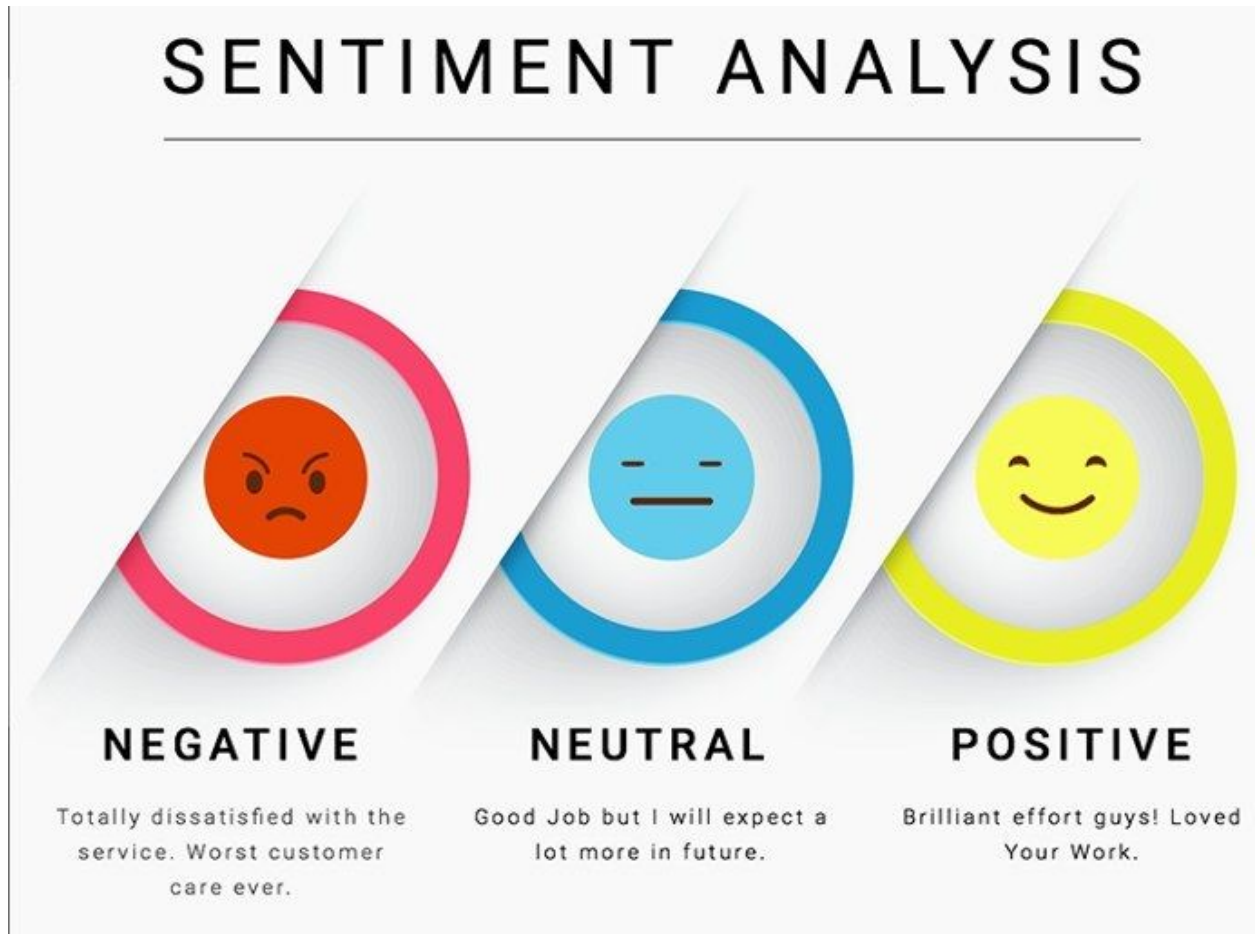


ASSIGNMENT 3 REPORT

Uncovering sentiments using EDGAR Dataset



Team 1

CASE STUDY INTRODUCTION

Goal: To build a sentiment analysis engine that can be used to understand the sentiments of earnings call transcript.

Review different methods to evaluate the sentiment analysis technique.

Dataset used: Earnings Call data for Q4 2018

Dataset used for Transfer Learning: IMDB dataset

DATA PREPARATION

Earnings call data for the Team Assigned company: **GOOGLE** is obtained through web scraping.

