MPL Summer'22 Research Internship and PE sem5 '22

Shreyansh Rai

Introduction

Problem Statement:

- Detecting self promotion in interviews
- Identifying self promoting statements in an answer
- Score an answer/interview based on self-promotion
- Provide feedback on how to improve
- Explain the prediction or the score

In the Beginning

- Started learning about ML and NLP from Stanford's courses (CS224N, CS229 and another lecture series on NLP with traditional ML)

- Learnt how to use essential Python libraries like NumPy, Matplotlib, Pandas, SpaCy, SciKit Learn, NLTK, Gensim, TensorFlow and PyTorch

Reading and Study Material

- Went through paper: Self-promotion Statements in Video Resumes: Frequency, intensity, and gender effects on job applicant evaluation - Link
- Went through paper: SimSensei Kiosk: A Virtual Human Interviewer for Healthcare Decision Support - Link
- Transformers: Link
- GPT-2: Link
- BERT: Link

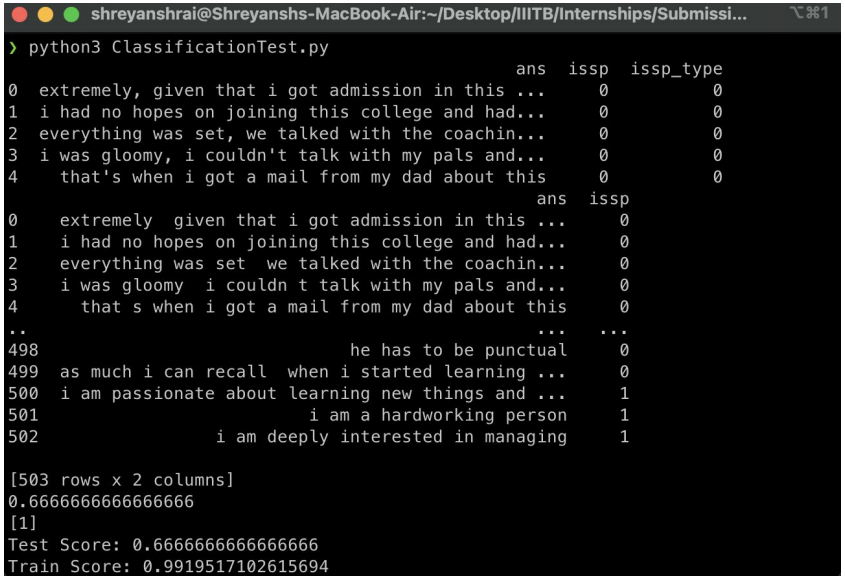
Using Sentiment Analysis

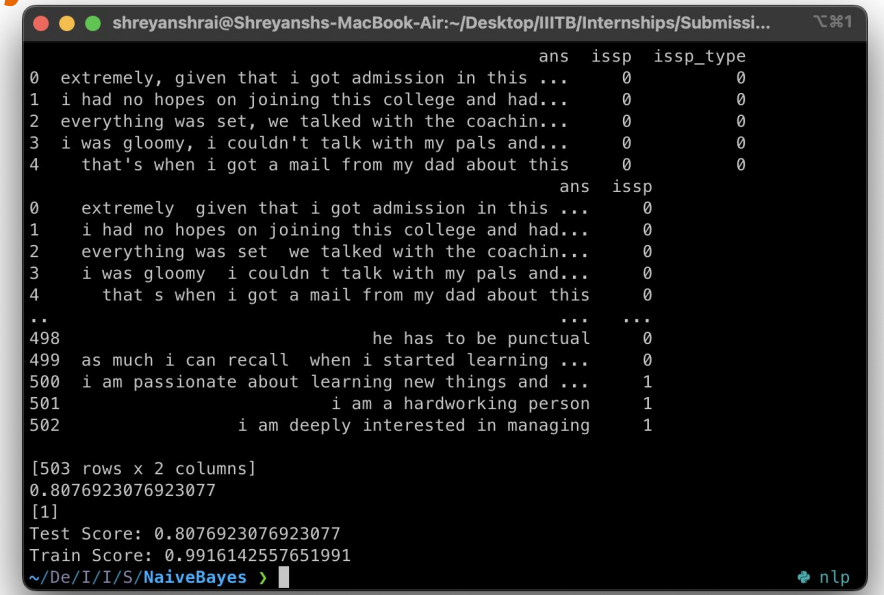
Methodology : Using a list of “Positive” words and their synonyms coupled with certain rules about pronouns being present or absent, tense of the sentence, presence of verbs and negations, Tried to filter out self promotional sentences

