

Deep NLP Classification

Daniel Alvarado

What is the situation?

- Chatbot prompt is 'Describe a time when you have acted as a resource for someone else'

- 80 responses

- If a response is 'not flagged', the user can continue talking to the bot. If it is 'flagged', the user is referred to help

- 125 resumes were queried from Indeed.com with keyword 'data scientist', location 'Vermont'

- If a resume is 'not flagged', the applicant can submit a modified resume at a later date. If it is 'flagged', the applicant is invited to interview.

Who cares?

- For the chatbot responses; mental health professionals, user of the chatbot, family of the user
- For the resumes; employer, potential employee

The Data

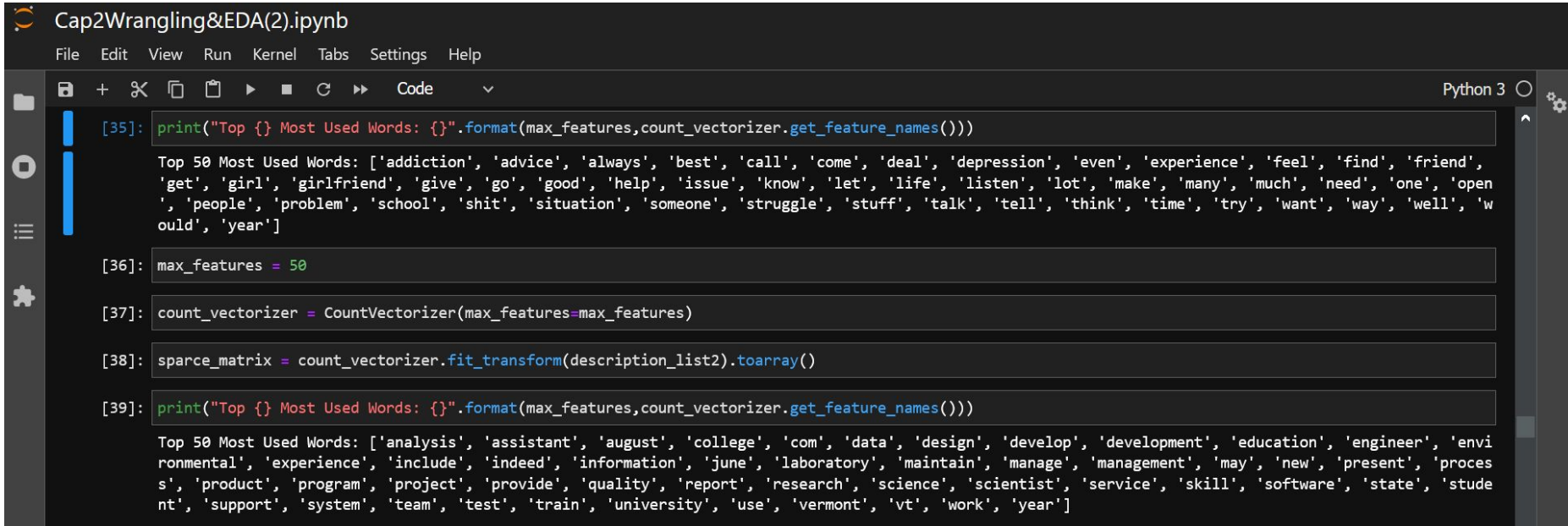
- Downloaded from <https://www.kaggle.com/samdeeplearning/deepnlp>
- This data is used to identify the variables behind flagged therapy chatbot responses and flagged resume submissions
- Flagged chatbot responses result in referring the user to professional mental health help and flagged resumes are invited to interview.

EDA

- Word clouds were created for both datasets
- Top 50 most used words
- Most common 5-word phrases
- Least common words
- Average word length and average sentence length analysis
- I will only upload a few images for the sake of presentation length



