Assignment 3

Team 8: Manoj Karru, Nishant Gandhi, Emily Strong
INFO 7374 Spring 2019

GitHub: https://github.com/erstrong/DeepNeuralNetworksandAI/tree/master/Assignment3

Experiment 1

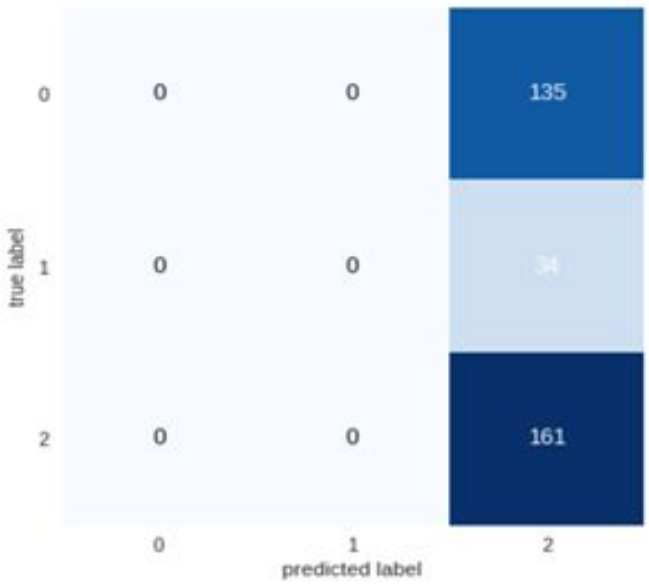
We combined the scored EDGAR quarterly earnings transcripts from the class and trained an RNN, an MLP with GloVe embeddings, and a bag of words model with an MLP. The highest test accuracy of 80% was achieved with the bag of words model, but it failed to label any texts as negative. The architectures and data are below.

Experiment 1 [RNN]:
Best model settings:
optimizer= 'rmsprop'
loss='binary_crossentropy'
epochs=25
batch_size=128
Structure:

1. Embedding Layer,
2. SimpleRNN,
3. Dense Layer

Accuracy:
train=0.66, test=0.65

Confusion Matrix:
positive: 0, negative:1 ,neutral :2



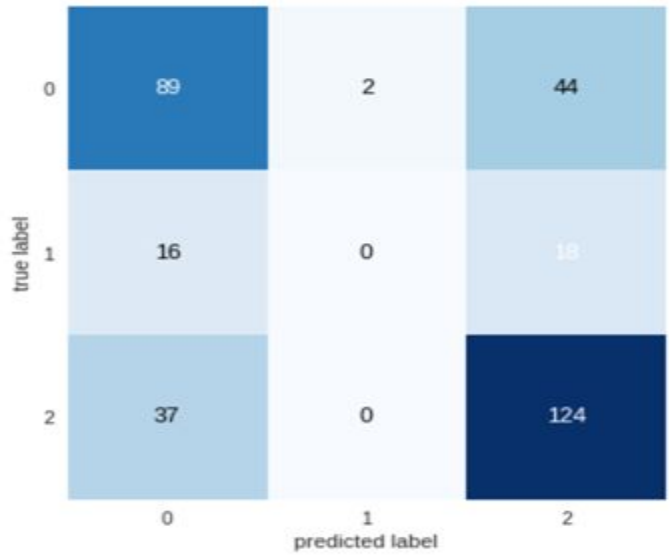
Experiment 1 With [GloVe]
Best model settings:
X xoptimizer= ‘rmsprop’
loss=’binary_crossentropy’
epochs=25
batch_size=128
Structure:

1. Embedding Layer,
2. Flatten,
3. Dense Layer

Accuracy:
train=0.68, test=0.68

Confusion Matrix:

positive: 0, negative:1 ,neutral :2



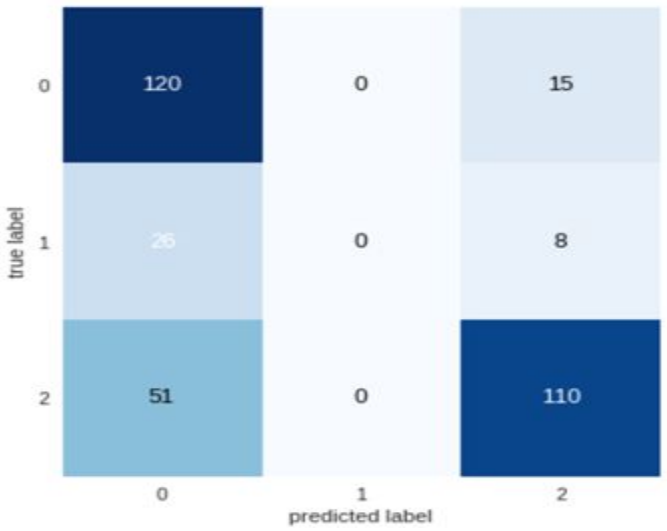
Experiment 1 [Bag of Words]:
Best model settings:
optimizer= ‘rmsprop’
loss=’binary_crossentropy’
epochs=25
batch_size=128
Structure:

- Dense Layer

Accuracy:
train=0.81, test=0.80

Confusion Matrix:

positive: 0, negative:1 ,neutral :2



Experiment 2

In the second set of experiments, we trained our models on the IMDB review data set and then tested those models on the EDGAR earnings call transcripts. Since the IMDB data set only has positive and negative reviews, we removed the neutral text from the EDGAR data for scoring.

