Task Report

Aim: Multilabel Document Categorization from Scratch

Sub task - 1: Unsupervised topic modelling:

The main aim of this task was to perform unsupervised learning to find topic clusters. As per my experience, topic modelling could be performed by various different dimension reduction techniques, the ones which I implemented while performing initial analysis were:

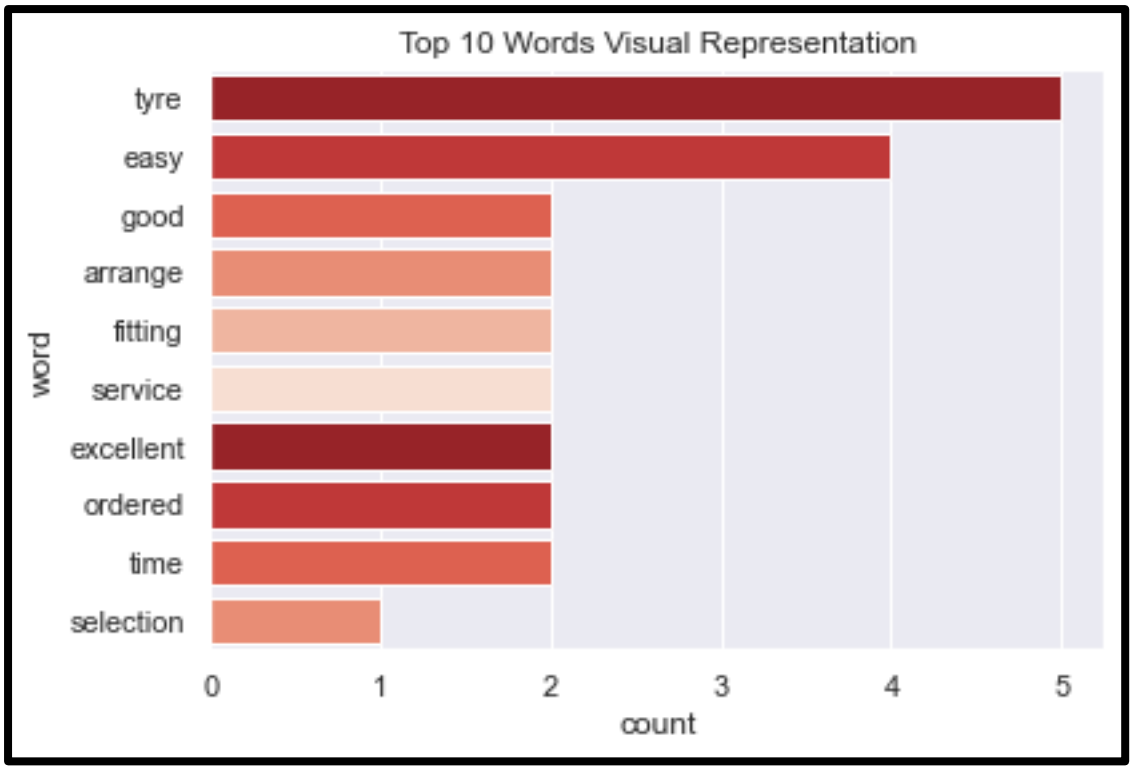
- NMF – Non negative Matrix Factorization
- SVD -Singular Value Decomposition
- LDA - Latent Dirichlet Allocation

A quick overview while working with the above algorithms – LDA accepts raw count values which are binary counts of tokenized words obtained, like 1 or 0, since it is a probabilistic algorithm. Whereas NMF and SVD Accepts Float integer values, therefore, I implemented TFIDF Scores while using them.

Initial Text Analysis :

Word Cloud For Highlights :

Top 10 Words After further cleaning of text :



The given dataset had no null values.

For Pre-Processing, I developed a custom function to perform normal text pre-processing like removal of stop words, punctuations and unwanted symbols, which has the flexibility to set up further pre-processing within built parameters steps like extra stop words removal or stemming and lemmatizing and so on.

Count Vectorizer and TFIDF Transformer's parameters were set with relevant parameters to get bag of words for LDA and TF-IDF Scores for SVD and NMF respectively.