Code for web scraping:

https://github.com/Shravsriddhar/INFO-7374-Assignment-3/blob/master/Google_scrape.ipynb

Data is cleaned post scraping and labelled with the following labels: positive, negative and neutral.

Finally, to prepare dataset with all team's data, it is combined into a new json and cleaned.

EXPERIMENT 1

3 types of model are built:

1. Bag of Words Model
2. Word Embeddings: Glove
3. Recurrent Neural Network

BAG OF WORDS MODEL

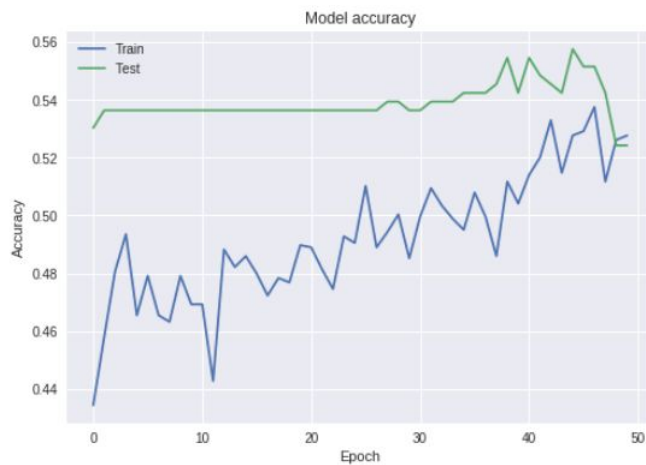
A BOW Model is where sentences are considered as bag of words. Words represent the content of sentence. Grammar and order of appearance are ignored in the process.

1. Features are extracted from text and used for modelling purpose.
2. Stop words, punctuations are removed from text.
3. Vocabulary is built with the given limit of features.
4. Words are scored across the document.
5. The document is converted into vector and sent across a Neural Network Model.

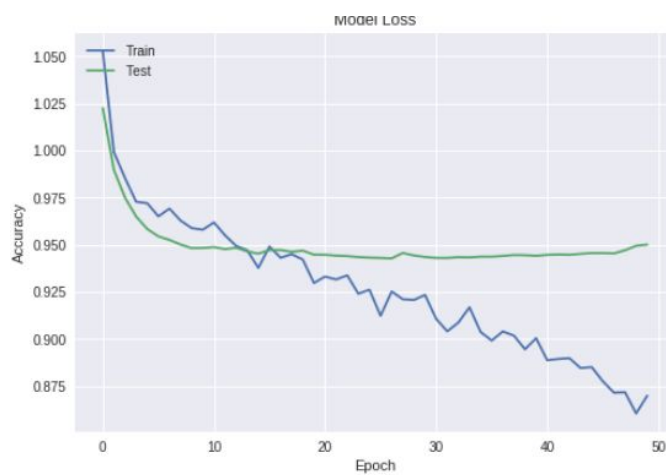
BEST MODEL PERFORMANCE:

Parameter Settings:

ACCURACY

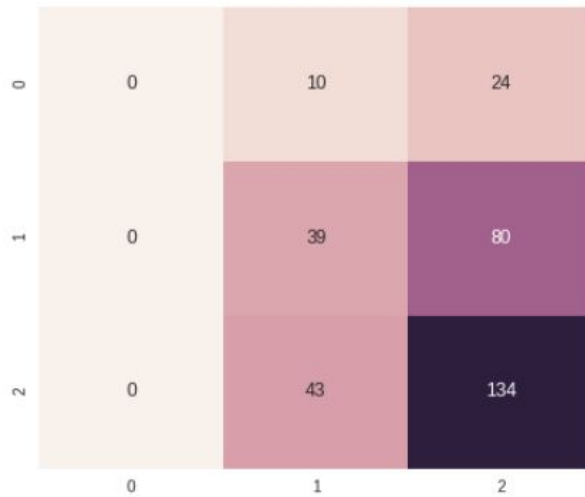


LOSS



CONFUSION MATRIX:

0: Negative, 1: Positive, 2: Neutral



Code

[link:https://github.com/Shravsriddhar/INFO-7374-Assignment-3/blob/master/Experiment1_BOW.ipynb](https://github.com/Shravsriddhar/INFO-7374-Assignment-3/blob/master/Experiment1_BOW.ipynb)

Link to sample observations taken during optimization:

<https://docs.google.com/spreadsheets/d/1-0wR00nzM0ds12M0gcA780TPm54hXMO0Fyzewvt8kAI/edit#gid=0>

OBSERVATIONS & INFERENCE

BOW provides one good way to start with in order to vectorize the text documents

Semantics are ignored which might cause loss of information.