We first set a baseline of performance by applying a simple tokenizer to the data and using the keras Embeddings layer in an MLP which had 52.8% accuracy on the call transcript data, i.e. it was performing at chance. For the bag of words method, we tested several approaches:

- Frequency matrix generated with the keras Tokenizer with an MLP
- Sklearn CountVectorizer with an MLP
- Sklearn CountVectorizer and TF-IDF with an MLP and with Naive Bayes

We used the pre-computed GloVe Wikipedia embeddings with an MLP, and also tried fine-tuning the embedding weights. And finally we tried an RNN and LSTM. For each of these we experimented with the number of layers and tested the best performing one against the transcript data. The summary table shows the scores from the best performing models.

From each of the three approaches (Bag of Words, GloVe, and RNN) we selected the best model and fine tuned it on 80% of the EDGAR data, reserving 20% for final scoring. Before fine tuning, the GloVe tuned embedding weights gave the best performance on the transcript data with an accuracy of 73.5%. The RNN had a higher accuracy score, but in looking at the confusion matrix this was because it was only predicting positives and because of the skew in the data distribution that gave it a deceptively high score. After fine tuning, the CountVectorizer bag of words had the highest accuracy at 82.8% and was predicting both classes. This is thus the best model from Experiment 2.

Transfer Learning Summary Table

Model	IMDB Loss	IMDB Accuracy	EDGAR Loss	EDGAR Accuracy	True Negative	False Positive	False Negative	True Positive
Embeddings	0.286	0.879	0.877	0.528	99	58	325	329
Tokenizer BOW	0.275	0.889	0.745	0.654	69	88	193	461
CountVectorizer BOW	0.343	0.875	0.714	0.663	43	114	159	495
TF-IDF + Naive Bayes	n/a	0.572	n/a	0.443	93	64	388	266
TF-IDF + MLP	0.299	0.881	0.999	0.466	100	57	376	278
GloVe	0.586	0.735	0.918	0.566	114	43	309	345
GloVe Tuned Embeddings	0.400	0.825	0.572	0.735	51	106	109	545
RNN	0.394	0.422	0.838	0.806	0	157	0	654
LSTM	0.383	0.867	0.793	0.194	157	0	654	0
Fine-Tuned BOW	n/a	n/a	0.488	0.828	12	24	4	123
Fine-Tuned GloVe	n/a	n/a	0.587	0.755	10	26	14	113
Fine-Tuned RNN	n/a	n/a	0.585	0.791	5	31	3	124

Experiment 3

In the third set of experiment, we used four off the self NLP services from four different vendors. They are IBM Watson NLP, GCP NLP, Azure NLP & Amazon Comprehend. Following are the results.

IBM
Accuracy: 0.5518
Confusion Matrix:

	Predicted Negative	Predicted Neutral	Predicted Positive
Actual Negative	35	10	112
Actual Neutral	68	291	479

Actual Positive	35	35	584

GCP
Accuracy: 0.6009
Confusion Matrix:

	Predicted Negative	Predicted Neutral	Predicted Positive
Actual Negative	8	122	27
Actual Neutral	9	686	143
Actual Positive	7	350	297

Azure
Accuracy: 0.4978
Confusion Matrix:

	Predicted Negative	Predicted Neutral	Predicted Positive
Actual Negative	2	144	11
Actual Neutral	9	750	79
Actual Positive	9	576	69

Amazon
Accuracy: 0.5579
Confusion Matrix:

	Predicted Mixed	Predicted Negative	Predicted Neutral	Predicted Positive
Actual Mixed	0	0	0	0
Actual Negative	1	20	113	23
Actual Neutral	1	44	682	111
Actual Positive	5	30	401	218

Experiment 4

In the fourth set of experiments we trained three auto machine learning libraries on the raw scores from Experiment 3. We ran TPOT three times experimenting with number of generations and population size. The simplest configuration (5 generations, population of 20) gave the best results with 61.8% accuracy. We ran Auto-Sklearn for an hour. Many of the generations had

errors related to no predictions being made for one of the categories (most likely negative). The final model was better at predicting negatives than the TPOT model but had a worse overall accuracy at 60.9%. The H2O model had the best performance overall with an 83.7% overall accuracy and 55.9% accuracy rate for the negative data.

TPOT Confusion Matrix
Accuracy: 0.6181818182

	Predicted Negative	Predicted Neutral	Predicted Positive
Actual Negative	1	23	4
Actual Neutral	4	116	38
Actual Positive	3	54	87

Auto-Sklearn Confusion Matrix
Accuracy: 0.6090909091

	Predicted Negative	Predicted Neutral	Predicted Positive
Actual Negative	9	9	10
Actual Neutral	11	90	57
Actual Positive	14	19	111

H2O AutoML Confusion Matrix
Accuracy: 0.8369230769

	Predicted Negative	Predicted Neutral	Predicted Positive
Actual Negative	19	9	6
Actual Neutral	0	140	21
Actual Positive	0	17	113

Final Data Analysis

We selected the H2O automl model as the best as it has both a high accuracy on the call transcripts used in the four experiments and had a higher accuracy on negative sentiment text than the other models. We ran this model on the GE call transcript that was not included in the initial call transcripts and it achieved a 59.1% accuracy and had a 100% error rate on the negative texts.

This could be an indication that the model doesn't generalize well, but more likely is a reflection of the GE data. The text was challenging to score as GE had a very bad quarter but most of that was discussed in very factual ways without expressing opinion, and when explicitly negative

sentences did occur they were usually in a mixed paragraph. This sort of ambiguity in text sentiment highlights why it is important to include a neutral category, which most of the data was.

Final GE Results

Accuracy: 0.5911602209944752

	Predicted Negative	Predicted Neutral	Predicted Positive
Actual Negative	0	6	1
Actual Neutral	1	83	53
Actual Positive	1	12	24