After Running Each Model, I found out LDA performed much better from transformed vector scores and topic index, while clustering topics. Therefore, I performed a Grid Search CV wit LDA model to find the better set of hyperparameters.

```
"""
Fitting 3 folds for each of 12 candidates, totalling 36 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worker
.
[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed: 10.7min finished
Best model's params:  {'learning_decay': 0.75, 'learning_offset': 30, 'n_compo
nents': 12}
Best log likelihood score:  -83160.21471701778
Model perplexity:  55.26047238754284
"""
```

After obtaining the best set, I ran the LDA model with tuned hyper parameters and used that as final model for later analysis.

Later, I performed the joining of the individual clusters up to n_top_words length, so that I can obtain 'n' topic categories.

After obtaining the topic numbers, I used the transformed probabilistic scores for each topic distribution and merged them into data frame along with text from raw dataset

NOTE: Topic Labels were assigned to each topic numbers and were mapped accordingly after analysing them with relevant clusters.

After getting the inference, the entire processed data was stored in a separate data frame and exported into datasets folder -> **"pre_processed_df.csv"**

A custom function was designed to implement inference for raw text to find out the probable topic cluster under which the text falls, from unsupervised model.

```
In [151]: sample_text="A perfectly easy way to order tyres online. Just enter your car registration number, check the recommended tyres are
          sample_text2="Delivered what I ordered and had in stock, excellent fitting service, price was decent and on time."

          topic_names=['value for money','garage service', 'mobile fitter',  'length of fitting', 'booking confusion', 'delivery punctualit

In [152]: len(topic_names)
Out[152]: 12

In [156]: get_inference(lda,tf_vectorizer,topic_names,sample_text,0.1)
Out[156]: ('ease of booking',
           array([[0.18937239, 0.00520834, 0.07334006, 0.08805057, 0.1835787 ,
                   0.00520859, 0.00520848, 0.17539373, 0.00520856, 0.00520879,
                   0.07151355, 0.19270824]]),
           {'booking confusion', 'ease of booking', 'location', 'value for money'},
           [11, 0, 4],
           ('ease of booking', 'value for money', 'booking confusion'))

In [157]: get_inference(lda,tf_vectorizer,topic_names,sample_text2,0)
Out[157]: ('garage service',
           array([[0.01041693, 0.36510497, 0.0104167 , 0.01041677, 0.01041683,
                   0.30556582, 0.01041677, 0.01041703, 0.01041695, 0.23557782,
                   0.01041669, 0.01041674]]),
           {'booking confusion',
            'change of date',
            'delivery punctuality',
            'discounts',
            'ease of booking',
            'garage service',
            'length of fitting',
            'location',
            'mobile fitter',
            'tyre quality',
            'value for money',
            'wait time'},
           [1, 5, 9],
           ('garage service', 'delivery punctuality', 'tyre quality'))
```

Sub Task - 2: Learning a supervised multi-Topic Classifier

The main aim of this task was to build a supervised learning classifier, which could take inputs as text and could predict a suitable category, from a list of topic categories obtained in subtask 1 of unsupervised learning.

The data frame which was obtained and exported from subtask1, was loaded and used for building a supervised learning classifier.

I have implemented a deep, ensembled neural network architecture for performing multi label classification for this purpose.

I have shown two implementations:

- Deep Neural Network consisting of Bidirectional LSTMs and ConvNet 1-D along with TensorFlow native embeddings
- Deep Neural Network consisting of Bidirectional LSTMs and ConvNet 1-D along with glove 300-D embeddings

Pre-processing:

The exported data frame -> **"pre_processed_df.csv"** was loaded and again pre processed for further cleaning and converting it into lowercase.

Data Preparing:

After cleaning of the text data, I performed a train test split of 80: 20 ratios, where 80% is for training data and 20 % stands for testing data.

After obtaining train and test data, test data was further subjected to reshaping, for model input.

A custom function was developed in order to find the maximum longest string the entire text corpus, so that I can get the max_length of a string for model input and while post padding other sequences.

Some common parameters were given an initial fixed value in support for our model training.