Careful selection of vocabulary size is required as it tends to overfit the model.

WORD EMBEDDINGS: GLOVE

What is Word Embedding?

Word Embeddings are text converted into numbers. There are number of ways to represent the numeric forms. Types of embeddings: Frequency based, Prediction based.

Frequency Based: Tf-idf, Co-occurrence matrix

Prediction-Based: BOW, Skip-gram model

Using Pre-trained word vectors: Word2vec, Glove

Word Embedding is done for the experiment with the pre trained word vector Glove.

Glove version used : 100-dimensional GloVe embeddings of 400k words computed on a 2014 dump of English Wikipedia

Training is performed on an aggregated global word-word co-occurrence matrix, giving us a vector space with meaningful substructures.

STEPS PERFORMED

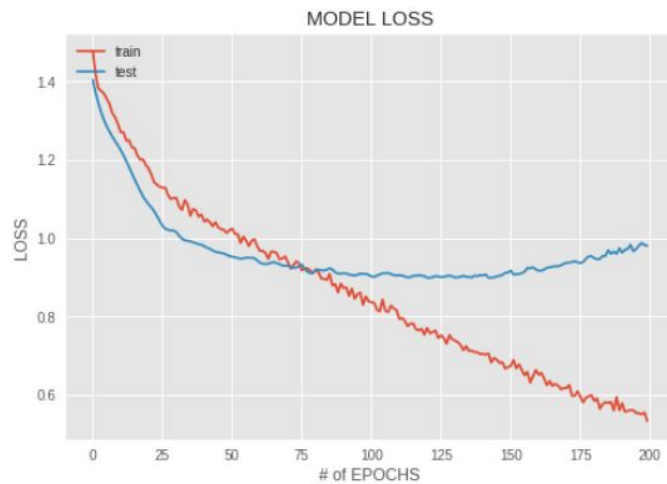
- Words are mapped to glove word embeddings.
- Co-occurrence count matrix is created for the above mappings.
- Embedding layer is created with the help of embedding matrix created and set as the input layer for model.
- Further, model is built and hyper-parameters are fine-tuned to obtain the best model

MODEL PERFORMANCE

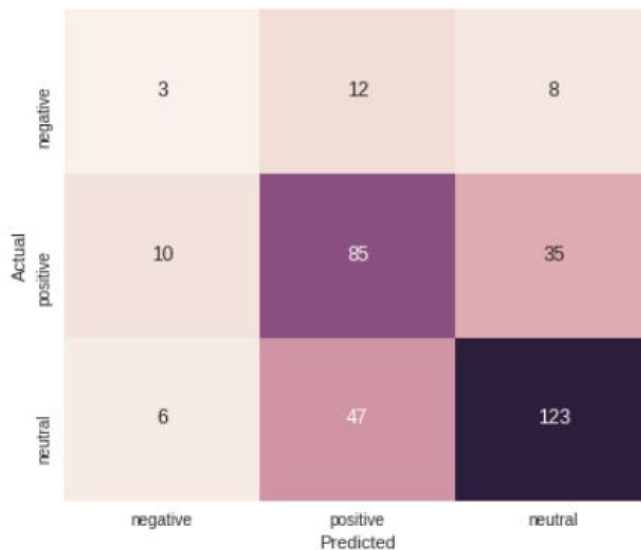
ACCURACY



LOSS



CONFUSION MATRIX



Code link:

https://github.com/Shravsriddhar/INFO-7374-Assignment-3/blob/master/Experiment1_Glove.ipynb

Link to sample observations taken during optimization:

<https://docs.google.com/spreadsheets/d/1-0wR00nzM0ds12M0gcA780TPm54hXMO0Fyzewvt8kAI/edit#gid=390076095>

OBSERVATIONS & INFERENCE

Model with pre-trained vectors Glove seems to perform better than the model built with bag of words representation.