Top 50 Most Used Words



```
Cap2Wrangling&EDA(2).ipynb
File Edit View Run Kernel Tabs Settings Help

[35]: print("Top {} Most Used Words: {}".format(max_features, count_vectorizer.get_feature_names()))

Top 50 Most Used Words: ['addiction', 'advice', 'always', 'best', 'call', 'come', 'deal', 'depression', 'even', 'experience', 'feel', 'find', 'friend',
'get', 'girl', 'girlfriend', 'give', 'go', 'good', 'help', 'issue', 'know', 'let', 'life', 'listen', 'lot', 'make', 'many', 'much', 'need', 'one', 'open',
', 'people', 'problem', 'school', 'shit', 'situation', 'someone', 'struggle', 'stuff', 'talk', 'tell', 'think', 'time', 'try', 'want', 'way', 'well', 'w
ould', 'year']

[36]: max_features = 50

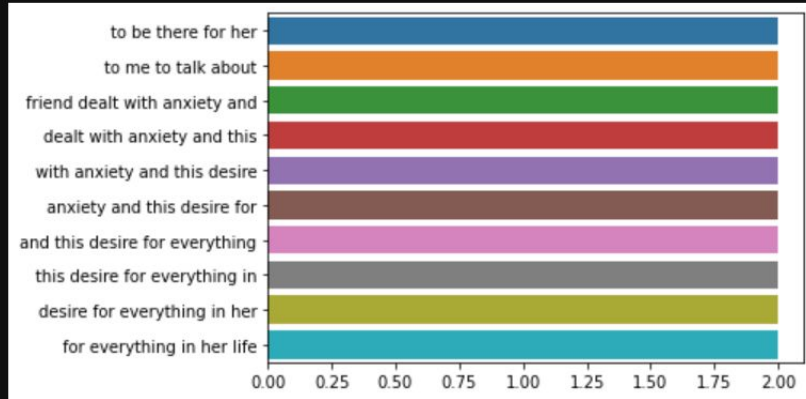
[37]: count_vectorizer = CountVectorizer(max_features=max_features)

[38]: sparse_matrix = count_vectorizer.fit_transform(description_list2).toarray()

[39]: print("Top {} Most Used Words: {}".format(max_features, count_vectorizer.get_feature_names()))

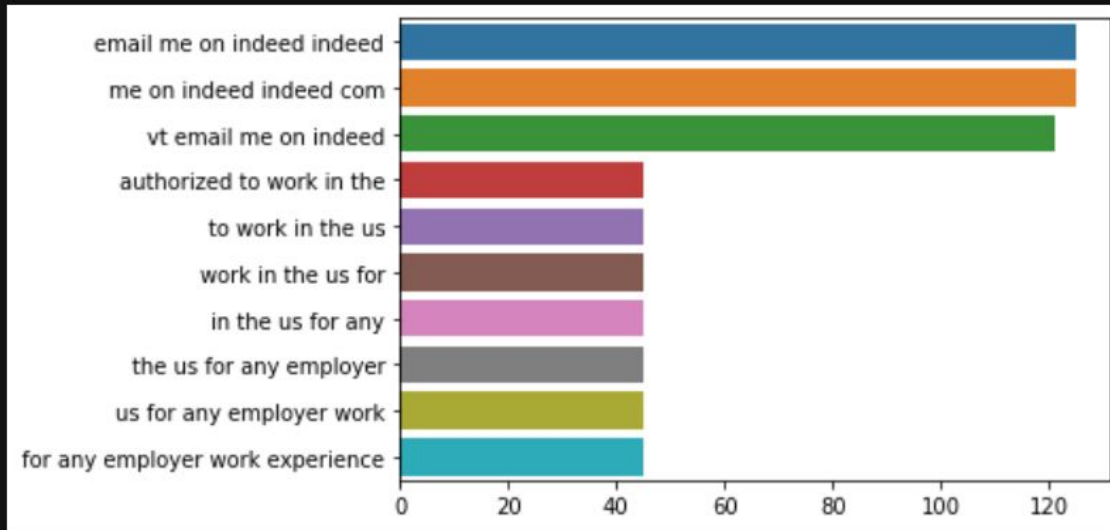
Top 50 Most Used Words: ['analysis', 'assistant', 'august', 'college', 'com', 'data', 'design', 'develop', 'development', 'education', 'engineer', 'envi
ronmental', 'experience', 'include', 'indeed', 'information', 'june', 'laboratory', 'maintain', 'manage', 'management', 'may', 'new', 'present', 'proces
s', 'product', 'program', 'project', 'provide', 'quality', 'report', 'research', 'science', 'scientist', 'service', 'skill', 'software', 'state', 'stude
nt', 'support', 'system', 'team', 'test', 'train', 'university', 'use', 'vermont', 'vt', 'work', 'year']
```

Most Common 5-word Phrases in Chatbot Responses



The pentagrams for the chatbot responses paints a clear picture into how people responded to the prompt. Being there for someone and dealing with anxiety are the main themes.

