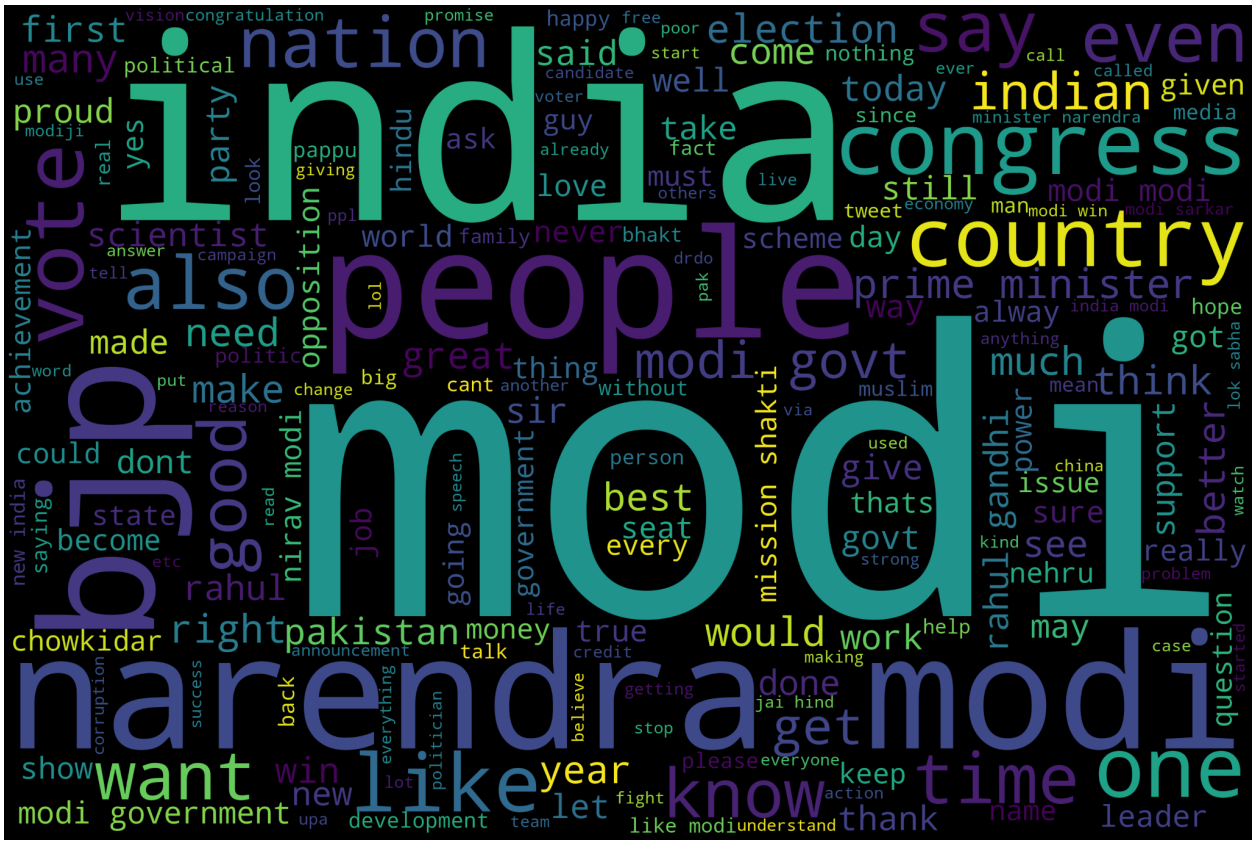