Model initially overfitted, hence introduced regularization and Dropout to reduce the effect of overfitting.

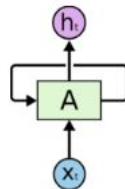
Word Embedding Model takes into account the semantics as well with the help of window size.

1. It learns semantics by passing a window over the corpus line-by-line and predicts the surroundings of a given word.
2. Another way is where it predicts a word given its surroundings.

RECURRENT NEURAL NETWORK MODEL

What is RNN?

It processes a sequence of vectors by applying a recurrence formula at every step.



Hidden state update in an RNN:

$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$, where h_t is the current state & h_{t-1} is the old state.

Weights are updated recursively through truncated backpropagation with time.

Problems in Vanilla RNN:

1. Exploding gradients
2. Vanishing gradients

To overcome the problems, a more complicated RNN structure LSTM is used which has better gradient flow properties.

LSTM: Maintains 2 hidden states at every time step. It is built with the help of following

gates:

- ❑ Forget gate- whether to erase cell
- ❑ Input gate- whether to write cell
- ❑ Gate gate- how much to write cell
- ❑ Output gate- how much to reveal cell

Back Propagation is done only through a single tanh and there is uninterrupted flow of gradient.

TYPES OF RNN IMPLEMENTED:

Simple RNN, GRU, LSTM, ConvLSTM2D, ConvLSTM2DCell, SimpleRNNCell, GRUCell, LSTMCell

STEPS PERFORMED:

1. Text is converted into sequence of integers with the help of keras API: **Tokenizer**.
2. Padding is performed to variables to equalize their length in order to avoid Contrived sequence problem.
3. RNN Model is built and best model is chosen post optimization.
4. Model performance is visualized with help of plots and confusion matrix.

For LSTM: From Embedding layer, new representations are passed to LSTM. These add recurrent connections to network to include information about data sequences.

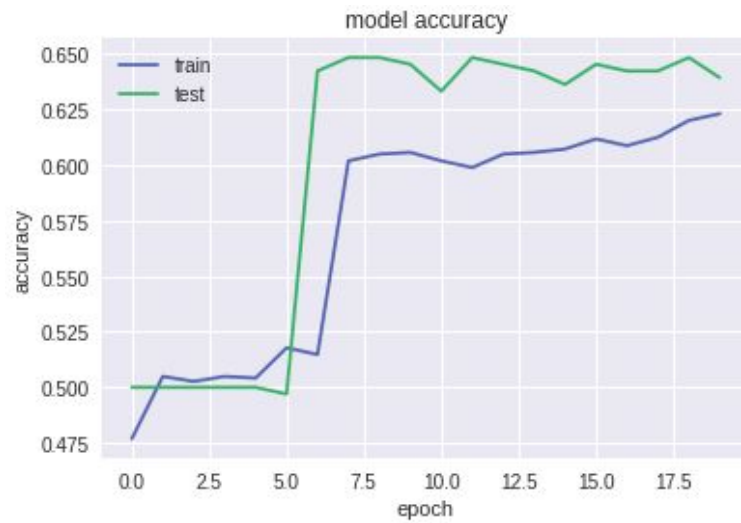
OBSERVATIONS & INFERENCE

RNN seems to perform the best when compared with Bag of words and Glove. LSTM is used in building the model.

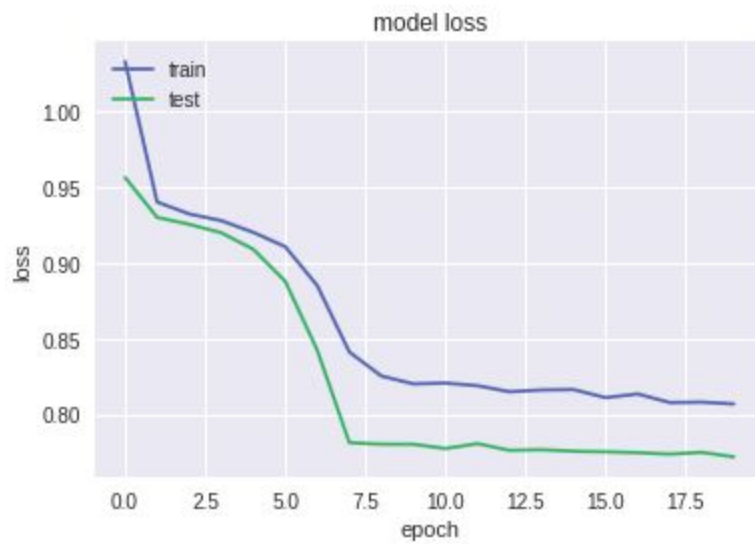
Model performance seems to increase with LSTM, maybe due to the steep gradients.