The maximum number of words to be used - >most frequent): vocab_size = 50000

Dimension of the dense embedding.: embedding_dim = 256

Max number of words in each complaint. max_length = 310 [obtained from custom function]

Truncate and padding options: trunc_type = 'post' padding_type = 'post' oov_tok = '<OOV>'

NOTE: parameter 'max_length' totally depends on pre processing of the text. If the pre processing function is further modified with additional parameters like enabling stemming, extra stop words removal, etc, it might change!

Tokenizer function was used to tokenize the cleaned text -> X_train corpus and obtain the word index with word vocabulary.

X_train and y_train were converted to sequences using text to sequences in order to convert it into numeric format as per the word index obtained.

For target classes, One Hot Encoder was used to convert them into categorical binary array consisting of all unique classes. This processing step gave training and testing labels in numeric format.

A custom function was developed for plotting the model performance with training and validation accuracy and loss, subjected to post training.

My Approach with Model Training:

Model Architecture used for building multi label classifier is as follows:

Deep Neural Network consisting of Bidirectional LSTMs and ConvNet 1-D along with embeddings.

Sequential Model API was used to stack individual network layers like LSTM, ConvNet1-D

Since this network consisted of LSTMs, there was a pretty good chance of overfitting of data. Therefore, I used regularizer -> Dropout layer.

ConvNet 1-D was used since it could help in feature extraction comparatively much better.

Bidirectional Layer was used along with LSTM to make forward and backward pass, so as to make sequence learning more efficient.

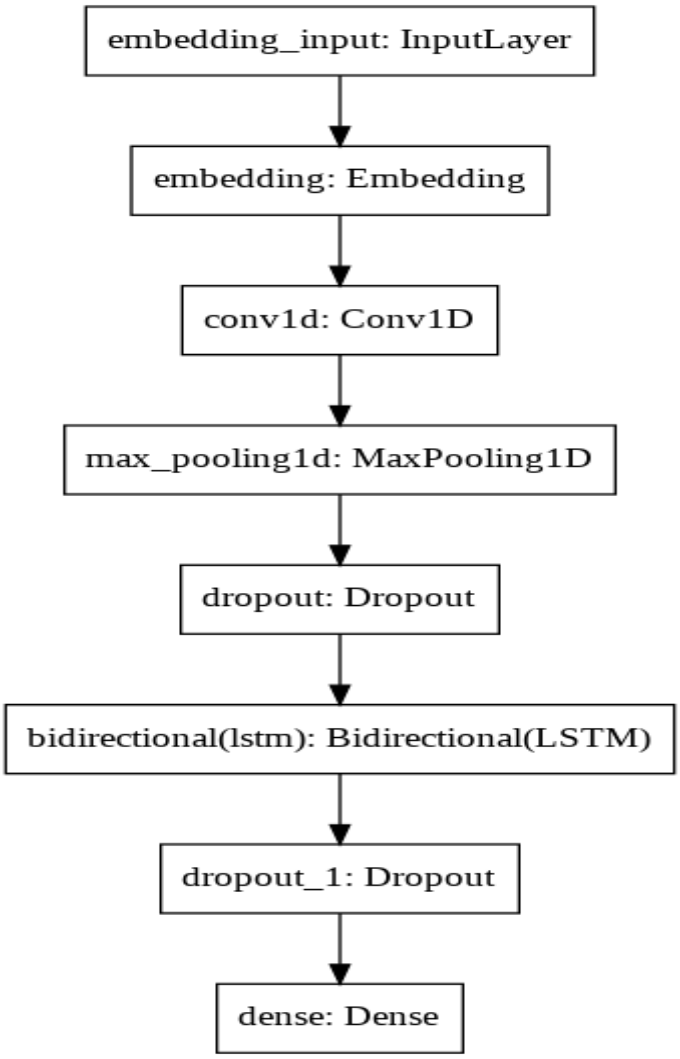
An Early Call-back was implemented to keep the track of "Validation Loss "with patience of 3, so that it can stop training if loss increased consecutively for 3 times in a row.

