Unveiling Sentiments in Political Speeches: Analyzing the Prime Minister's Address" (PM replies to Motion of No Confidence in Lok Sabha, 10 Aug, 2023)

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

In [2]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
file_path = "pm speech.txt"
```

In [4]:

```
with open(file_path, "r", encoding="utf-8") as file:
    speech_text = file.read()
```

In [5]:

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

In [6]:

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\hp5cd\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\hp5cd\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\hp5cd\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Out[6]:

```
True
```

In [7]:

```
speech_text_cleaned = re.sub(r'[^\w\s]', '', speech_text)
speech_text_cleaned = speech_text_cleaned.lower()
words = word_tokenize(speech_text_cleaned)
stop_words = set(stopwords.words("english"))
words_filtered = [word for word in words if word not in stop_words]
```

In [8]:

```
lemmatizer = WordNetLemmatizer()
words_lemmatized = [lemmatizer.lemmatize(word) for word in words_filtered]
```

In [9]:

```
sia = SentimentIntensityAnalyzer()
sentiment_scores = [sia.polarity_scores(word)["compound"] for word in words_lemmatized]
average_sentiment = sum(sentiment_scores) / len(sentiment_scores)
```

In [10]:

```
print('The average sentiment is:', average_sentiment)
```

```
The average sentiment is: 0.014298377028714108
```

In [11]:

```
positive_words = [word for i, word in enumerate(words_filtered) if sentiment_scores[i] >
negative_words = [word for i, word in enumerate(words_filtered) if sentiment_scores[i] <
neutral_words = [word for i, word in enumerate(words_filtered) if sentiment_scores[i] >=
```

In [12]:

```
print('The positive words are:', positive_words)
```

```
The positive words are: ['gratitude', 'trust', 'free', 'trust', 'fulfill',
'dreams', 'trust', 'confidence', 'top', 'freedom', 'fighters', 'ensure',
'peace', 'assure', 'faith', 'commitment', 'party', 'revered', 'confidenc
e', 'gratitude', 'trust', 'confidence', 'strength', 'lucky', 'confidence',
'blessings', 'better', 'important', 'interest', 'party', 'free', 'energy',
'determination', 'huge', 'dreams', 'strengths', 'dreams', 'free', 'courag
e', 'opportunity', 'confidence', 'confidence', 'growth', 'trust', 'fulfil
l', 'dreams', 'marvel', 'helping', 'save', 'helping', 'save', 'helping',
'save', 'trust', 'like', 'wish', 'well', 'best', 'profit', 'increased', 's
uccess', 'growing', 'stronger', 'responsible', 'vision', 'top', 'definit
e', 'confidence', 'top', 'faith', 'like', 'agree', 'peace', 'trusting', 't
rust', 'certain', 'opportunity', 'trust', 'trust', 'confidence', 'help',
'parties', 'faith', 'dwelled', 'fascination', 'freedom', 'fighters', 'dedi
cated', 'party', 'freebies', 'winning', 'assurances', 'interested', 'grea
t', 'confidence', 'honest', 'ensure', 'assure', 'peace', 'assured', 'assur
ed', 'strong', 'responsible', 'emotional', 'attachment', 'rich', 'goods',
'reached', 'like', 'increased', 'honoured', 'awards', 'hero', 'like', 'cel
ebrated', 'faith', 'commitment', 'assure', 'devote', 'party', 'revered',
'certain', 'devoted', 'trust', 'confidence', 'trust', 'trust', 'inspires',
'credited', 'growing', 'trust', 'growth', 'confidence', 'succeeded', 'stro
ng', 'confidence', 'parties', 'best']
```

In [13]:

```
print('The negative words are:', negative_words)
```

```
The negative words are: ['scams', 'poor', 'distrust', 'crimes', 'unaccepta
ble', 'guilty', 'punished', 'pressure', 'stop', 'poor', 'deprived', 'betra
yal', 'disappointed', 'scams', 'stressed', 'unsuccessful', 'poor', 'povert
y', 'poverty', 'poor', 'poor', 'criticizing', 'distrust', 'bad', 'bad', 'c
riticism', 'bad', 'misinformation', 'confuse', 'scam', 'crisis', 'severel
y', 'attacked', 'ills', 'questioned', 'lack', 'poverty', 'hard', 'distrust
ing', 'lack', 'strike', 'enemy', 'ill', 'misinformed', 'insecurity', 'misi
nformed', 'low', 'fool', 'arrogance', 'arrogant', 'contradictions', 'damag
es', 'suffered', 'victims', 'perturbed', 'stuck', 'warned', 'havoc', 'lame
nted', 'reckless', 'pressure', 'violence', 'saddening', 'crimes', 'unaccep
table', 'guilty', 'punished', 'protest', 'failure', 'attack', 'neglect',
'conflict', 'forbidden', 'forbidden', 'loss', 'lack', 'pressure', 'stop',
'worse', 'petty', 'pain', 'suffering']
```

In [14]:

```
print('The neutral words are:', neutral_words)
```

```
The neutral words are: ['come', 'express', 'immense', 'towards', 'ever
y', 'citizen', 'india', 'repeatedly', 'showing', 'government', 'many',
'key', 'legislations', 'get', 'discussion', 'deserved', 'opposition',
'put', 'politics', 'time', 'period', '21st', 'century', 'impact', 'coun
try', 'next', 'thousand', 'years', 'single', 'focus', 'given', 'youth',
'india', 'government', 'today', 'arisen', 'heart', 'opposition', 'abl
e', 'see', 'people', 'steeped', '2028', 'bring', 'motion', 'country',
'among', '3', 'opposition', 'believes', 'changing', 'names', 'cant', 'c
hange', 'work', 'culture', 'founding', 'fathers', 'country', 'always',
'opposed', 'dynasty', 'politics', 'women', 'central', 'government', 'st
ate', 'government', 'work', 'manipur', 'march', 'path', 'development',
'people', 'manipur', 'mothers', 'daughters', 'manipur', 'nation', 'stan
ds', 'house', 'stands', 'government', 'leave', 'stone', 'unturned', 'ma
nipur', 'gets', 'back', 'track', 'development', 'government', 'given',
'first', 'priority', 'development', 'northeast', 'us', 'sabka', 'saat
h', 'sabka', 'vishwas', 'slogan', 'article', 'parliament', 'platform',
'parliament', 'highest', 'body', 'country', 'every', 'second', 'utilize
d', 'country', 'india', 'today', 'crumble', 'india', 'today', 'bend',
'tire', 'prime', 'minister', 'shri', 'narendra', 'modi', 'replied', 'mo
```

In [15]:

```
word_freq_positive = nltk.FreqDist(positive_words)
word_freq_negative = nltk.FreqDist(negative_words)
word_freq_neutral = nltk.FreqDist(neutral_words)
```

In [16]:

```
print('The positive words frequency is:', word_freq_positive)
```

```
The positive words frequency is: <FreqDist with 74 samples and 138 outcome
s>
```

In [17]:

```
print('The negative words frequency is:', word_freq_negative)
```

```
The negative words frequency is: <FreqDist with 61 samples and 82 outcomes
>
```

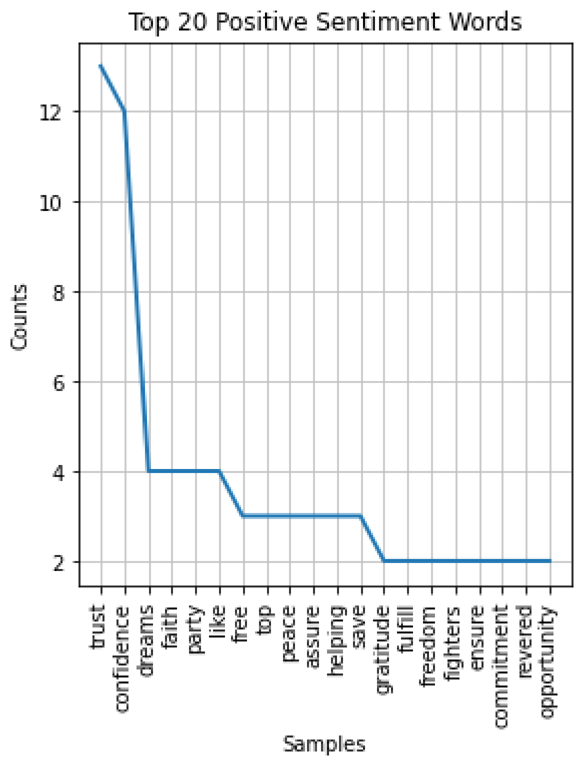
In [18]:

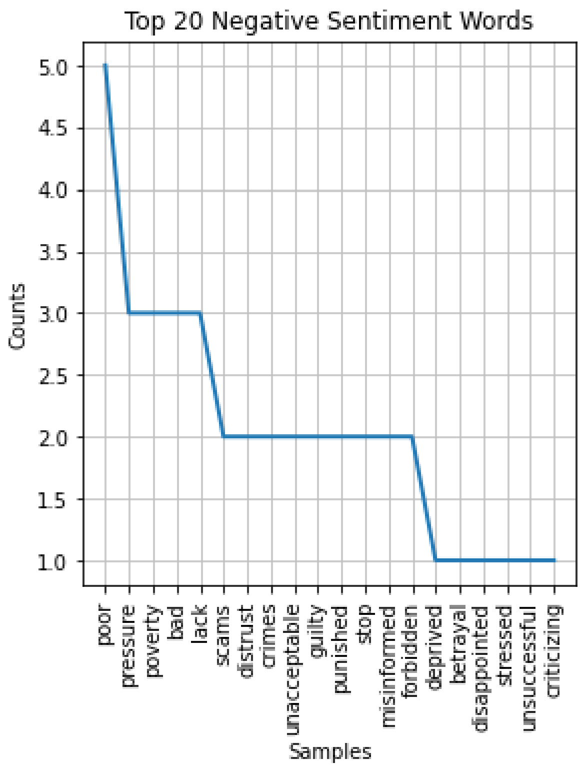
```
print('The neutral words frequency is:', word_freq_neutral)
```

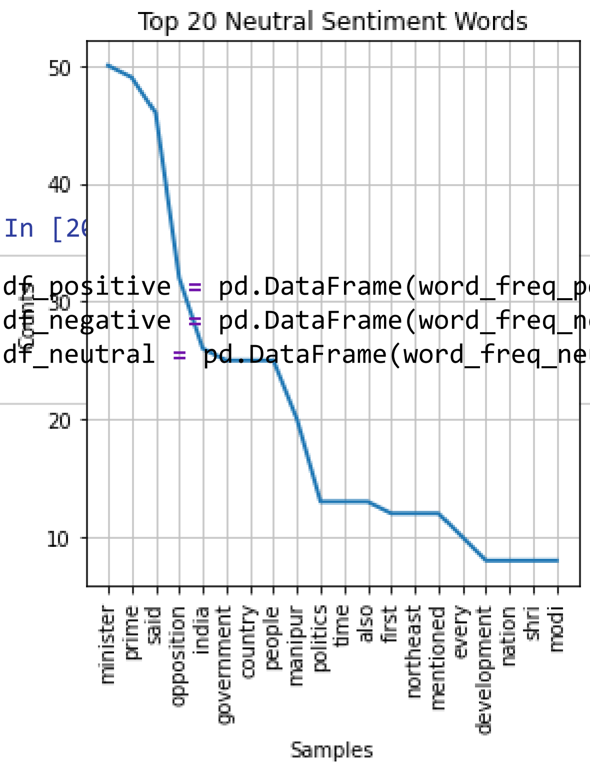
```
The neutral words frequency is: <FreqDist with 658 samples and 1382 outcom
es>
```

In [19]:

```
plt.figure(figsize=(15, 5))

plt.subplot(131)
word_freq_positive.plot(20, title="Top 20 Positive Sentiment Words")

plt.figure(figsize=(15, 5))
plt.subplot(132)
word_freq_negative.plot(20, title="Top 20 Negative Sentiment Words")

plt.figure(figsize=(15, 5))
plt.subplot(133)
word_freq_neutral.plot(20, title="Top 20 Neutral Sentiment Words")

plt.tight_layout()
plt.show()
```







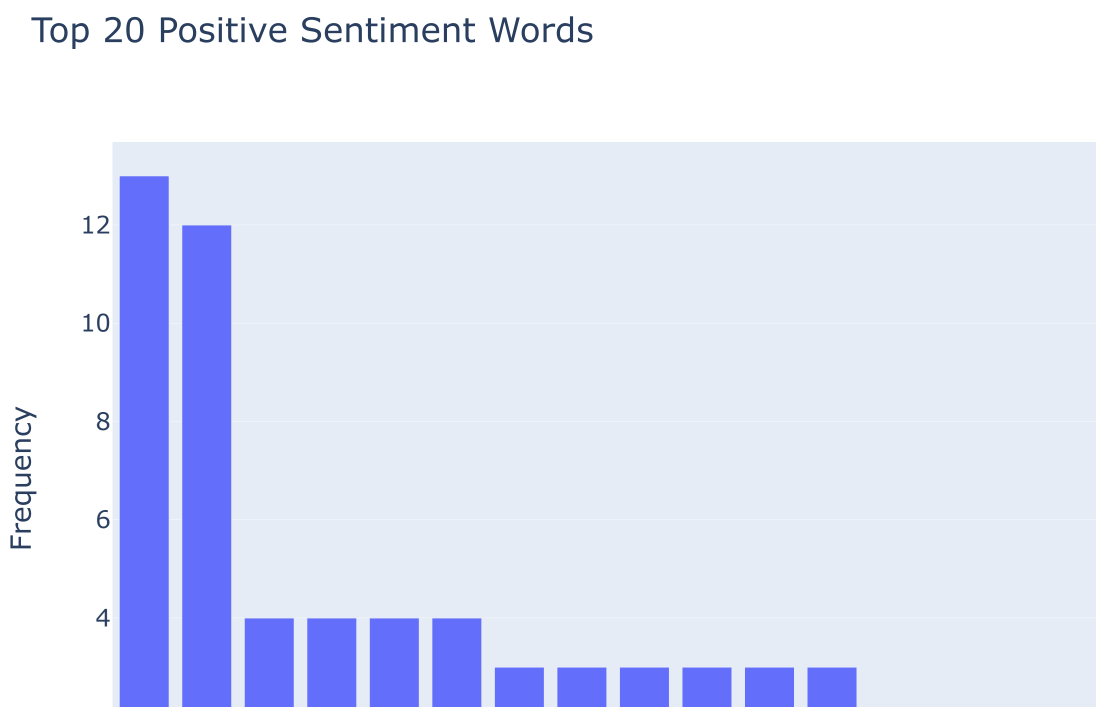
In [2

```
df_positive = pd.DataFrame(word_freq_positive.most_common(20), columns=['Word', 'Frequen
df_negative = pd.DataFrame(word_freq_negative.most_common(20), columns=['Word', 'Frequen
df_neutral = pd.DataFrame(word_freq_neutral.most_common(20), columns=['Word', 'Frequency
```

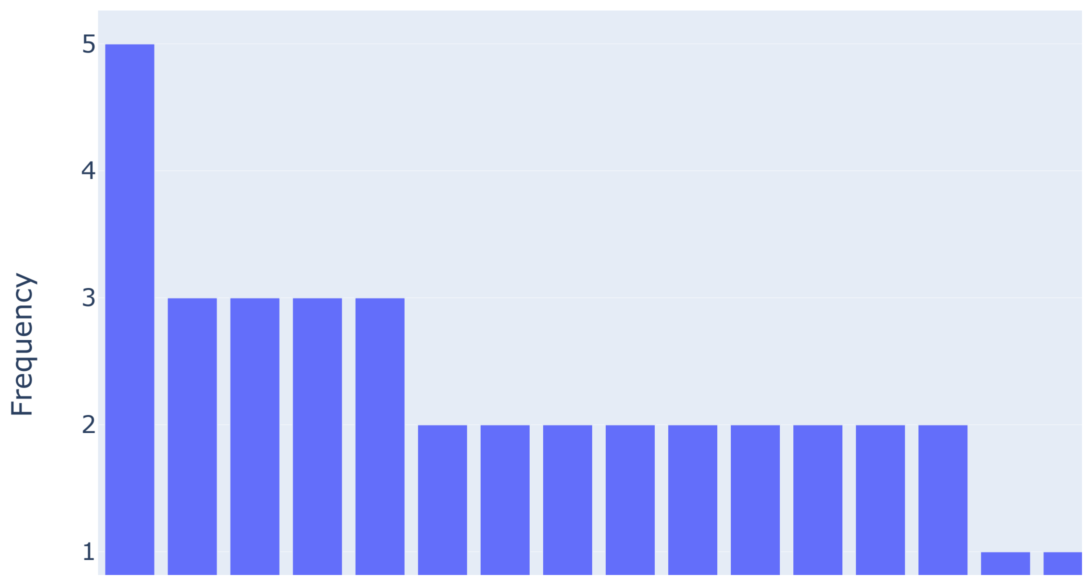
```
<Figure size 432x288 with 0 Axes>
```

In [21]:

```
fig_positive = px.bar(df_positive, x='Word', y='Frequency', title="Top 20 Positive Senti
fig_negative = px.bar(df_negative, x='Word', y='Frequency', title="Top 20 Negative Senti
fig_neutral = px.bar(df_neutral, x='Word', y='Frequency', title="Top 20 Neutral Sentimen

fig_positive.show()
fig_negative.show()
fig_neutral.show()
```



Top 20 Negative Sentiment Words

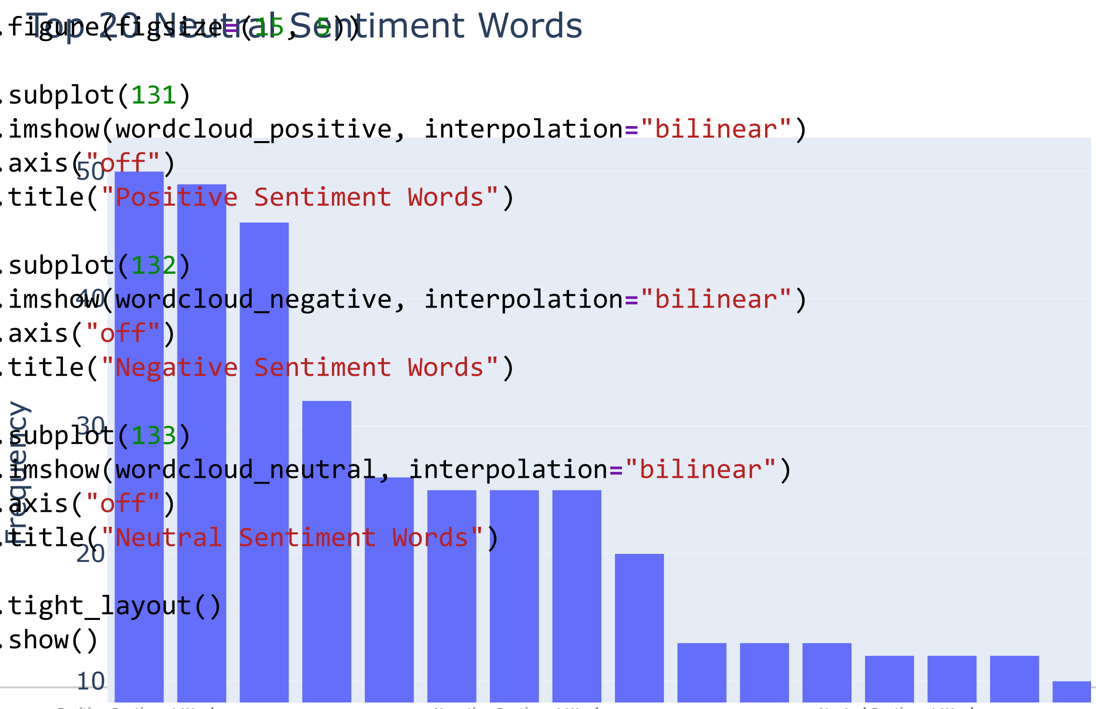


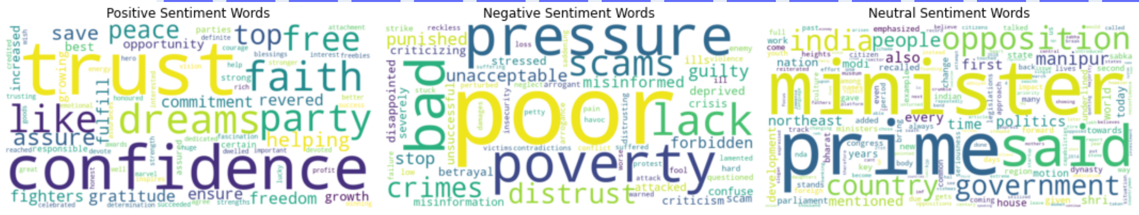
In [22]:

```
wordcloud_positive = WordCloud(width=800, height=400, background_color="white").generate
wordcloud_negative = WordCloud(width=800, height=400, background_color="white").generate
wordcloud_neutral = WordCloud(width=800, height=400, background_color="white").generate_
```

In [23]:

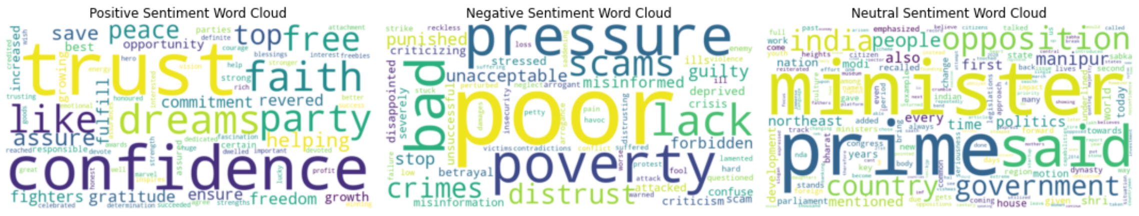
```
plt.figure(figsize=(15, 5))

plt.subplot(131)
plt.imshow(wordcloud_positive, interpolation="bilinear")
plt.axis("off")
plt.title("Positive Sentiment Words")

plt.subplot(132)
plt.imshow(wordcloud_negative, interpolation="bilinear")
plt.axis("off")
plt.title("Negative Sentiment Words")

plt.subplot(133)
plt.imshow(wordcloud_neutral, interpolation="bilinear")
plt.axis("off")
plt.title("Neutral Sentiment Words")

plt.tight_layout()
plt.show()
```





In [24]:

```
plt.figure(figsize=(15, 10))

plt.subplot(131)
plt.imshow(wordcloud_positive, interpolation="bilinear")
plt.title("Positive Sentiment Word Cloud")
plt.axis("off")

plt.subplot(132)
plt.imshow(wordcloud_negative, interpolation="bilinear")
plt.title("Negative Sentiment Word Cloud")
plt.axis("off")

plt.subplot(133)
plt.imshow(wordcloud_neutral, interpolation="bilinear")
plt.title("Neutral Sentiment Word Cloud")
plt.axis("off")

plt.tight_layout()
plt.show()
```



In [25]:

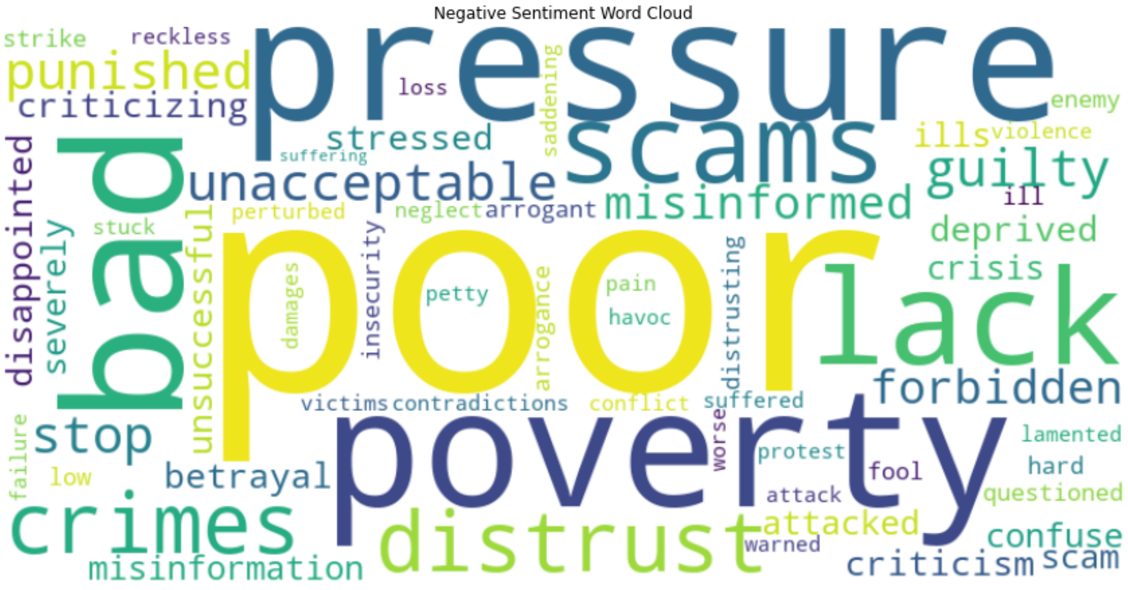
```python
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud_positive, interpolation="bilinear")
plt.title("Positive Sentiment Word Cloud")
plt.axis("off")
plt.show()
```

Positive Sentiment Word Cloud

credited
increased
wish
save
peace
parties
definite
top
free
attachment
best
opportunity
courage
blessings
interest
freebies
trust
hero
growing
energy
vision
help
stronger
strong
faith
rich
fulfill
interested
trusting
honoured
commitment
revered
better
success
like
emotional
goods
assurances
dreams
party
assure
dedicated
fascination
helping
reached
responsible
awards
devote
strength
assured
huge
certain
dwelled
important
devoted
confidence
great
well
honest
marvel
inspires
lucky
profit
fighters
gratitude
ensure
freedom
growth
celebrated
determination
succeeded
agree
strengths
winning

In [26]:

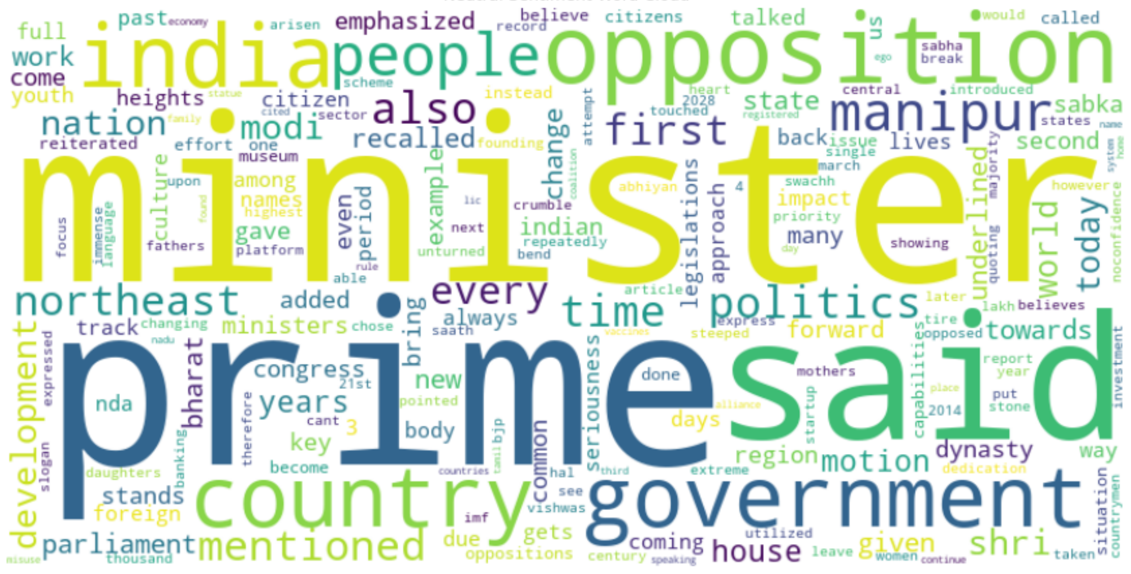
```python
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud_negative, interpolation="bilinear")
plt.title("Negative Sentiment Word Cloud")
plt.axis("off")
plt.show()
```


Negative Sentiment Word Cloud

In [27]:

```
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud_neutral, interpolation="bilinear")
plt.title("Neutral Sentiment Word Cloud")
plt.axis("off")
plt.show()
```

Neutral Sentiment Word Cloud



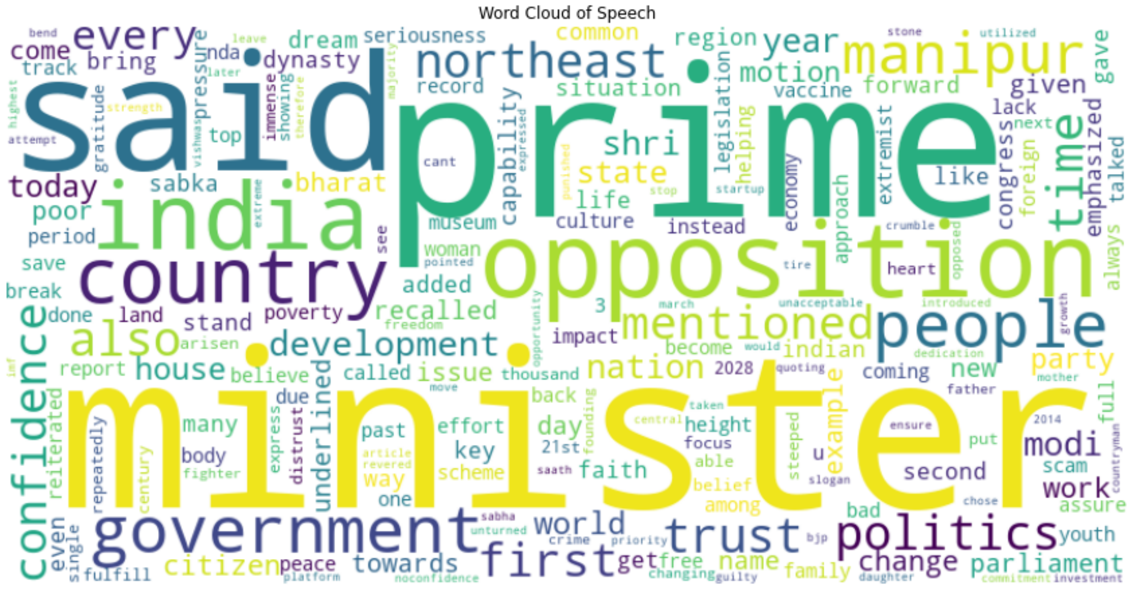
In [28]:

```
word_freq = nltk.FreqDist(words_lemmatized)
wordcloud = WordCloud(width=800, height=400, background_color="white").generate_from_fre
```

In [29]:

```
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Word Cloud of Speech")
plt.show()
```

Word Cloud of Speech



In [30]:

```
total_words = len(words_filtered)
positive_percentage = (len(positive_words) / total_words) * 100
negative_percentage = (len(negative_words) / total_words) * 100
neutral_percentage = (len(neutral_words) / total_words) * 100

print("Positive Sentiment Percentage:", positive_percentage)
print("Negative Sentiment Percentage:", negative_percentage)
print("Neutral Sentiment Percentage:", neutral_percentage)
```

```
Positive Sentiment Percentage: 8.614232209737828
Negative Sentiment Percentage: 5.118601747815231
Neutral Sentiment Percentage: 86.26716604244695
```

In [31]:

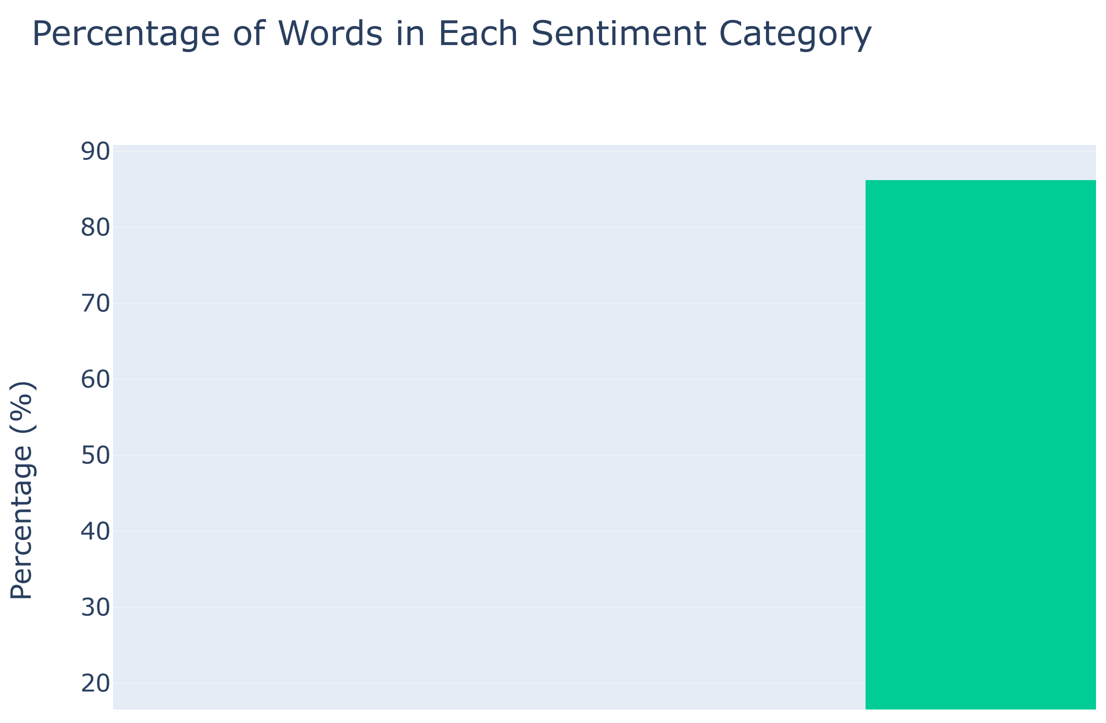
```
data = {'Sentiment': ['Positive', 'Negative', 'Neutral'],
        'Percentage': [positive_percentage, negative_percentage, neutral_percentage]}
```

In [32]:

```
df_percentages = pd.DataFrame(data)
```

In [33]:

```
fig = px.bar(df_percentages, x='Sentiment', y='Percentage', color='Sentiment',
             labels={'Sentiment': 'Sentiment Category', 'Percentage': 'Percentage (%)'},
             title='Percentage of Words in Each Sentiment Category')
fig.show()
```



In [34]:

```
import gensim
from gensim import corpora
from gensim.models.ldamodel import LdaModel
```

In [35]:

```
dictionary = corpora.Dictionary([words_filtered])
```

In [36]:

```
corpus = [dictionary.doc2bow(words_filtered)]
```

In [37]:

```
lda_model = LdaModel(corpus, num_topics=5, id2word=dictionary, passes=15)

topics = lda_model.print_topics(num_words=5)
for topic in topics:
    print(topic)
```

```
(0, '0.001*"prime" + 0.001*"said" + 0.001*"minister" + 0.001*"opposition"
+ 0.001*"government"')
(1, '0.001*"prime" + 0.001*"said" + 0.001*"minister" + 0.001*"opposition"
+ 0.001*"government"')
(2, '0.001*"minister" + 0.001*"prime" + 0.001*"opposition" + 0.001*"said"
+ 0.001*"india"')
(3, '0.001*"minister" + 0.001*"said" + 0.001*"prime" + 0.001*"government"
+ 0.001*"opposition"')
(4, '0.029*"minister" + 0.028*"prime" + 0.026*"said" + 0.018*"opposition"
+ 0.015*"india"')
```

In [38]:

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp(speech_text)
entities = [(ent.text, ent.label_) for ent in doc.ents]

for entity, label in entities:
    print(f"Entity: {entity}, Label: {label}")
```

```
Entity: India, Label: GPE
Entity: the 21st century, Label: DATE
Entity: the next thousand years, Label: DATE
Entity: India, Label: GPE
Entity: 2028, Label: DATE
Entity: 3, Label: CARDINAL
Entity: the Central Government, Label: ORG
Entity: the State Government, Label: ORG
Entity: House, Label: ORG
Entity: first, Label: ORDINAL
Entity: Northeast, Label: LOC
Entity: Parliament, Label: ORG
Entity: Party, Label: ORG
Entity: Parliament, Label: ORG
Entity: second, Label: ORDINAL
Entity: The India of today, Label: WORK_OF_ART
Entity: India, Label: GPE
Entity: Shri Narendra Modi, Label: PERSON
Entity: the Motion of No Confidence, Label: ORG
```

In [39]:

```python
from keybert import KeyBERT
kw_extractor = KeyBERT()
keywords = kw_extractor.extract_keywords(speech_text)
for keyword in keywords:
    print(keyword[0])
```

```
Downloading                                   1.18k/1.18k [00:00<00:00,
(…)e9125/.gitattributes: 100%                 40.0kB/s]

Downloading                                   190/190 [00:00<00:00,
(…)_Pooling/config.json: 100%                 6.44kB/s]

Downloading                                   10.6k/10.6k
(…)7e55de9125/README.md:                      [00:00<00:00, 289kB/s]
100%

Downloading                                   612/612 [00:00<00:00,
(…)55de9125/config.json: 100%                 19.1kB/s]

Downloading                                   116/116 [00:00<00:00,
(…)ce_transformers.json: 100%                 4.24kB/s]

Downloading                                   39.3k/39.3k
(…)125/data_config.json: 100%                 [00:00<00:00, 206kB/s]

Downloading                                   90.9M/90.9M [00:08<00:00,
pytorch_model.bin: 100%                       12.2MB/s]

Downloading                                   53.0/53.0 [00:00<00:00,
(…)nce_bert_config.json: 100%                 2.33kB/s]

Downloading                                   112/112 [00:00<00:00,
(…)cial_tokens_map.json: 100%                 7.07kB/s]

Downloading                                   466k/466k [00:00<00:00,
(…)e9125/tokenizer.json: 100%                 806kB/s]

Downloading                                   350/350 [00:00<00:00,
(…)okenizer_config.json: 100%                 8.68kB/s]

Downloading                                   13.2k/13.2k
(…)9125/train_script.py: 100%                 [00:00<00:00, 411kB/s]
```

```
Downloading                                  232k/232k [00:00<00:00,
(…)7e55de9125/vocab.txt: 100%                13.5MB/s]

Downloading                                  349/349 [00:00<00:00,
(…)5de9125/modules.json: 100%                32.2kB/s]
```

```
manipur
nehru
bjp
rajya
gandhi
```