Accuracy : Low - 30-40%

Reason for failure : Compound sentences and phrases were impossible to handle. There was a need to identify probability that a certain word would be present in the self promotional sentence and then use that information to classify. This led to the Naive Bayes approach.

Classical ML : Multinomial Bayes Model





500 Sentences : Training - 90% Testing - 10%
Testing Accuracy : 66.66%
As data for training reduces the data available for calculating probabilities goes down resulting in lower accuracy.

500 Sentences : Training - 95 % Testing - 5%
Testing accuracy : 80.77%

Model - Basic Neural Network

This was our first approach to solving the problem using Deep Learning methods. We created a basic Neural Network on PyTorch and used randomly initialised embeddings.

For 100 sentences, on a 90/10 split, we achieved an accuracy of 70%.

For 400 sentences, on a 90/10 split, we achieved an accuracy of 72%

For 400 sentences, on a 95/5 split, we achieved an accuracy of 73%

For 500 sentences, on a 95/5 split, we achieved an accuracy of 73%

Model - GloVe Embeddings with a basic Neural Network

As recommended, instead of using randomly initialised word embeddings, we used GloVe Vectors as embeddings. As the GloVe vectors were pre trained, we expected the model to have more contextual word embeddings.

This achieved a marginally better accuracy but sacrificed a lot on training time and resources.

For 500 sentences, on a 95/5 split, we achieved an accuracy around 75-76%

Recurrent Neural Net

```
3/3 [==============================] - 0s 77ms/step - loss: 0.0043 - accuracy: 1.0000 - val_loss: 0.8562 - val_accuracy: 0.9000
Epoch 37/40
3/3 [==============================] - 0s 80ms/step - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.8710 - val_accuracy: 0.9000
Epoch 38/40
3/3 [==============================] - 0s 79ms/step - loss: 0.0038 - accuracy: 1.0000 - val_loss: 0.8698 - val_accuracy: 0.9000
Epoch 39/40
3/3 [==============================] - 0s 80ms/step - loss: 0.0035 - accuracy: 1.0000 - val_loss: 0.8756 - val_accuracy: 0.9000
Epoch 40/40
3/3 [==============================] - 0s 77ms/step - loss: 0.0031 - accuracy: 1.0000 - val_loss: 0.8984 - val_accuracy: 0.9000
```

Since we wanted to have some context over the input we researched a little for an ideal solution. We came across RNNs and I thought to implement that. (This implementation was supported by the TF docs https://www.tensorflow.org/guide/keras/rnn)

