Analysing a sample of speeches by Modi

Text Analysis is the term describing the very process of computational analysis of texts. while. Text Analytics involves a set of techniques and approaches towards bringing textual content to a point where it is represented as data and then mined for insights/trends/patterns.

Political scientists have applied automated content analysis across a diverse set of texts. ... Exploring large amounts of text data and assigning text to categories is the most common use of text analysis software in political science.

Natural-language processing (NLP) is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to fruitfully process large amounts of natural language data (wikipedia).

This rapidly improving area of artificial intelligence covers tasks such as speech recognition, natural-language understanding, and natural language generation.

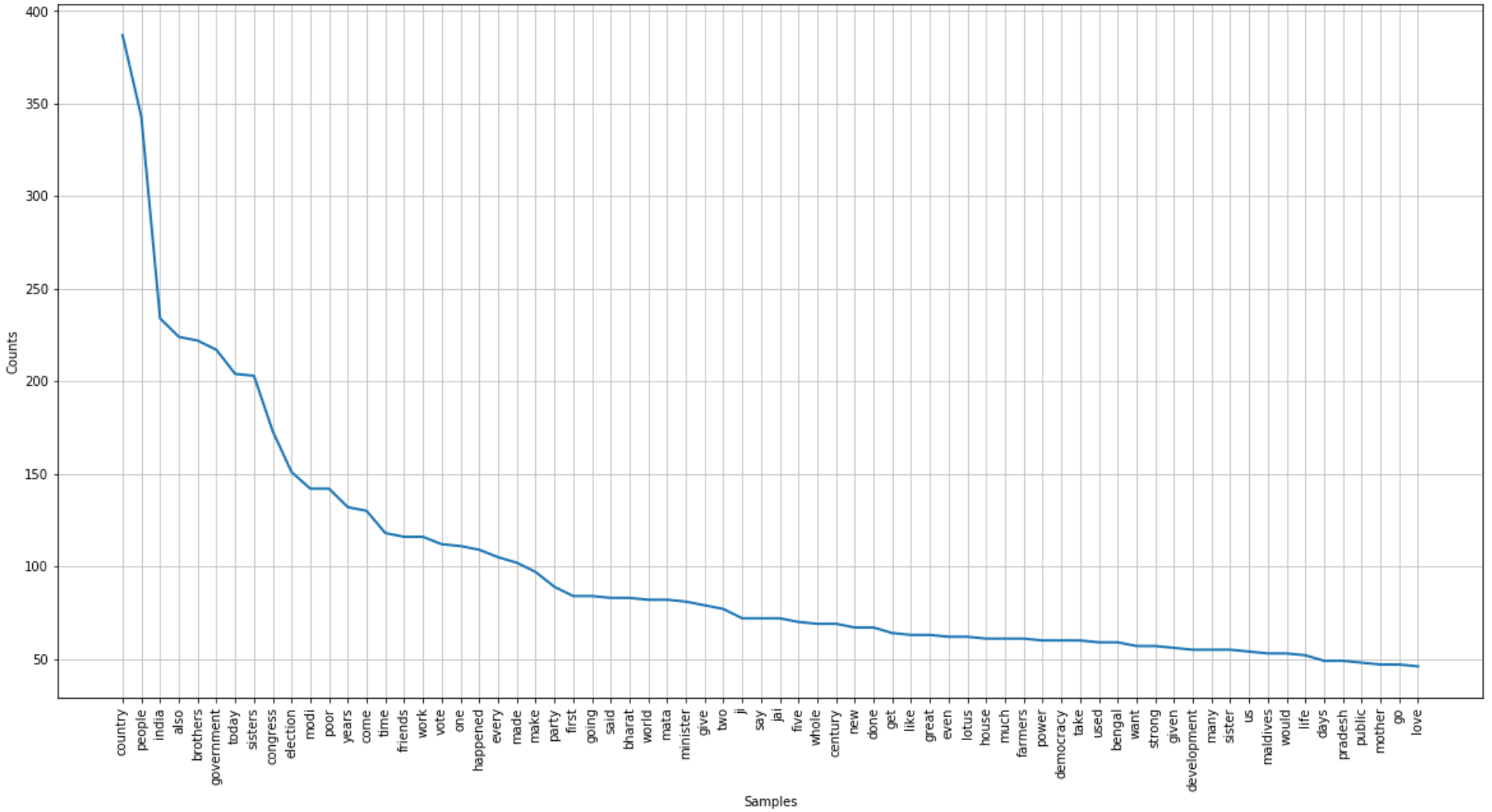
I am going to be using the Natural Language Toolkit (NLTK) which is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.

P.S. : The speeches were in Hindi(mostly), translated to English by Google Translate. I don't take any responsibility for any discrepancy created by any of the wrongly translated data.

