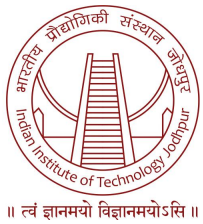


Natural Language Understanding - CSL7340

Predictive Model To Predict Ratings

Ayanabha Ghosh (M21CS055)
Puja Gupta (M21MA004)
Shaonli Pal (M21MA007)

Module - 2 (Predictive Model to predict Ratings):

Tasks: To predict the rating using different predictive models.

Dataset used:

This dataset contains review texts along with numeric ratings given by the user. It doesn't have aspects marked. It consists of a total of 82,066 reviews.

1. Use the best model (from 1(a) and 1(b)), to determine the aspects and their sentiments from each review.

Link of Google Colab Code File:

https://colab.research.google.com/drive/1AtAyrPF3PBv8QRVjxMZrlFAz2h2A9VtZ#scrollTo=jgjOWujhIccR

We have stored the best model which was giving the highest accuracy as a bin file in our drive and used that model on entire dataset to determine the aspects and their polarity (sentiment) for each review and used this new dataset with aspects determined as the new feature columns to train our model using different machine learning classifiers to predict the ratings for each of the reviews.

2. Develop a predictive model to predict the ratings based on aspects and sentiments.

Steps Involved:

2.1. Converted the CSV into a dataframe with columns ***text*** containing the reviews given by the customers and ***Ratings*** assigned by them.

2.2. Since the dataset is quite large, we have used the first 10,000 reviews for our analysis.

2.3. Removed HTML tags containing html and css markup from the text column.

2.4. Removed the punctuations from the text column and also lemmatized its contents.

2.5. We have used the best model stored in Module 1 to determine the aspects of each review text and their polarity (sentiment) based on the aspect score.

	id	text	aspect	aspect_score	polarity
0	1	Thank you thank you thank you !! I want to th...	food	0.896121	0
1	1	Thank you thank you thank you !! I want to th...	staff	0.333275	0
2	1	Thank you thank you thank you !! I want to th...	miscellaneous	0.557642	0
3	1	Thank you thank you thank you !! I want to th...	place	0.343386	2
4	1	Thank you thank you thank you !! I want to th...	service	0.383351	2
5	1	Thank you thank you thank you !! I want to th...	menu	0.227896	2
6	1	Thank you thank you thank you !! I want to th...	ambience	0.436657	2
7	1	Thank you thank you thank you !! I want to th...	price	0.167834	2

2.4. Created 8 new feature columns based on the Aspect Category used in Module 1 i.e. *"food", "staff", "miscellaneous", "place","service","menu","ambience","price"* and their polarity (sentiment) is assigned as label value for each text (Positive: 0, Negative: 1 & Neutral: 2). In case any aspect is missing from any text, its polarity is given as neutral(2) for all the corresponding aspect categories for the text.

	text	food	staff	miscellaneous	place	service	menu	ambience	price	Ratings
0	Thank you thank you thank you !! I want to th...	0	0	0	2	2	2	2	2	4
1	A Humane Society store at the Biltmore? Inter...	2	0	2	0	2	2	0	1	5
2	Don't buy Nike sneakers if you want to return ...	2	1	2	2	2	2	2	1	1
3	I have to say I love most things about Sprouts...	0	1	0	0	1	2	2	1	3
4	The tire pressure light came on a day or so ag...	2	1	2	2	2	1	2	1	5

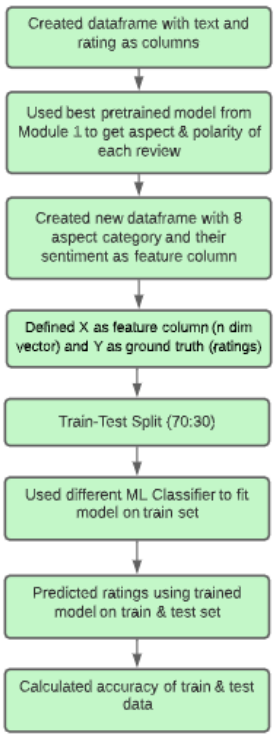
Our new dataset looks like the above figure which is further used to predict the ratings using different ML Classifiers.

2.5. The feature columns X used to train the model are (*"food", "staff", "miscellaneous", "place","service","menu","ambience","price"),* with ground truth stored in Y(Ratings). Thus creating an n-dimensional vector of aspects for each review and these vectors are used as features to predict ratings.

2.6. Divided our dataset into train and test split in the ratio 70:30.

2.7. Used the inbuilt libraries to train the model with different machine learning classifiers to predict the ratings of the reviews.