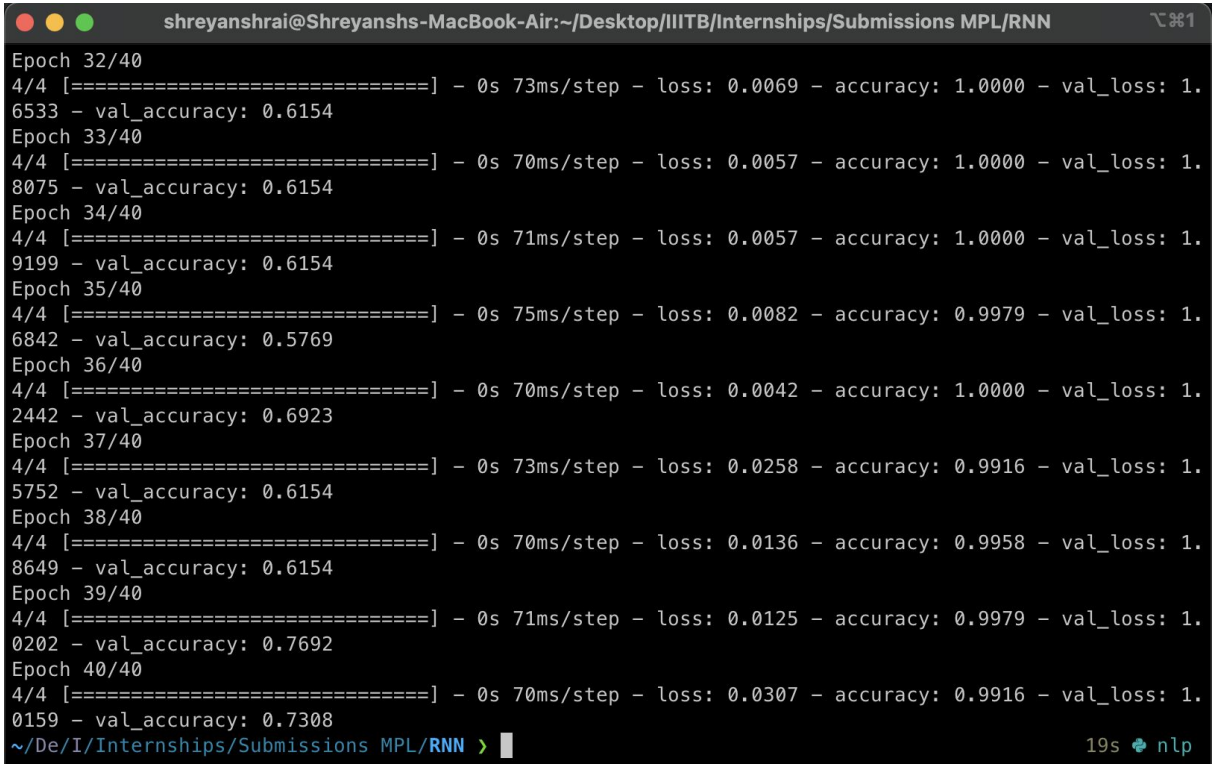
400 Sentences : 95% Training data and 5% testing data - 90% accuracy

However this model turned out to be far from ideal. We decided to test this further by increasing the data set size to 500.

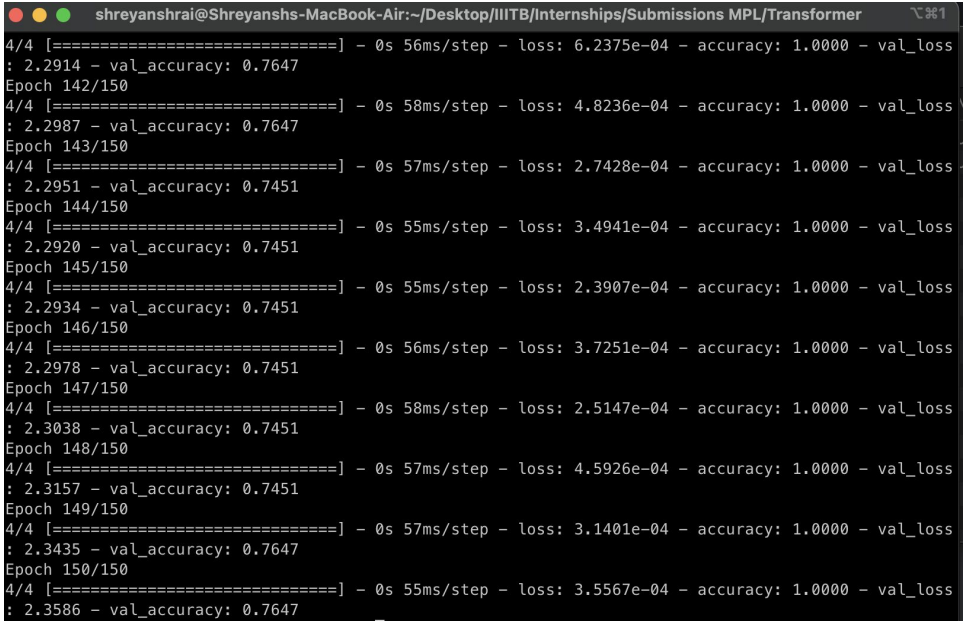
500 Sentences : 95 % Training data and 5% testing data - 73.1% accuracy