MODEL PERFORMANCE

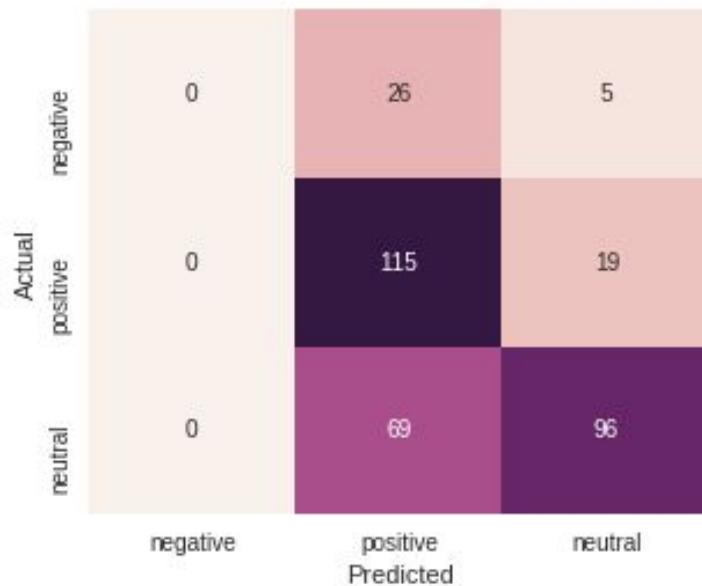
ACCURACY



LOSS



CONFUSION MATRIX



A confusion matrix heatmap for sentiment classification. The y-axis is labeled 'Actual' with categories 'negative', 'positive', and 'neutral'. The x-axis is labeled 'Predicted' with categories 'negative', 'positive', and 'neutral'. The cells contain counts: (Actual negative, Predicted negative) = 0, (Actual negative, Predicted positive) = 26, (Actual negative, Predicted neutral) = 5, (Actual positive, Predicted negative) = 0, (Actual positive, Predicted positive) = 115, (Actual positive, Predicted neutral) = 19, (Actual neutral, Predicted negative) = 0, (Actual neutral, Predicted positive) = 69, (Actual neutral, Predicted neutral) = 96. The diagonal elements (0, 115, 96) are highlighted in darker colors (dark purple, dark blue, and dark green respectively) to indicate correct classifications.

Actual	negative	positive	neutral
	0	26	5
	0	115	19
neutral	0	69	96
	negative	positive	neutral
Predicted			

Code link:

https://github.com/Shravsridhar/INFO-7374-Assignment-3/blob/master/Experiment1_RNN.ipynb

EXPERIMENT 2: TRANSFER LEARNING

What is Transfer Learning?

Transfer Learning is storing knowledge gained while solving one problem and applying it to a different but associated problem. In this task, it is illustrated with a Sentimental analysis problem.

Dataset used for transfer learning: Imdb dataset

About dataset: A binary set of 25,000 highly polar movie reviews for training, and 25,000 for testing.

Why Transfer Learning?

The dataset used for building models in experiment 1 is the earnings call data which is small in number with around 1650 texts. Since the dataset is small, model performance doesn't improve after a point. To overcome this problem, transfer learning is used where the model is trained with imdb data which is huge and then tested on earnings call data.

This helps in generalization as well.

