2.1e+03
5.8e+02
5.7e+02
4.7e+02
2.4e+03
4.8e+02
5.1e+02
5.1e+02
2.2e+03
2250
2000
1750
1500
1250
1000
750
500

2)Deep Network-

Approach for training-

The same csv files that were preprocessed after downloading the dataset are being used. Thus input here is two sentences and output is the label which is suitably encoded into 0,1,2. Thus each sentence before feeding into the neural network model must be converted into a suitable representation which can be fed into the deep network. For pre processing the same, different representations are available. Two representations were tried namely word2vec and glove which generate a suitable embedding in a higher dimensional space.

1)Word2Vec- As the name suggests it converts each word to a vector. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space. A corpus of sentences is used to convert each word from the corpus into a appropriate fixed dimensional representation. Each word having occurrence in corpus as more than 4 times were represented as a vector. As a result 23028 words were selected as the final vocabulary and dimensional size of 300 was fixed for each word. It gives an embedding and ordering based on the occurrences of word In corpus. i.e. the most occurring words are assigned an index of 0 and least occurring are assigned higher index.

e.g In the given corpus. These were the most common eight words ['the', 'in', 'is', 'man', 'on', 'and', 'are', 'of']. These were the least common eight words ['japanes','glowstick', 'spidermen', 'stond', 'constumes', 'graphs','flapjack', 'lamborghini']. Generally the rarer words are more significant and convey more significant information.

2)Glove- It is statistically more significant and is better able to model the relationship differences between the word. Again the corpus is used to generate embeddings for each word and a resulting 300d embedding is generated. Now the words with occurrence of more than 2 are kept. All the numerals or non alphabets are removed from the vocabulary , as a result a vocabulary of 31292 words is generated. The least significant word is the '_unk' token here and the most significant words are again the common words like ['a', 'in', 'is', 'the', 'man', 'on', 'and', 'are']. It gives a better vector like representation for the word embeddings which are analogy preserving under linear arithmetic.

Different types of word processing were done-

- Each of the input sentence was converted to a vector which was basically index of its individual words in the corpus.
- Some common words were removed in generating the list of indices and model was trained both with and without removing common words. The difference was majorly in the speed of the execution with the common words removed representation being faster at training.
- All those words not in the vocabulary were replaced with the unk token which was the least significant word in the vocabulary.
- All the sentences were padded to accommodate for the largest sentence in the corpus so that all the input sentences are of equal length.
- Two different approaches were used one where the premise and hypothesis were concatenated to form a single input and the other one without explicitly concatenating.

The test dataset was divided into two equal halves with one half being the validation set and the other half as the testset. The training dataset was shuffled before each epoch to add the regularising effect. Both the Word2Vec embeddings and Glove embeddings were tried and both showed similar results.

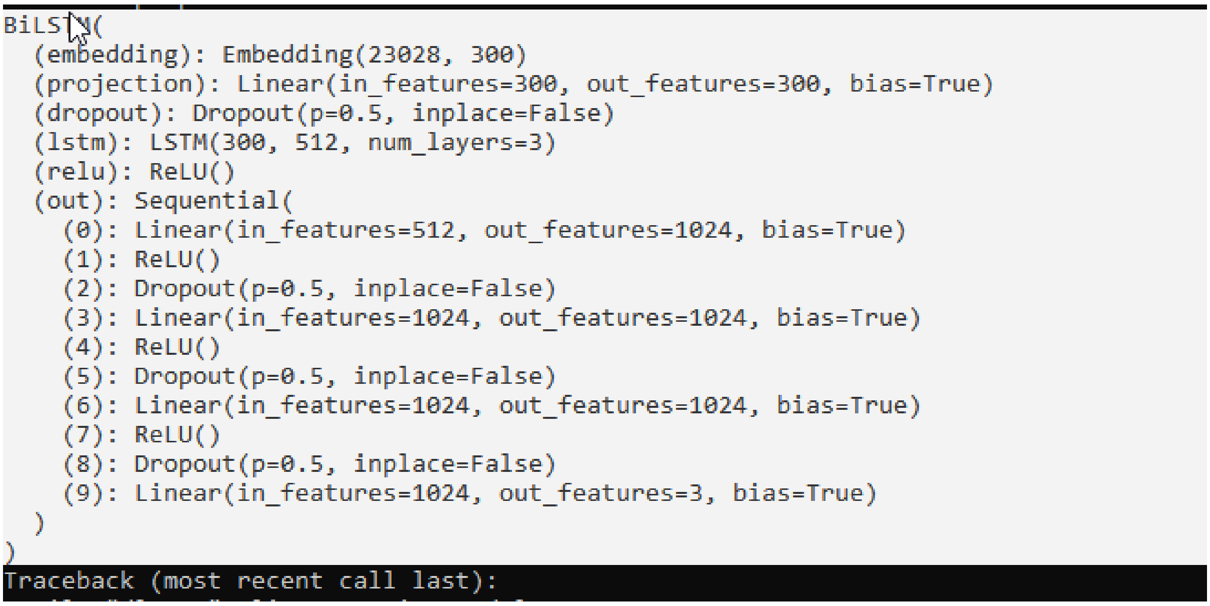
Choosing different hyperparameters of the network-

