
MPL Summer'22 Research Internship

— 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)
- Currently going through the Sequence Model course on coursera to better understand the inner workings of some models and get better performance out of them
- 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

```
shreyanshrai@Shreyanshs-MacBook-Air:~/Desktop/IIITB/Internships/Submissi...
> python3 ClassificationTest.py

      ans  issp  issp_type
0 extremely, given that i got admission in this ... 0 0
1 i had no hopes on joining this college and had... 0 0
2 everything was set, we talked with the coachin... 0 0
3 i was gloomy, i couldn't talk with my pals and... 0 0
4 that's when i got a mail from my dad about this 0 0

      ans  issp
0 extremely given that i got admission in this ... 0
1 i had no hopes on joining this college and had... 0
2 everything was set we talked with the coachin... 0
3 i was gloomy i couldn t talk with my pals and... 0
4 that s when i got a mail from my dad about this 0
..
498 he has to be punctual 0
499 as much i can recall when i started learning ... 0
500 i am passionate about learning new things and ... 1
501 i am a hardworking person 1
502 i am deeply interested in managing 1

[503 rows x 2 columns]
0.6666666666666666
[1]
Test Score: 0.6666666666666666
Train Score: 0.9919517102615694
```

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.

```
shreyanshrai@Shreyanshs-MacBook-Air:~/Desktop/IIITB/Internships/Submissi...
      ans  issp  issp_type
0 extremely, given that i got admission in this ... 0 0
1 i had no hopes on joining this college and had... 0 0
2 everything was set, we talked with the coachin... 0 0
3 i was gloomy, i couldn't talk with my pals and... 0 0
4 that's when i got a mail from my dad about this 0 0

      ans  issp
0 extremely given that i got admission in this ... 0
1 i had no hopes on joining this college and had... 0
2 everything was set we talked with the coachin... 0
3 i was gloomy i couldn t talk with my pals and... 0
4 that s when i got a mail from my dad about this 0
..
498 he has to be punctual 0
499 as much i can recall when i started learning ... 0
500 i am passionate about learning new things and ... 1
501 i am a hardworking person 1
502 i am deeply interested in managing 1

[503 rows x 2 columns]
0.8076923076923077
[1]
Test Score: 0.8076923076923077
Train Score: 0.9916142557651991
~/De/I/I/S/NaiveBayes > nlp
```

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

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

```
shreyanshrai@Shreyanshs-MacBook-Air:~/Desktop/IIITB/Internships/Submissions MPL/RNN
Epoch 32/40
4/4 [=====] - 0s 73ms/step - loss: 0.0069 - accuracy: 1.0000 - val_loss: 1.6533 - val_accuracy: 0.6154
Epoch 33/40
4/4 [=====] - 0s 70ms/step - loss: 0.0057 - accuracy: 1.0000 - val_loss: 1.8075 - val_accuracy: 0.6154
Epoch 34/40
4/4 [=====] - 0s 71ms/step - loss: 0.0057 - accuracy: 1.0000 - val_loss: 1.9199 - val_accuracy: 0.6154
Epoch 35/40
4/4 [=====] - 0s 75ms/step - loss: 0.0082 - accuracy: 0.9979 - val_loss: 1.6842 - val_accuracy: 0.5769
Epoch 36/40
4/4 [=====] - 0s 70ms/step - loss: 0.0042 - accuracy: 1.0000 - val_loss: 1.2442 - val_accuracy: 0.6923
Epoch 37/40
4/4 [=====] - 0s 73ms/step - loss: 0.0258 - accuracy: 0.9916 - val_loss: 1.5752 - val_accuracy: 0.6154
Epoch 38/40
4/4 [=====] - 0s 70ms/step - loss: 0.0136 - accuracy: 0.9958 - val_loss: 1.8649 - val_accuracy: 0.6154
Epoch 39/40
4/4 [=====] - 0s 71ms/step - loss: 0.0125 - accuracy: 0.9979 - val_loss: 1.0202 - val_accuracy: 0.7692
Epoch 40/40
4/4 [=====] - 0s 70ms/step - loss: 0.0307 - accuracy: 0.9916 - val_loss: 1.0159 - val_accuracy: 0.7308
~/De/I/Internships/Submissions MPL/RNN > 19s nlp
```


Transformers

```
shreyanshrai@Shreyanshs-MacBook-Air:~/Desktop/IIITB/Internships/Submissions MPL/Transformer
4/4 [=====] - 0s 56ms/step - loss: 6.2375e-04 - accuracy: 1.0000 - val_loss
: 2.2914 - val_accuracy: 0.7647
Epoch 142/150
4/4 [=====] - 0s 58ms/step - loss: 4.8236e-04 - accuracy: 1.0000 - val_loss
: 2.2987 - val_accuracy: 0.7647
Epoch 143/150
4/4 [=====] - 0s 57ms/step - loss: 2.7428e-04 - accuracy: 1.0000 - val_loss
: 2.2951 - val_accuracy: 0.7451
Epoch 144/150
4/4 [=====] - 0s 55ms/step - loss: 3.4941e-04 - accuracy: 1.0000 - val_loss
: 2.2920 - val_accuracy: 0.7451
Epoch 145/150
4/4 [=====] - 0s 55ms/step - loss: 2.3907e-04 - accuracy: 1.0000 - val_loss
: 2.2934 - val_accuracy: 0.7451
Epoch 146/150
4/4 [=====] - 0s 56ms/step - loss: 3.7251e-04 - accuracy: 1.0000 - val_loss
: 2.2978 - val_accuracy: 0.7451
Epoch 147/150
4/4 [=====] - 0s 58ms/step - loss: 2.5147e-04 - accuracy: 1.0000 - val_loss
: 2.3038 - val_accuracy: 0.7451
Epoch 148/150
4/4 [=====] - 0s 57ms/step - loss: 4.5926e-04 - accuracy: 1.0000 - val_loss
: 2.3157 - val_accuracy: 0.7451
Epoch 149/150
4/4 [=====] - 0s 57ms/step - loss: 3.1401e-04 - accuracy: 1.0000 - val_loss
: 2.3435 - val_accuracy: 0.7647
Epoch 150/150
4/4 [=====] - 0s 55ms/step - loss: 3.5567e-04 - accuracy: 1.0000 - val_loss
: 2.3586 - val_accuracy: 0.7647
```

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.

Finale : DistilBERT - A language model based approach

▼ Testing the model

```
benchmarks = model.evaluate(tfdataset_test, return_dict=True, batch_size=BATCH_SIZE)
print(benchmarks)

51/51 [=====] - 8s 121ms/step - loss: 0.2774 - accuracy: 0.9307
{'loss': 0.2774370312690735, 'accuracy': 0.9306930899620056}
```

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)

▼ Testing the model

```
[ ] benchmarks = model.evaluate(tfdataset_test, return_dict=True, batch_size=BATCH_SIZE)
print(benchmarks)

26/26 [=====] - 4s 103ms/step - loss: 0.0989 - accuracy: 1.0000
{'loss': 0.09890166670084, 'accuracy': 1.0}
```

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 as 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