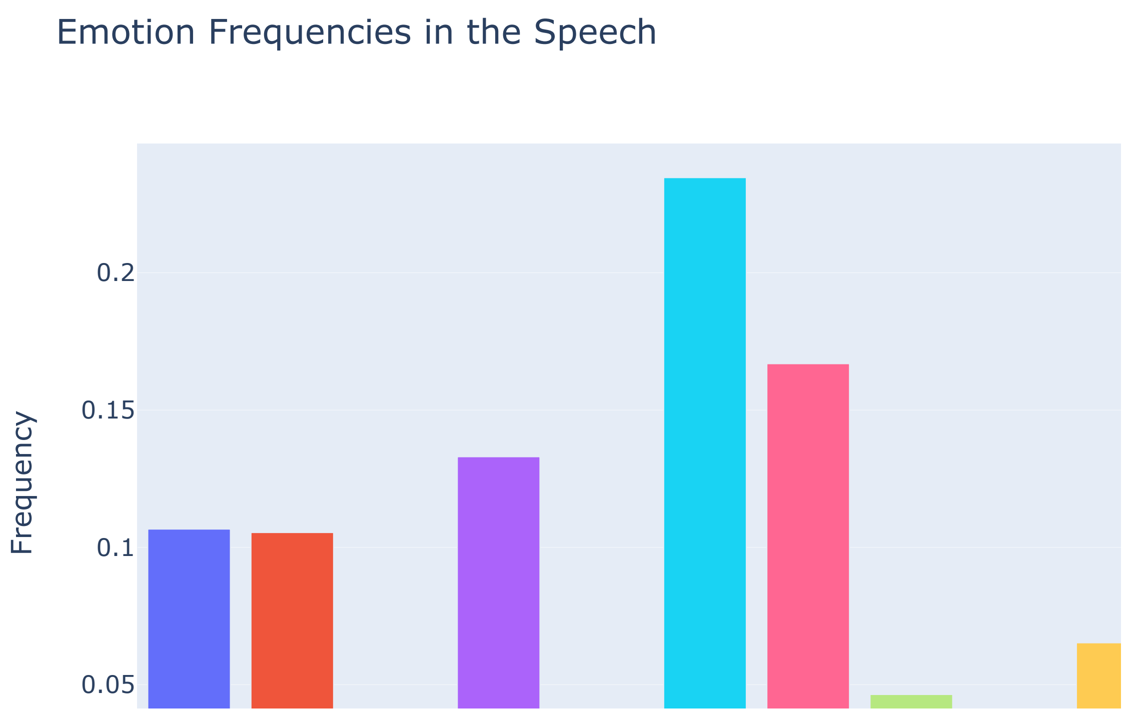
In [40]:

```python
from nrclex import NRCLex
text_emotion = NRCLex(speech_text_cleaned)
emotions = text_emotion.affect_frequencies
for emotion, frequency in emotions.items():
    print(f"Emotion: {emotion}, Frequency: {frequency}")
```

```
Emotion: fear, Frequency: 0.10664993726474278
Emotion: anger, Frequency: 0.1053952321204517
Emotion: anticip, Frequency: 0.0
Emotion: trust, Frequency: 0.1329987452948557
Emotion: surprise, Frequency: 0.02258469259723965
Emotion: positive, Frequency: 0.23462986198243413
Emotion: negative, Frequency: 0.1668757841907152
Emotion: sadness, Frequency: 0.04642409033877039
Emotion: disgust, Frequency: 0.033877038895859475
Emotion: joy, Frequency: 0.06524466750313676
Emotion: anticipation, Frequency: 0.08531994981179424
```

In [41]:

```
data = {'Emotion': [], 'Frequency': []}
for emotion, frequency in emotions.items():
    data['Emotion'].append(emotion)
    data['Frequency'].append(frequency)

df_emotions = pd.DataFrame(data)

fig = px.bar(df_emotions, x='Emotion', y='Frequency', color='Emotion',
             labels={'Emotion': 'Emotion', 'Frequency': 'Frequency'},
             title='Emotion Frequencies in the Speech')
fig.show()
```



In [42]:

```
import textstat

flesch_score = textstat.flesch_reading_ease(speech_text_cleaned)
flesch_grade = textstat.flesch_kincaid_grade(speech_text_cleaned)
smog_index = textstat.smog_index(speech_text_cleaned)

print(f"Flesch Reading Ease Score: {flesch_score}")
print(f"Flesch-Kincaid Grade Level: {flesch_grade}")
print(f"SMOG Index: {smog_index}")
```

```
Flesch Reading Ease Score: -3018.87
Flesch-Kincaid Grade Level: 1192.8
SMOG Index: 0.0
```

In [43]:

```
from nltk.collocations import BigramAssocMeasures, BigramCollocationFinder

tokens = nltk.word_tokenize(speech_text_cleaned)

bigram_measures = BigramAssocMeasures()
finder = BigramCollocationFinder.from_words(tokens)

pmi_scores = finder.score_ngrams(bigram_measures.pmi)

for bigram, pmi in pmi_scores[:10]:
    print(f"Bigram: {bigram}, PMI: {pmi}")
```

```
Bigram: ('135', 'crore'), PMI: 11.576484346796851
Bigram: ('400', 'night'), PMI: 11.576484346796851
Bigram: ('5', 'economies'), PMI: 11.576484346796851
Bigram: ('50000', 'per'), PMI: 11.576484346796851
Bigram: ('account', 'yoga'), PMI: 11.576484346796851
Bigram: ('air', 'travel'), PMI: 11.576484346796851
Bigram: ('almost', 'eradicated'), PMI: 11.576484346796851
Bigram: ('arunachal', 'pradesh'), PMI: 11.576484346796851
Bigram: ('aspirations', 'whatever'), PMI: 11.576484346796851
Bigram: ('azad', 'hind'), PMI: 11.576484346796851
```