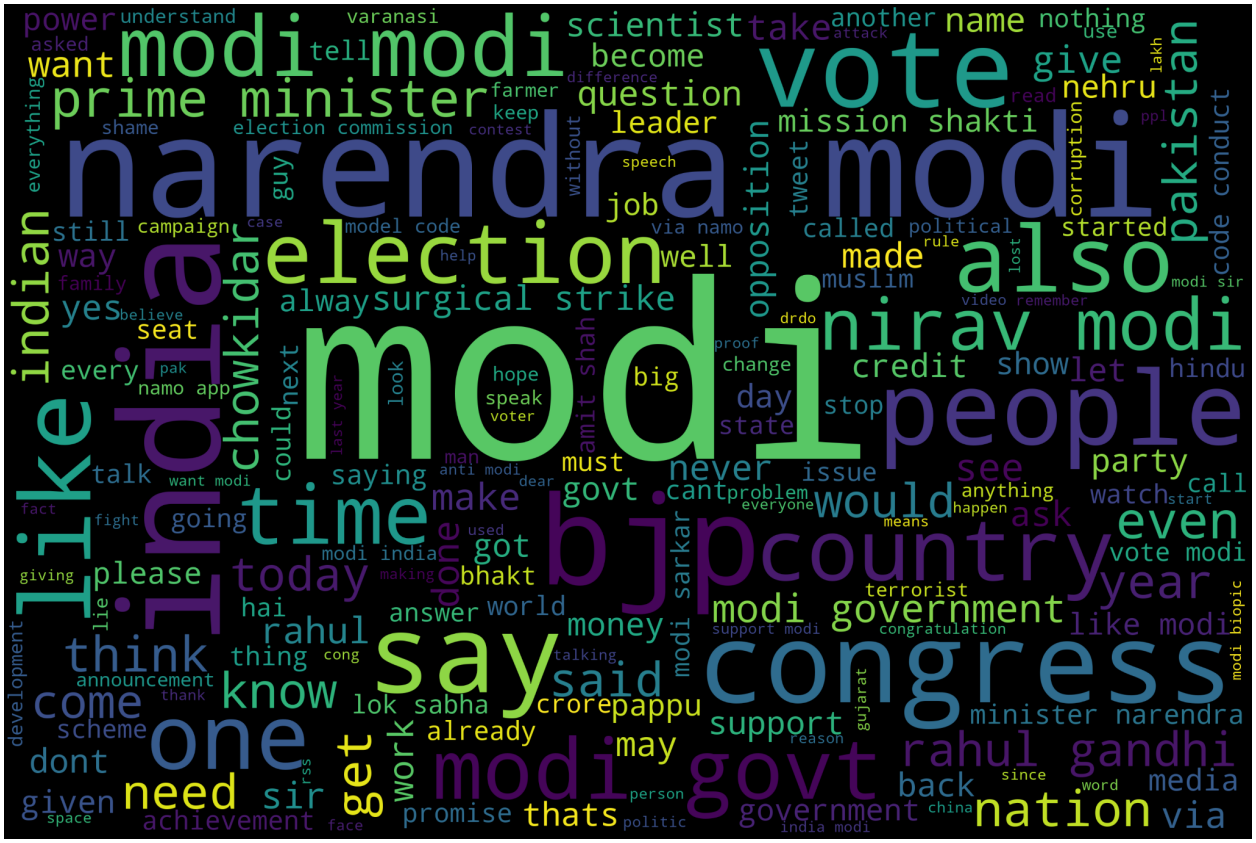
Sentiment Analysis