Given Below is the code snippet of the core model:

```
 model = Sequential()

  model.add(Embedding(vocab_size, embedding_dim,input_length=train_padd
ed.shape[1]))
  model.add(Conv1D(filters=256, kernel_size=3, padding='same', activati
on='relu'))
  model.add(MaxPooling1D(pool_size=2))
  model.add(Dropout(0.3))
  model.add(Bidirectional(LSTM(embedding_dim)))
  model.add(Dropout(0.3))
  model.add(Dense(12, activation='softmax'))

  model.summary()
  optim = tf.keras.optimizers.Adam()
  model.compile(
      loss='categorical_crossentropy',
      optimizer=optim,
      metrics=['accuracy'],
  )
  return model
epochs = 10
batch_size = 32
tqdm_callback = tfa.callbacks.TQDMProgressBar()
```

```
callbacks_list = [EarlyStopping(monitor='val_loss', mode='min', patienc
e=3, verbose=0),tqdm_callback]

model=create_model()
history=model.fit(train_padded, training_labels, shuffle=True ,
                  epochs=epochs, batch_size=batch_size,
                  callbacks=callbacks_list,validation_data=(validatio
n_padded, validation_labels))

loss,acc=model.evaluate(validation_padded, validation_labels)
print("Validation Loss : ",loss)
print("Validation Accuracy : ",acc)
```

Model Architecture:



Optimizer: Adam

Epochs: 10

Batch Size: 32

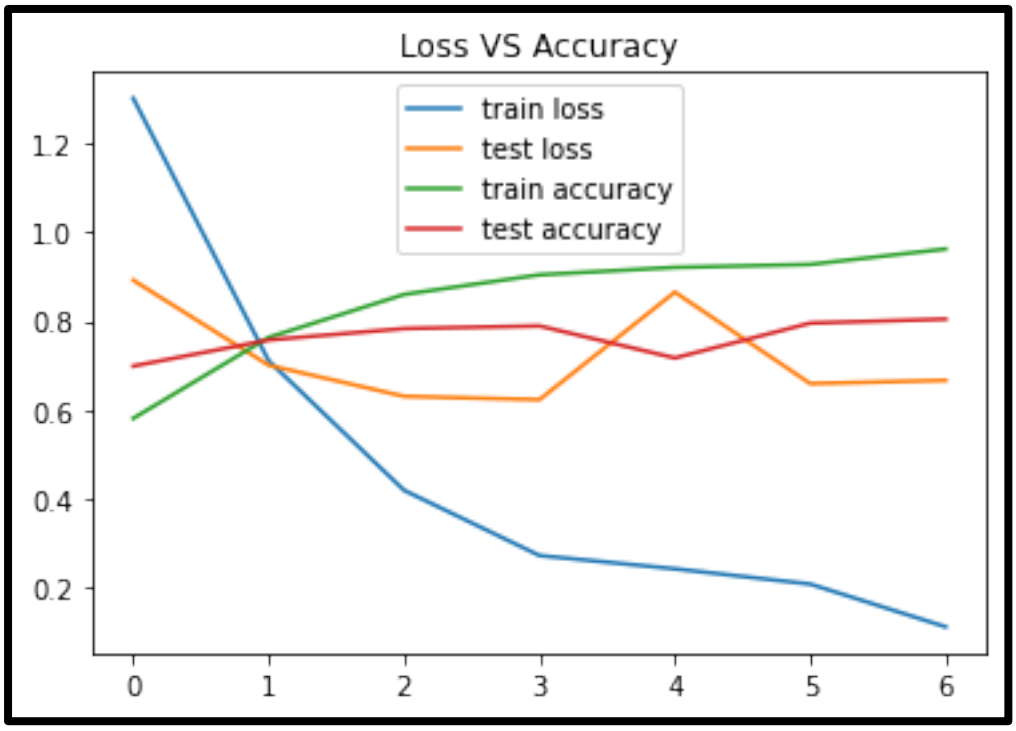
Loss Function: Categorical Cross entropy

Results:

The same model was implemented with glove 300-D embeddings and the results obtained from this model and results obtained from the first model were:

Model	Accuracy	Loss
Bidirectional LSTMs + Convnet 1-D + Latent Embeddings - 256-D	76.22 %	0.808
Bidirectional LSTMs + Convnet 1-D + GloVe Embeddings - 300-D	80.46 %	0.667