Most Common 5-word Phrases in Resumes



The resume pentagrams are identical. 5 word phrases in resumes pertain to working and invitations to email via [indeed.com](https://www.indeed.com)

Modeling and Methods

- Supervised learning problem
- Binary classification: 0 = not flagged, 1 = flagged
- Class label had to be converted to binary values

Models

- Accuracy and precision were the metrics to measure for both the classification models
- Multinomial and Gaussian Naive Bayes
- Random Forest
- Support Vector Classifier
- Tensorflow deep learning model

Modeling Steps

- Minimal to light preprocessing required, data was well written and abundant
- Used sklearn's CountVectorizer for preprocessing
- Train/Test split of 75/25

Results

Multinomial Naive Bayes

```
In [8]: x = data.response_text
y = data['class']
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = 0.25, random_state=42)
x_train_dtm = vect.fit_transform(x_train)
x_test_dtm = vect.transform(x_test)
NB.fit(x_train_dtm,y_train)
y_predict = NB.predict(x_test_dtm)
metrics.accuracy_score(y_test,y_predict)
```

Out[8]: 0.65

```
In [9]: metrics.precision_score(y_test,y_predict)
```

Out[9]: 0.3333333333333333

```
In [10]: metrics.f1_score(y_test,y_predict)
```

Out[10]: 0.46153846153846156

Random Forest

```
In [11]: rf = RandomForestClassifier(max_depth=10,max_features=10)
rf.fit(x_train_dtm,y_train)
rf_predict = rf.predict(x_test_dtm)
metrics.accuracy_score(y_test,rf_predict)
```

Out[11]: 0.75

Results

Multinomial Naive Bayes for Resumes

```
In [12]: x = data2.resume_text
y = data2['class']
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = 0.25, random_state=42)
x_train_dtm = vect.fit_transform(x_train)
x_test_dtm = vect.transform(x_test)
NB.fit(x_train_dtm,y_train)
y_predict = NB.predict(x_test_dtm)
metrics.accuracy_score(y_test,y_predict)
```

```
Out[12]: 0.71875
```

```
In [13]: metrics.precision_score(y_test,y_predict)
```

```
Out[13]: 0.75
```

```
In [14]: metrics.f1_score(y_test,y_predict)
```

```
Out[14]: 0.39999999999999997
```

Random Forest for Resumes

```
In [15]: rf = RandomForestClassifier(max_depth=10,max_features=10)
rf.fit(x_train_dtm,y_train)
rf_predict = rf.predict(x_test_dtm)
metrics.accuracy_score(y_test,rf_predict)
```

```
Out[15]: 0.65625
```

The precision and f1_scores for the random forest models was 0, so these metrics will be ignored.

Results

- The random forest classifier gave the highest accuracy score for the chatbot responses at 0.75, 0.10 more than the multinomial naive bayes classifier.
- For the resumes, the multinomial naive bayes classifier gave a slightly higher accuracy score than the random forest. 0.72 for multinomial naive bayes and 0.65 for the random forest classifier.

Results

- The f1 score and precision score of the chatbot responses are 0.46 and 0.33, respectively for multinomialNB. For the resumes, the multinomialNB f1 score and precision scores are 0.4 and 0.75, respectively.
- Precision can be thought of as the fraction of positive predictions that actually belong to the positive class. The multinomialNB classifier on the resumes yield a high precision score, close to its accuracy score.
- This model is what we're looking for in a resume classifier, a predictor that has a low false positive rate!

Deep learning

```
Epoch 1/30
6/6 [=====] - 3s 326ms/step - loss: 0.6898 - accuracy: 0.7226 - val_loss: 0.6810 - val_accuracy: 0.7317
Epoch 2/30
6/6 [=====] - 0s 11ms/step - loss: 0.6814 - accuracy: 0.7084 - val_loss: 0.6729 - val_accuracy: 0.7317
Epoch 3/30
6/6 [=====] - 0s 12ms/step - loss: 0.6750 - accuracy: 0.6927 - val_loss: 0.6654 - val_accuracy: 0.7317
Epoch 4/30
6/6 [=====] - 0s 12ms/step - loss: 0.6669 - accuracy: 0.7061 - val_loss: 0.6584 - val_accuracy: 0.7317
Epoch 5/30
6/6 [=====] - 0s 11ms/step - loss: 0.6612 - accuracy: 0.7020 - val_loss: 0.6524 - val_accuracy: 0.7317
Epoch 6/30
6/6 [=====] - 0s 12ms/step - loss: 0.6506 - accuracy: 0.7280 - val_loss: 0.6458 - val_accuracy: 0.7317
Epoch 7/30
6/6 [=====] - 0s 14ms/step - loss: 0.6481 - accuracy: 0.7085 - val_loss: 0.6400 - val_accuracy: 0.7317
Epoch 8/30
6/6 [=====] - 0s 14ms/step - loss: 0.6357 - accuracy: 0.7378 - val_loss: 0.6344 - val_accuracy: 0.7317
Epoch 9/30
6/6 [=====] - 0s 14ms/step - loss: 0.6428 - accuracy: 0.6912 - val_loss: 0.6303 - val_accuracy: 0.7317
Epoch 10/30
6/6 [=====] - 0s 13ms/step - loss: 0.6325 - accuracy: 0.7105 - val_loss: 0.6259 - val_accuracy: 0.7317
Epoch 11/30
6/6 [=====] - 0s 14ms/step - loss: 0.6236 - accuracy: 0.7234 - val_loss: 0.6216 - val_accuracy: 0.7317
Epoch 12/30
6/6 [=====] - 0s 16ms/step - loss: 0.6301 - accuracy: 0.6939 - val_loss: 0.6167 - val_accuracy: 0.7317
Epoch 13/30
6/6 [=====] - 0s 13ms/step - loss: 0.6181 - accuracy: 0.7124 - val_loss: 0.6113 - val_accuracy: 0.7317
Epoch 14/30
6/6 [=====] - 0s 14ms/step - loss: 0.5985 - accuracy: 0.7470 - val_loss: 0.6067 - val_accuracy: 0.7317
```