3. Discuss the problem setup you have followed along with the evaluation.



Observation:

1. Classifier Used: SVM	
Train Accuracy	**Test Accuracy**
41.82%	42.17%
2. Classifier Used: Decision Tree	
Train Accuracy	**Test Accuracy**
47.1%	40.76%
3. Classifier Used: Random Forest	
Train Accuracy (%)	**Test Accuracy (%)**
46.78%	42.06%

Inference:

Here we can observe that the classifier which is giving highest accuracy both on the Train & Test set is Random Forest with Train Accuracy of 46.78% and Test Accuracy of 42.06%.

4. Develop a second model for predicting ratings directly using the text. You can use LSTM, BERT or any other model for the purpose.

For this part we have used the ***"bert-base-uncased model"*** for the rating prediction.

4.1. Link of Google Colab Code File:

https://colab.research.google.com/drive/1Wf-to8Z6Fqxk5hwX_2dDTpylh0OLzGAn#scrollTo=xsFB0Qt_n5D9

4.2. Data Preprocessing:

1. Converted the CSV into a dataframe with columns ***text*** containing the reviews given by the customers and ***Ratings*** assigned by them.
2. Since the dataset is quite large, we have used the first 10,000 reviews for our analysis.
3. We have assigned polarity (sentiment) for each review based on their ratings.

Review Ratings	Polarity(Sentiment) assigned
4 and 5	Positive
3	Neutral
1 and 2	Negative

	text	Ratings	polarity
	Thank you thank you thank you !! I want to th...	4	positive
	A Humane Society store at the Biltmore? Inter...	5	positive
	Don't buy Nike sneakers if you want to return ...	1	negative
	I have to say I love most things about Sprouts...	3	neutral
	The tire pressure light came on a day or so ag...	5	positive

4. Checked the polarity distribution for each polarity i.e. "Positive, Neutral & Negative". We observed that there is a large non uniform distribution of the data within the different polarities.
5. Performed undersampling to balance uneven dataset by keeping all the data in the minority class (neutral & negative) and decreasing the size of the majority class (positive).

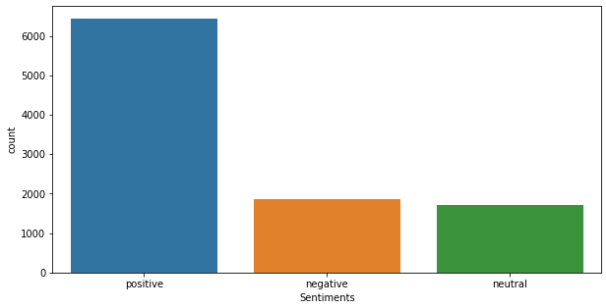


Fig: Distribution of data before sampling

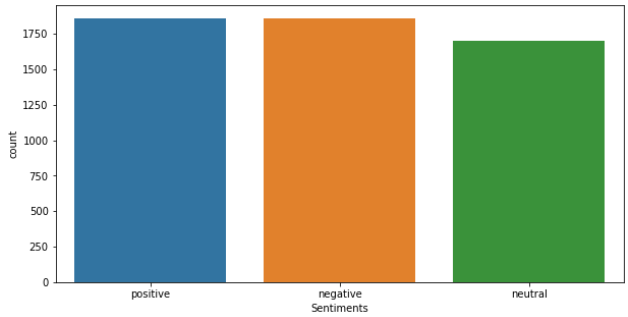


Fig: Distribution of data after undersampling

6. Performed Label Encoding on Polarity i.e.(Positive: 0, Negative: 1 & Neutral: 2) and assigned its value in a new column "*polarity_new*".
7. Removed HTML tags containing html and css markup from the text column.
8. Removed the punctuations from the text column and also lemmatized its contents.

	id	text	Ratings	polarity	polarity_new
0	7901	Every location of InnOut Burger Ive ever visit...	5	positive	0
1	2424	Its been a very long time since I visited Clai...	2	negative	1
2	2443	I cant believe this place is not packed on a S...	4	positive	0
3	21297	We came by for lunch on a Saturday which turne...	3	neutral	2
4	10480	I had heard about the growth of this chain and...	2	negative	1

" Point 5 of problem statement: You can choose custom train-val-test splits from the datasets. (Note - This is a big dataset and you may not be able to use all of it. Choose judiciously.)