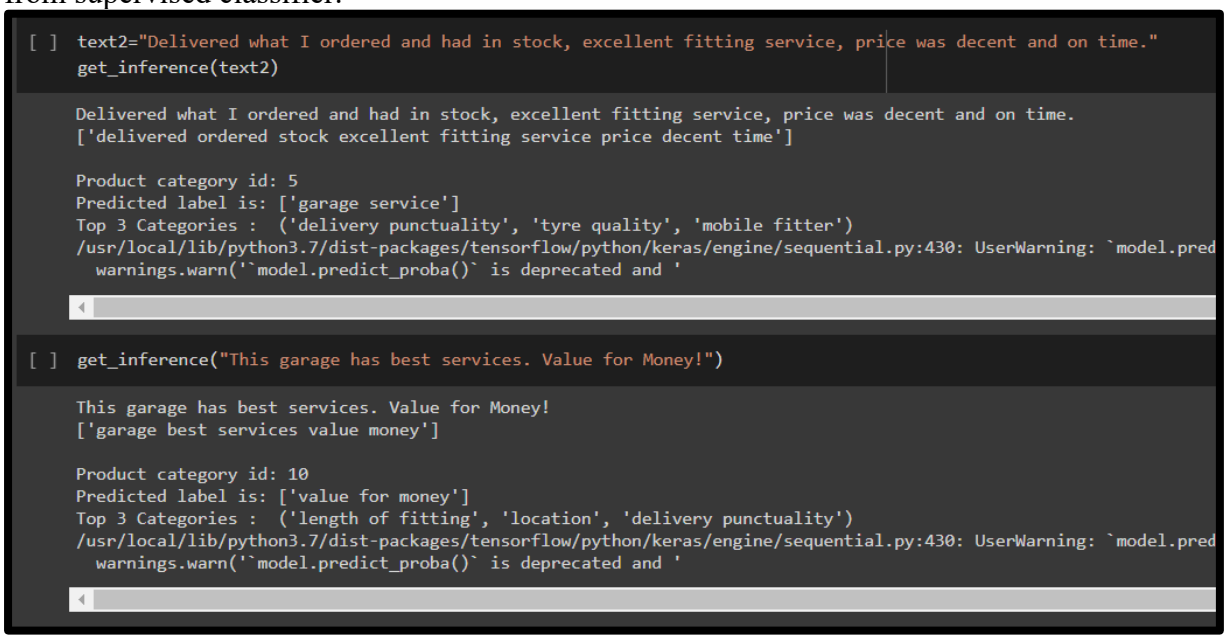
The custom function generated the following model performance graph:



A custom function was designed to save the model architecture, model weights and tokenizer. json for loading it for future use and getting inferences.

Later, the saved model architecture, weights and tokenizer was loaded to test for different text samples given below in inferences section.

A custom function was developed to test out inferences for getting predicted classified topics from supervised classifier.



```
[ ] text2="Delivered what I ordered and had in stock, excellent fitting service, price was decent and on time."
    get_inference(text2)

    Delivered what I ordered and had in stock, excellent fitting service, price was decent and on time.
    ['delivered ordered stock excellent fitting service price decent time']

    Product category id: 5
    Predicted label is: ['garage service']
    Top 3 Categories :  ('delivery punctuality', 'tyre quality', 'mobile fitter')
    /usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequential.py:430: UserWarning: `model.pred
      warnings.warn('`model.predict_proba()` is deprecated and '

[ ] get_inference("This garage has best services. Value for Money!")

    This garage has best services. Value for Money!
    ['garage best services value money']

    Product category id: 10
    Predicted label is: ['value for money']
    Top 3 Categories :  ('length of fitting', 'location', 'delivery punctuality')
    /usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequential.py:430: UserWarning: `model.pred
      warnings.warn('`model.predict_proba()` is deprecated and '
```

Scope For Future Work :

By querying big dataset, we can train the models with scope of more refined vocabulary.

Different unsupervised clustering algorithms could be implemented like hierarchical clustering.

State of the art models like Transformers or BERT could be used for more improved results and classification, given available computational constraints.

My given implementation consists of native TensorFlow embeddings and glove 300 D extracted embedding. In future, ELMO Embeddings could be also used for better results.