Network architecture-

Initially made simple 2 layer network of 200D for word embeddings and one fully connected layer of size 400 as final layer. Trained for 20 epochs using momentum with SGD. Got 35% accuracy and loss was also not decreasing much.

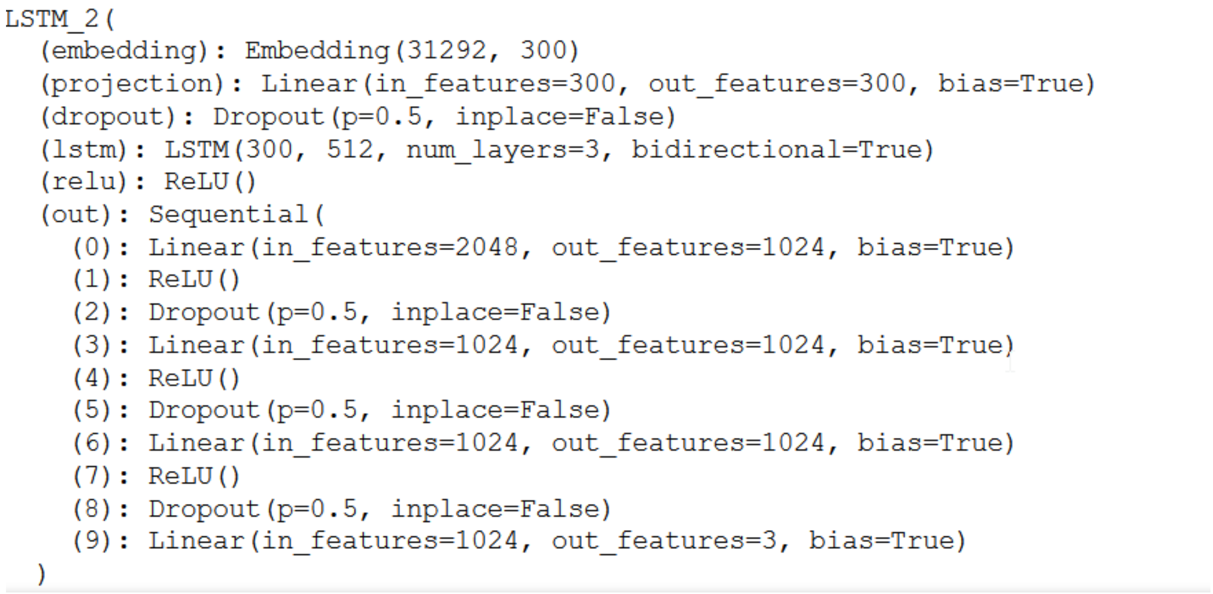
Changed the model used a denser mode of 300D embedding but not pretrained ,did not see any progress.