(Photo on next slide)

RNN 500 Sentence test



Transformers



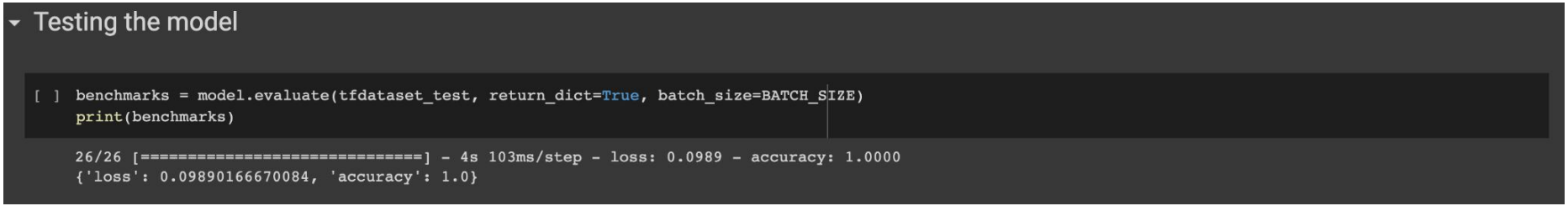
After our attempts with Neural Nets we were guided to try Transformers and Language models :

500 Sentences - 90% Training and 10% Testing split
Accuracy : 76.47%

Transformers was a topic we had trouble dealing with owing to our limited time spent working with NNs and is something we hope to better in the future.

DistilBERT - A language model based approach

500 Sentences : Train size - 80% Test size - 20%
Testing Accuracy : 93.07% (always in the ballpark of 92-95% depending on the random test set)



500 Sentences : Train size - 90% Test size - 10%
Testing Accuracy : 100.00% (always in the ballpark of 94-100% depending on the random test set)

The best model thus far was the pretrained Bert model most impressively obtaining an accuracy of 93% on just 80% of the data and correctly classifying around 93 out of 100 sentences that were passed in for testing.

A note on the data :

Most sentences were long and complicated : (green is self promotion)

at that time, my idea was good and new but the problem was that i didn't have any proof that my idea will work but after some days of research, i found proof that it is possible to implement and then we all agreed on that topic.

apart from learning new technologies, learning about different architectures and design patterns followed by different product of it industry, i would also like to lead a team where i can deploy my technical skills a well as managerial skills

i never loved taking risks i always wanted things to be straight forward but as the time has passed i have learnt to take risks and believe me i have taken a few very serious risks and one of them is dropping iit kharagpur and joining iiit bengaluru

Enough with the tailored data set, Time for real data!

So the data tested above was dummy data that is essentially the same interview but broken down into sentences and then labelled true or false for self promotion by us.

Below is a script I wrote for the automation of data collection. Which also presents some limitations :

1) We relied on the google speech api to transcribe the interviews, and noticed words that must have been mis-transcribed because they were out of place in the sentence.
2) The data size was relatively low, which made training the model harder.

Running the script to get mp4 to text :

https://github.com/Shreyansh-Rai/MP4toTranscript

Drawing your attention to our final push :

The data : It has osprom columns 1-3 and 5, The transcript data and the time column can be used as a unique identifier.

https://github.com/Shreyansh-Rai/MPLSummer-22_Identifying_Self_Promotion/blob/main/FINAL_SUBMISSION_NOTEBOOK_LINGFEAT_ROBERTA.ipynb

Has the code that we used. We tried modify the data to make it more usable than its original state, details of which again can be found in the notebook linked above.

On to the processing done on the data.

Models and Challenges -

We focused on all the columns, average scores given for each osprom column (details in ipynb) and the transcript data for those columns.

Our initial testing was based on bert embeddings trained on our transcript data and then this was passed into a neural net and the output column was varied to be from osprom 1 through 5 and the average of all the values as well. The tests we performed had around 76% accuracy on Interview embeddings via BERT and the target was the average of all the osprom columns.

Solving the problem of data shortage -

We were not able to get over the data shortage and just the embeddings were clearly not enough for the prediction task. Worsened exceptionally due the limited data that we had. This led to us reading papers and this one in particular was recommended to us and caught our attention it was on handcrafted features to improve readability analysis tasks.