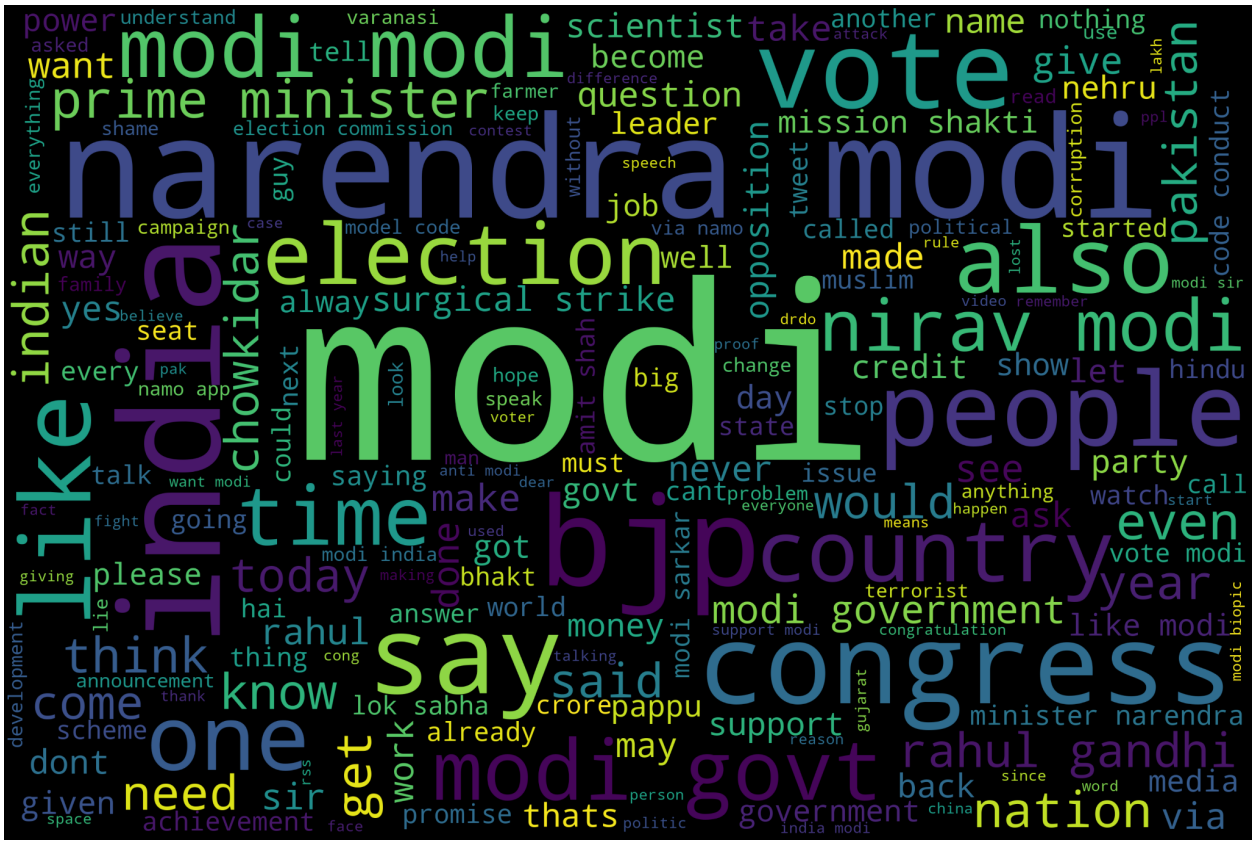
Positive Sentiment Common Words



Negative Sentiment Common Words



Neutral Sentiment Common Words



Confusion Matrix Report

```
confusion_matrix(val['category'], val_preds)
```

[63] ✓ 0.0s

```
array([[ 9572,   264,   855],
       [  198, 16215,   231],
       [  687,   261, 20608]], dtype=int64)
```

In this case:

- First row and column represent the negative class (-1)
- Second row and column represent the neutral class (0)
- Third row and column represent the positive class (1)

Report

	precision	recall	f1-score	support
-1.0	0.92	0.90	0.91	10691
0.0	0.97	0.97	0.97	16644
1.0	0.95	0.96	0.95	21556
accuracy			0.95	48891
macro avg	0.94	0.94	0.94	48891
weighted avg	0.95	0.95	0.95	48891

Report:

- The report includes precision, recall, F1-score metrics, and weighted average for each class (-1.0, 0.0, and 1.0).
- It also shows the support which is the number of samples in each class in the validation set.

Output:

- From the output, we can see that the model has **good** precision, recall, and F1-scores for all classes.
- An overall accuracy of **95%**.
- The macro average F1-score and weighted average F1-score are both **0.94**, indicating a balanced performance across all classes.

Examples (Machine Learning Prediction)

Positive Sentiment

Positive sentiment

```
# example text to predict the sentiment for
text = "I really enjoyed the movie. The acting was great and the plot was interesting."

# transform the text using the same tf-idf vectorizer
text_abs = vec.transform([text])

# predict the sentiment of the text
pred = clf.predict(text_abs)

# print the predicted sentiment
if pred == -1:
    print('Negative')
elif pred == 0:
    print('Neutral')
else:
    print('Positive')
```

[65] ✓ 0.0s

```
Positive
```

Negative Sentiment

Negative sentiment

```
text = "I am feeling really sad and disappointed about the news, it's just heartbreaking. I hope things will get better soon."

# transform the text using the same tf-idf vectorizer
text_abs = vec.transform([text])

# predict the sentiment of the text
pred = clf.predict(text_abs)

# print the predicted sentiment
if pred == -1:
    print('Negative')
elif pred == 0:
    print('Neutral')
else:
    print('Positive')
```

[66] ✓ 0.0s

```
Negative
```

Neutral Sentiment

Neutral sentiment

```
text = "The weather today is partly cloudy with a chance of rain later in the evening."

# transform the text using the same tf-idf vectorizer
text_abs = vec.transform([text])

# predict the sentiment of the text
pred = clf.predict(text_abs)

# print the predicted sentiment
if pred == -1:
    print('Negative')
elif pred == 0:
    print('Neutral')
else:
    print('Positive')
```

✓ 0.0s

```
Neutral
```