A denser model was made and an LSTM with the hidden size of 512 was used. The vectors were concatenated to form a single input here. An accuracy of 71% was achieved.



```
BiLSTM(
  (embedding): Embedding(23028, 300)
  (projection): Linear(in_features=300, out_features=300, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (lstm): LSTM(300, 512, num_layers=3)
  (relu): ReLU()
  (out): Sequential(
    (0): Linear(in_features=512, out_features=1024, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=1024, out_features=1024, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=1024, out_features=1024, bias=True)
    (7): ReLU()
    (8): Dropout(p=0.5, inplace=False)
    (9): Linear(in_features=1024, out_features=3, bias=True)
  )
)
Traceback (most recent call last):
```

The model was modified and now the inputs were not concatenated and each layer acted separately on each of the premise and hypothesis which are combined in the end in model before being passed through a final fully connected layer. Thus this was the final model that was used.



```
LSTM_2(
  (embedding): Embedding(31292, 300)
  (projection): Linear(in_features=300, out_features=300, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (lstm): LSTM(300, 512, num_layers=3, bidirectional=True)
  (relu): ReLU()
  (out): Sequential(
    (0): Linear(in_features=2048, out_features=1024, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=1024, out_features=1024, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=1024, out_features=1024, bias=True)
    (7): ReLU()
    (8): Dropout(p=0.5, inplace=False)
    (9): Linear(in_features=1024, out_features=3, bias=True)
  )
```

LSTM was used as the recurrent network to capture long term impacts of the sentences. Both the premise and hypothesis were separately fed into the LSTM unit. Layers were added to fed the previous input to the next layer through stacking of LSTMs together to form a stacked LSTM. The layers were varied from 1 to 3. Both single and bidirectional LSTM was used to model the problem with bidirectional one performing better. The two directional outputs are concatenated thus passing double the outputs to the next layer thus capturing more contextual features.

Masking of inputs were done and sequence length of the individual premise and hypothesis were also passed so as to pad the sequence length and not consider those inputs which were replaced by the unknown tokens during preprocessing and were unpacked after passing through the LSTM layer.

Batch size- It was varied from 32 to 256 scaling by a factor of two each time. The size was varied from 32 to 128 and difference in the training loss and validation loss was recorded which showed not much difference when increased till 128 after which it decreased. So Batch size of 128 was found be to faster and chosen for final model.

Learning rate- It is the most important model of the parameter with slower learning rates not being able to converge and faster learning rates may lead to optima being skipped. It was varied from 10^{-4} in logarithmic scales to 0.1. The best learning rate was 0.0005 and it was chosen for the final model.

Loss function- Cross Entropy loss was used as the loss function for backpropagating the loss. Two variants were tried for the reduction of loss one being the traditional mean one and the other being the sum one. When the loss reduction was kept to sum, loss values were on the higher side as the loss was divided by the number of examples in reduction by mean. So lesser learning rates were able to achieve faster convergence. Both the methods were tried and similar results were obtained in the accuracy with difference in only the loss values and the speed of convergence.

Optimization Algorithm- Adam was used as an optimization algorithm because of its generalized faster convergence rates. Weight decay of 10^{-4} was also added to add the regularizing effect when the model was trying to overfit. But not significant improvement in accuracy was obtained.

Other parameters- Dropout was used with a p value 0.5 for the two layers to introduce regularizing effects in the final fully connected layer.

Findings-

Data	**Loss**	**Accuracy**
Training	**0.004173**	**79.38**
Validation	**0.004464**	**77.30**
Test	**0.004582**	**76.90**