9. Splitted the data into three parts i.e. train, validation and test set in the ratio (80:10:10)

4.3. Hyperparameters used to train the model, epochs, learning rate, hidden representation size etc.

1. Imported the pre-trained BERT model and added

    1.1. Dropout layer with probability 0.3

    1.2. One extra linear layer with input features as 768 and output features as 3 (since we have three sentiment classes in our data, i.e. positive, negative & neutral). We have used the ***'bert-base-uncased'*** architecture of BERT for our analysis.

```
      )
      (pooler): BertPooler(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (activation): Tanh()
      )
    )
    (drop): Dropout(p=0.3, inplace=False)
    (out): Linear(in_features=768, out_features=3, bias=True)
  )
```

2. We have used BertTokeniser to tokenize our input train & validation data to the model.
3. Trained the Bert Model on train data with following hyperparameters:

```
TRAIN_MAX_LEN = 140
VALID_MAX_LEN = 140
TRAIN_BATCH_SIZE = 16
VALID_BATCH_SIZE = 16
EPOCHS = 10
BERT_MODEL = 'bert-base-uncased'
LEARNING_RATE = 3e-5
```

Loss Function used	Optimizer used
CrossEntropy Loss	Adam Optimizer

If review text length is too big then we will clip it to 140 characters. The Batch size used to train the model was taken as 16 and learning rate of 3e-5.

4. We have trained our model for 5 epochs and stored the one with best accuracy as the bin file which is used for predicting rating of the test data.

4.4. Observation:

Epoch	Validation Accuracy (%)
1	64.37
2	66.93
3	67.82
4	69.56
5	70.74

" Point 6 of Problem statement: Report the training-validation loss/metric plots."

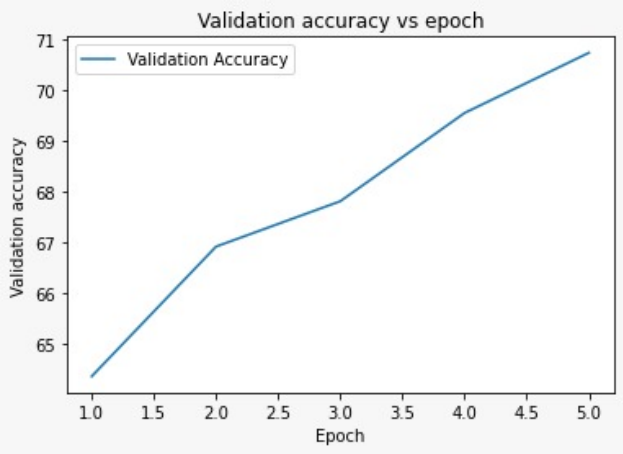


Fig: Validation Accuracy Plot

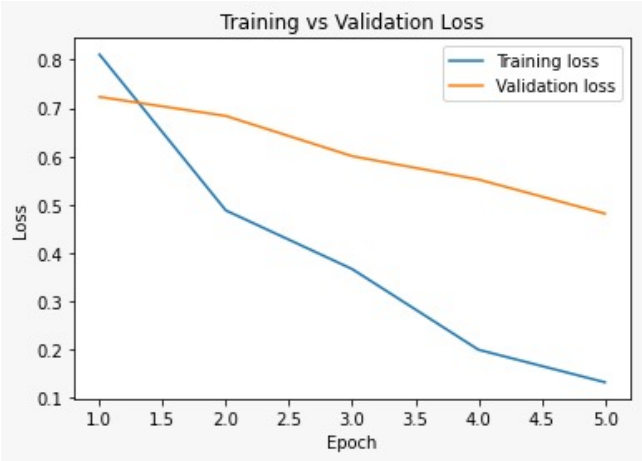


Fig: Training- Validation Loss Plot

4.5. Inference:

We have determined the ratings for each review text based on the following logic:

If predicted polarity = 0 and polarity score >= 0.75:
assign rating = 5 (positive)

elseif predicted polarity = 0 and polarity score >= 0.4 and polarity score <0.75:
assign rating = 4 (positive)

elseif predicted polarity = 2:
assign rating = 3 (Neutral)

elseif predicted polarity = 0 and polarity score >= 0.4 and polarity score <0.75:
assign rating = 1 (negative)

elseif predicted polarity = 1 and polarity score >= 0.75:
assign rating = 2 (negative)

7. Compare the performance of models developed in steps 2 and 4.

7.1. Performance using models developed in Step 2:

Step 2 uses different ML Classifiers to predict the rating of the review text based on the aspect sentiment assigned to each of the reviews using a pretrained model with best accuracy developed in Module 1.

1. Classifier Used: SVM	
Train Accuracy	**Test Accuracy**
41.82%	42.17%
2. Classifier Used: Decision Tree	
Train Accuracy	**Test Accuracy**
47.1%	40.76%
3. Classifier Used: Random Forest	
Train Accuracy (%)	**Test Accuracy (%)**
46.78%	42.06%

Here Random Forest is giving the best accuracy with train accuracy of 46.78% and test accuracy of 42.06%.

7.2. Performance using models developed in Step 4:

Step 4 uses ***"bert-base-uncased model"*** to predict the ratings for the review text. The best validation accuracy obtained using this model is ***70.74%.***

Out of these two models developed in Step 2 and 4, we can clearly see that the model developed in Step 4 ,i.e. ***"bert-base-uncased model"*** is giving the highest accuracy of ***70.74%.***