STEPS PERFORMED:

1. Model is built and optimized with IMDB dataset.
2. Model is tested against the earnings call data.

MODEL PERFORMANCE:

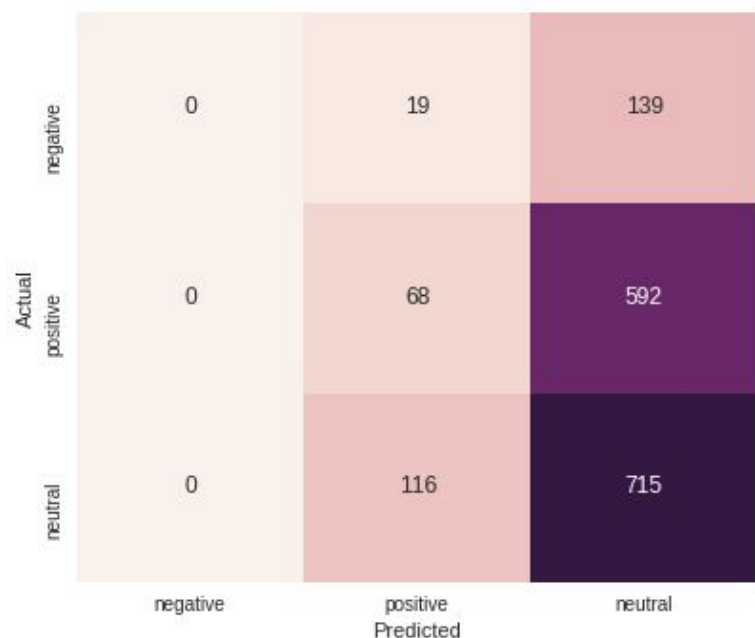
MODEL: **BAG OF WORDS**

ACCURACY: 66%, LOSS: 0.74

Code link:

https://github.com/Shravsriddhar/INFO-7374-Assignment-3/blob/master/BOW_TransferLearning.ipynb

CONFUSION MATRIX ON TESTING WITH FINANCE DATA:



Actual	negative	positive	neutral
	0	19	139
	0	68	592
neutral	0	116	715
		positive	neutral
		Predicted	

MODEL: **GLOVE**

CONFUSION MATRIX ON FINANCE DATA:

Actual	negative	56	4	98
	positive	258	26	376
	neutral	415	14	402
		negative	positive	neutral
		Predicted		

ACCURACY: 66%, LOSS: 0.64

Code link:

https://github.com/Shravsridhar/INFO-7374-Assignment-3/blob/master/BOW_TransferLearning.ipynb

Link to sample observations taken during optimization:

<https://docs.google.com/spreadsheets/d/1-0wR00nzM0ds12M0gcA780TPm54hXMO0Fyzewvt8kAI/edit#gid=815572437>

MODEL: **RNN**

L J

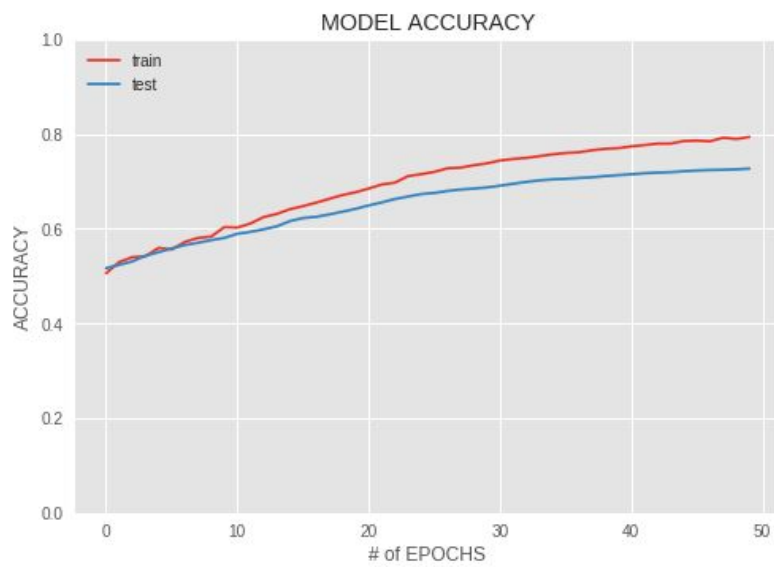
Actual	negative	96	0	62
	positive	422	0	238
	neutral	674	0	157
		negative	positive	neutral
		Predicted		