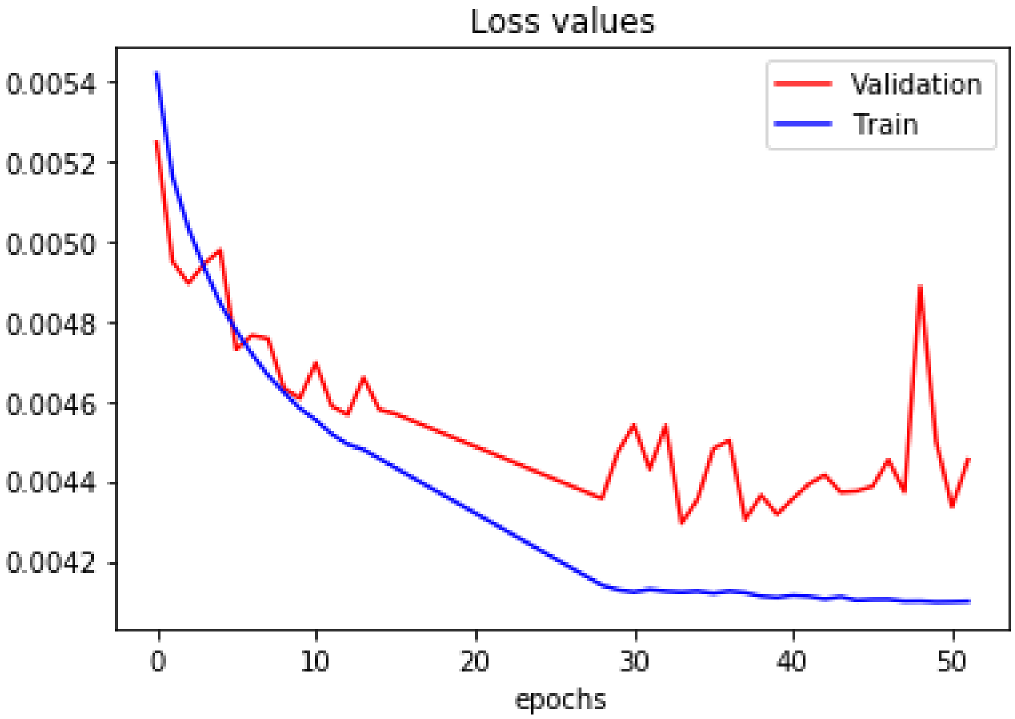
Early stopping was used and 100 epochs were used for training. The above losses are generated using mean reduction of loss and learning rate of 0.0005. Thus it can be seen that all the losses are of similar magnitude and the difference in the training and validation loss is also very less which shows that the model is not overfitting.

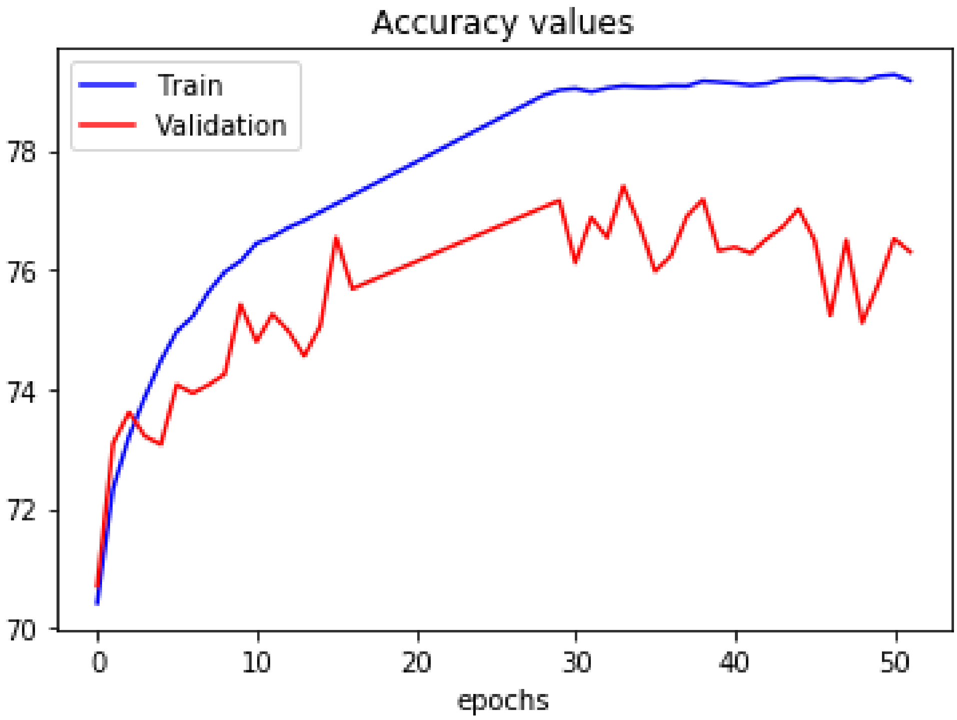
As more epochs were run the training loss started decreasing but validation loss started increasing, thus increasing the generalisation error and in turn reducing the accuracy on the test set.