LINGFEAT - Github

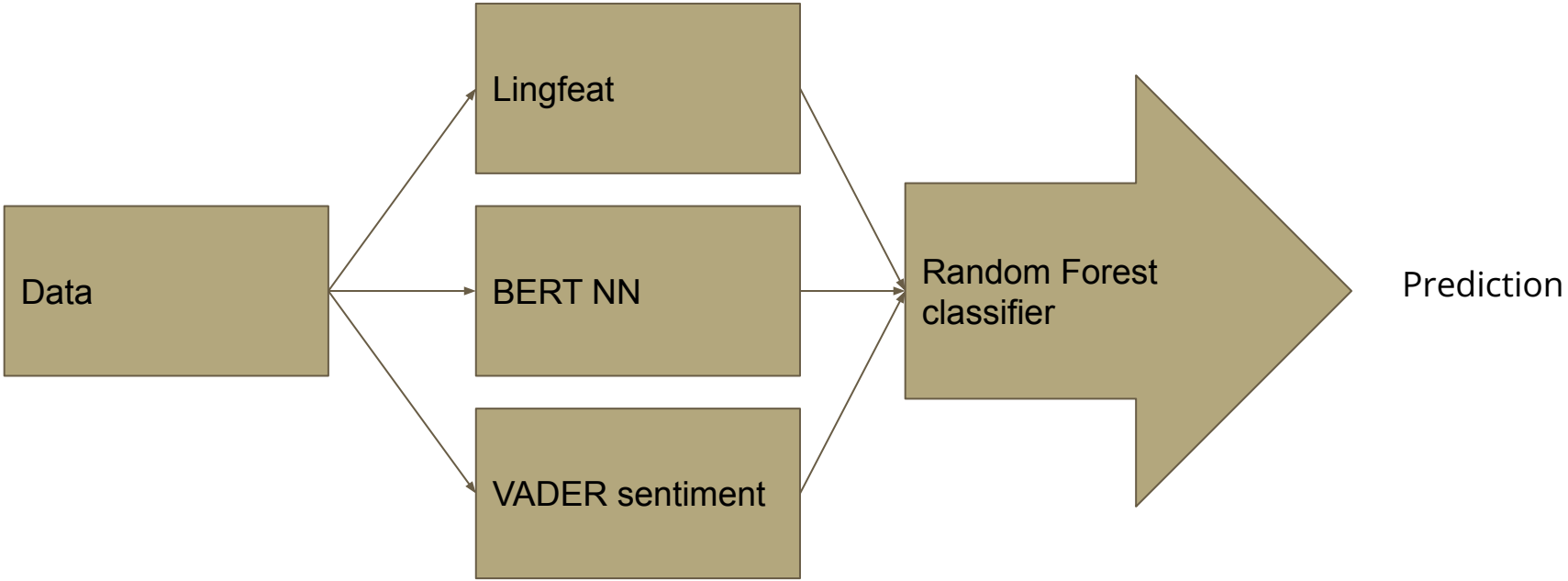
Intermediate models :

We made several models mainly tuning the ensemble to figure out which non neural model helps achieve the best accuracy for the task. This is similar to the approach followed in the paper I mentioned above. For details on our intermediate steps I recommend going through our notebook. It has the models in a well labelled manner.

Neural Model (BERT) - LF-Features + Sentiment -> Non-Neural Model (Random Forest) - LF-Features + Sentiment

This was our final model that we ran explainability on. There was another model after this with a robustly optimised BERT, RoBERTa that gave a slightly better accuracy of 93.44% on testing data (details in notebook) but since we have not run the explainability code for that model, we will focus on the aforementioned model.

The idea was to use a subset of the handcrafted lingfeat features that we thought are important in identifying self promotion, and using the transformer embeddings and NN, appending vader sentiment compound scores to this, and additionally the lingfeat features to predict avg(osprom)


Data
Lingfeat
BERT NN
VADER sentiment
Random Forest classifier
Prediction

Explainability code on this model

Model analysis revealed that the most important parameters for determining the target output always had the bert based predictions in the top 4 features that helped determine the target output, along with some pos features.