Deep learning

```
Epoch 15/30
6/6 [=====] - 0s 14ms/step - loss: 0.6209 - accuracy: 0.6836 - val_loss: 0.6034 - val_accuracy: 0.7317
Epoch 16/30
6/6 [=====] - 0s 14ms/step - loss: 0.5903 - accuracy: 0.7414 - val_loss: 0.5993 - val_accuracy: 0.7317
Epoch 17/30
6/6 [=====] - 0s 14ms/step - loss: 0.6027 - accuracy: 0.7076 - val_loss: 0.5960 - val_accuracy: 0.7317
Epoch 18/30
6/6 [=====] - 0s 14ms/step - loss: 0.5859 - accuracy: 0.7302 - val_loss: 0.5925 - val_accuracy: 0.7317
Epoch 19/30
6/6 [=====] - 0s 15ms/step - loss: 0.5857 - accuracy: 0.7217 - val_loss: 0.5897 - val_accuracy: 0.7317
Epoch 20/30
6/6 [=====] - 0s 10ms/step - loss: 0.5868 - accuracy: 0.7146 - val_loss: 0.5877 - val_accuracy: 0.7317
Epoch 21/30
6/6 [=====] - 0s 12ms/step - loss: 0.5864 - accuracy: 0.7174 - val_loss: 0.5861 - val_accuracy: 0.7317
Epoch 22/30
6/6 [=====] - 0s 11ms/step - loss: 0.5962 - accuracy: 0.6925 - val_loss: 0.5845 - val_accuracy: 0.7317
Epoch 23/30
6/6 [=====] - 0s 12ms/step - loss: 0.5766 - accuracy: 0.7174 - val_loss: 0.5823 - val_accuracy: 0.7317
Epoch 24/30
6/6 [=====] - 0s 13ms/step - loss: 0.5455 - accuracy: 0.7567 - val_loss: 0.5805 - val_accuracy: 0.7317
Epoch 25/30
6/6 [=====] - 0s 13ms/step - loss: 0.5629 - accuracy: 0.7267 - val_loss: 0.5794 - val_accuracy: 0.7317
Epoch 26/30
6/6 [=====] - 0s 14ms/step - loss: 0.5948 - accuracy: 0.6745 - val_loss: 0.5785 - val_accuracy: 0.7317
Epoch 27/30
6/6 [=====] - 0s 14ms/step - loss: 0.5889 - accuracy: 0.6794 - val_loss: 0.5768 - val_accuracy: 0.7317
Epoch 28/30
6/6 [=====] - 0s 13ms/step - loss: 0.5684 - accuracy: 0.7063 - val_loss: 0.5754 - val_accuracy: 0.7317
Epoch 29/30
6/6 [=====] - 0s 14ms/step - loss: 0.5496 - accuracy: 0.7260 - val_loss: 0.5745 - val_accuracy: 0.7317
Epoch 30/30
6/6 [=====] - 0s 13ms/step - loss: 0.5563 - accuracy: 0.7174 - val_loss: 0.5740 - val_accuracy: 0.7317
```

Further Research

Comparing different chatbot prompt responses would give more insight into what responses are flagged or not.

- Granted, these prompts would have to be similar, in the same vein as the original. An example prompt could be 'Explain a situation where you provided comfort to someone.'

- For the resumes, including other jobs in the tech industry would yield a more comprehensive assessment of what employers look for in a resume when hiring.

- It would also be interesting to analyze data science resumes over a broader region. This may yield insights into what companies across this hypothetical region require in a data scientist.