Observations-

Training and Validation losses versus epochs

As the epochs were increased the difference in the training loss and validation error decreased reaching global minima at epoch 34 and after that it started increasing because training loss was decreasing and validation loss was increasing.



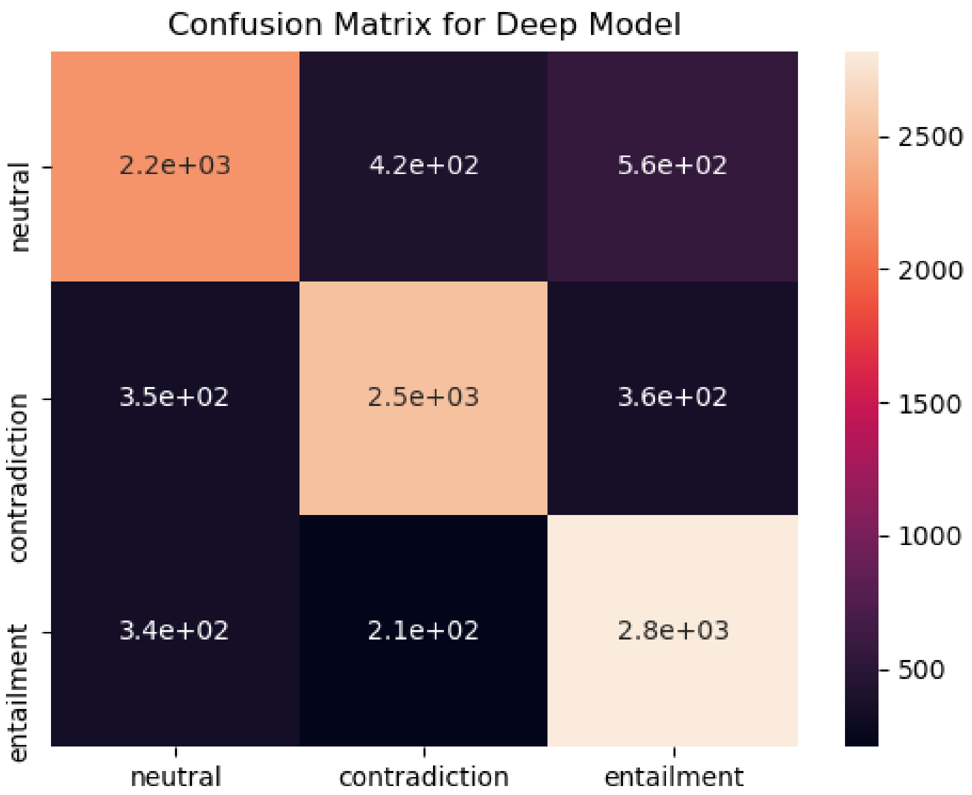


Training and Validation accuracies vs epochs-

Thus both training and validation accuracies increase as the epochs increase. The training accuracy reaches its maximum at epoch 34 after which it decreases and again starts showing zigzag behaviour.

Confusion Matrix-

- On the test set of 4912 sentences the following confusion matrix was plotted. It can be seen that the entailment class has the highest accuracy of 83.5% and is less misclassified as something else whereas the neutral class sentences are often misclassified as the other two with 69% class accuracy.
- However the contradiction class has the highest precision score of 79% and thus out of all the sentences classified as contradiction 79% are actually contradictory.



Other observations-

There were certain Zero length sequences that were formed after preprocessing as they were containing words that were present only once or were not alphabets or were containing common words only. Some were having spelling issues also. Some of these sentences present in the training dataset are –

```
The pers
They are in the Phillipines .
It is 7:00pm .
Snidely Whiplash is a philanthropist .
They are altheltes
Feltch .
A ma
He is a teetotaler .
She is a seemstress
Wolverine does the watusi .
There is a mountan .
https://www.youtube.com/watch?v=tXrsvC25GH8
```

These were not helpful for the training and they were explicitly padded to have seqeunce of length 1 containing only the unknown data.

There were two sentences which contain all the unknown words in the testset and they were not useful.

```
There is a sculture
Thi
```

Also those sentences beginning with an unknown token were also explicitly padded as a sequence of length 1.