Code link: <https://github.com/Shravsriddhar/INFO-7374-Assignment-3>

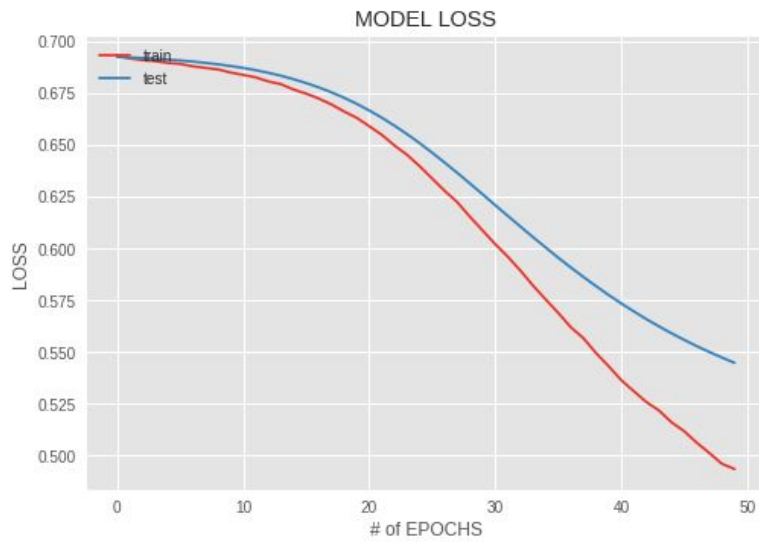
Note: Trial experiments are added in the trial folder in repo

MODEL PERFORMANCE WITH RESPECT TO TRAINING DATA(IMDB):

ACCURACY



LOSS



This model seems to perform well with training data which is large imdb dataset, and also it doesn't suffer from overfitting issues and trains faster as well.

EXPERIMENT 3: Using API's

STEPS PERFORMED:

1. The following API's are used and sentiment scores are obtained for the entire dataset: AWS Comprehend, Google Cloud Natural language, Azure Text analysis API, Watson Natural Language understanding.
2. Obtained scores are normalized and average score is taken to determine the overall sentiment.
3. Confusion matrix is obtained with the manually labelled data and compared.

AWS Comprehend:

- ❑ Input format: The required text documents are to be saved in a csv format.
- ❑ A task for sentimental analysis is created and the file is passed from s3 bucket.
- ❑ Output is received with the following scores: mixed score, positive score, negative score, neutral score, output label.
 - ❑ Range: 0 to 1
- ❑ Required score for each sentence is extracted from obtained output file.

Azure Text analysis:

- ❑ Input format: Json format with id, text and language for each document.
- ❑ Text API is called using a http POST request.
- ❑ With API key, the texts are passed and output score is obtained.
- ❑ Output returns a sentiment score for each document, ranging from 0 (negative) to 1 (positive).

IBM Watson Natural language understanding:

- ❑ Input format: Text format with all the text documents and id.
- ❑ Created API key on cloud for accessing IBM watson NLU API.
- ❑ Called the method in API for getting the features required for the text including sentiment score, keywords, emotions.
- ❑ Output format contains sentiment score and label for the text document parsed.

Google API:

- ❑ Input format: List containing all the text documents.
- ❑ Got the credentials from Google and validated them.

- ❑ Call is made to api and response is stored in list.
- ❑ List is then accessed to get overall score.
- ❑ Output contains the text id along with sentiment score for each of it.

CONFUSION MATRIX FOR NORMALIZED SCORE VS MANUAL LABELED SCORE

Actual	negative	34	0	124
	positive	202	1	456
	neutral	181	0	652
		negative	positive	neutral
		Predicted		

ACCURACY:

True Positive + True Negative + True neutral / Total = 687/1650 = 47 %

Accuracy is 47% for the normalized score with respect to keeping the manually labelled scores as true data.

EXPERIMENT 4: ENSEMBLE LEARNING USING AUTOML

OBJECTIVE:

Model is built with the help of raw sentiment scores from API's.

Input: Sentiment score from 4 API

Output: Manually labelled score

STEPS PERFORMED:

1. The following Automl are used for this experiment: AutoSKLearn ,TPOT, H2O
2. File is passed with the above described format and split into 80:20 ratio prior passing into their classifiers.
3. Performance is obtained for the three classifiers along with time and compared selecting the best one.

METRIC	AUTO SKLEARN	TPOT	H2O
ABOUT	It uses Bayesian optimization, meta-learning and ensemble construction	A genetic search algorithm to find the best parameters and model ensembles	Stacked Ensembles are used in the process
MODEL PERFORMANCE	Accuracy seems to be lower when compared to the other two methods	Accuracy seems to be low	Accuracy seems
ALGORITHM USED	Xgboost classifier	Random Forest Classifier	Random Forest Estimator
BEST ACCURACY RECEIVED	58%	63%	60%

Which Model is better? Resultant Model from Experiment 3 or 4?

According to the observations recorded, it is clear that Experiment 4 accuracy is better. Model obtained from Normalized score gives accuracy of 47%. With Experiment 4, using any one of the automl above gives a higher accuracy than Experiment 3 model. TPOT seems to take a training time of less than 15 minutes to give an accuracy of 63%.

Hence, Ensemble learning using Automl is recommended.

Code link:

H2O

[https://github.com/Shravsridhar/INFO-7374-Assignment-3/blob/master/H2O_Edgar\(1\).ipynb](https://github.com/Shravsridhar/INFO-7374-Assignment-3/blob/master/H2O_Edgar(1).ipynb)

TPOT

<https://github.com/Shravsridhar/INFO-7374-Assignment-3/blob/master/TPOT.ipynb>

AutoSKLearn

https://github.com/Shravsridhar/INFO-7374-Assignment-3/blob/master/auto_sklearn_api.py

FINAL MODEL TO USE IN PRODUCTION:

Transfer learning with Glove model is recommended as final model for production.

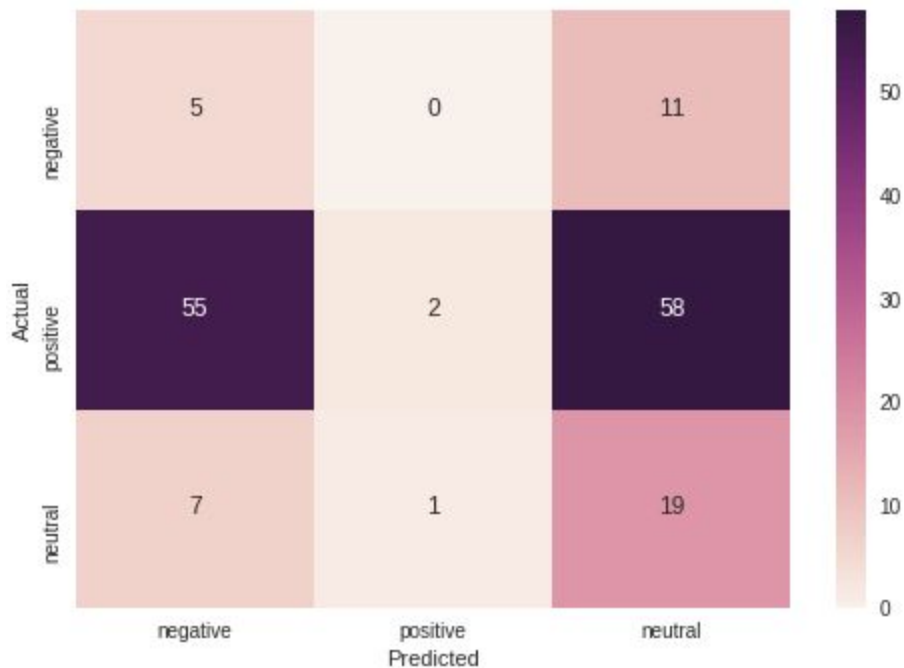
Why Transfer Learning is recommended? Since the financial dataset is not large enough, a much better model would be the one built with imdb dataset which is large and tested upon finance data. Also, high accuracy is obtained in this case.

Why Glove Model?

BOW model doesn't cover the semantics part and RNN seems to take more training time. In a production environment, always the trade-off between model performance and time taken for training is to be considered. Considering all the factors, Glove seems to be the best from the experiments conduction in this case study.

OUTPUT RELATED TO TESTING:

Testing with GE Earnings Call Data:



It is observed that giving a new data, recommended model gives model accuracy of 66%. In addition, training time is less than 5 minutes.

INFERENCE:

- ❑ Further optimizing the model with more data could help give us better results. Also, to start with we could use one of the automl to get a clear idea. API Scores can be sent as an input to the classifier as done in the above experiment.
- ❑ If Memory constraint is not given, then RNN: LSTM could be deployed as well. But, it always takes more time to train and uses too much memory which is a big drawback in production.
- ❑ For research problems with text, RNN's could be optimized and used.