In [1]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
import numpy as np
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from collections import Counter
import os
import string
import argparse
import operator
```

In [2]:

```
log = open("text.txt", "r")

#print(log.read())

tokenized_word = word_tokenize(log.read().lower())
#print(tokenized_word)
tokenized_word = [w.replace('elections', 'election') for w in tokenized_word]

from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))
items = [',', '.','...','-',':',';','?',"'s"]

l = list(stop_words)

for x in items:
    l.append(x)


filtered_sentence = []
for w in tokenized_word:
    if w not in l:
        filtered_sentence.append(w)


#print(filtered_sentence)

fdist = FreqDist(filtered_sentence)
plt.figure(figsize=(20,10))
fdist.plot(70,cumulative=False)
plt.show()
```

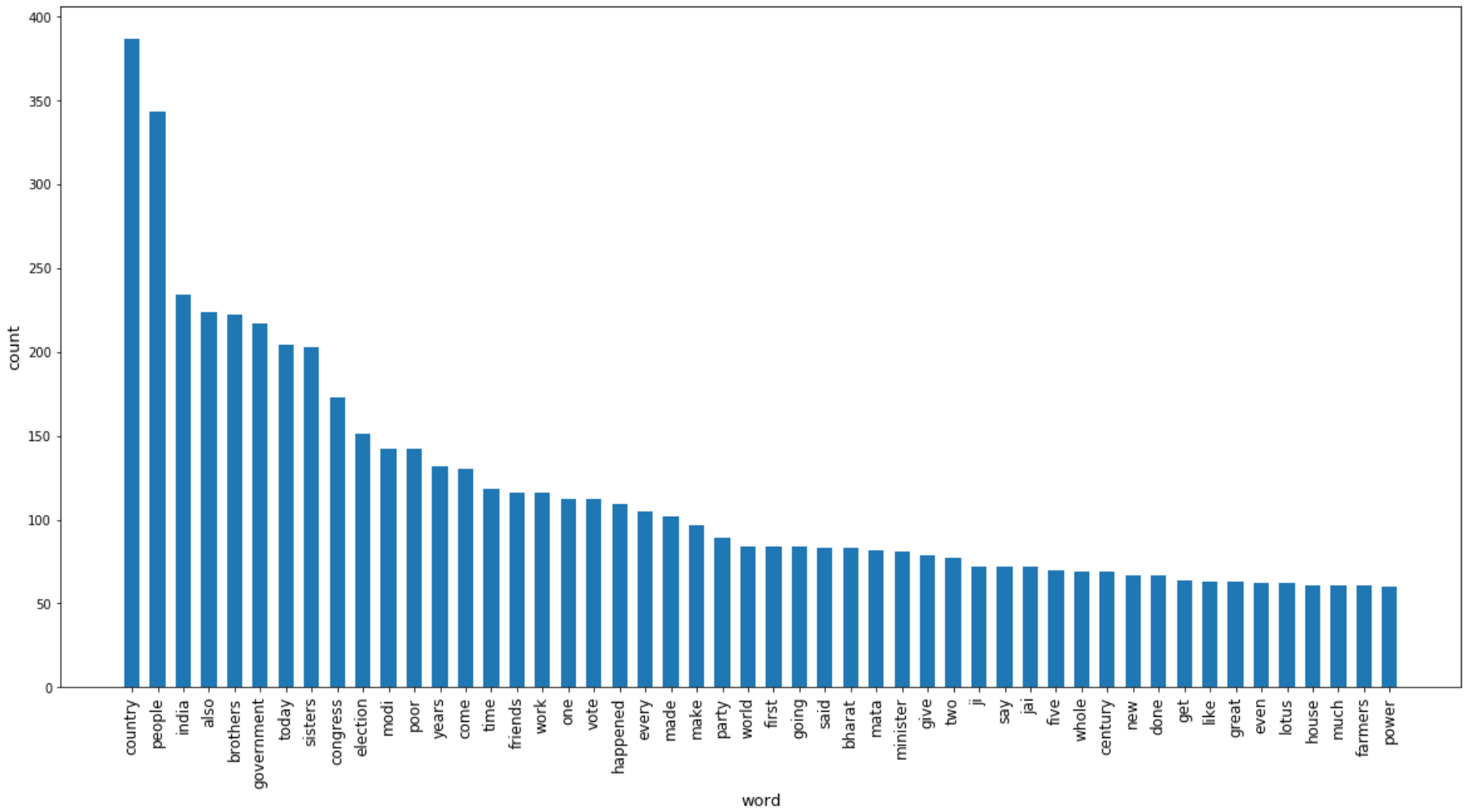


In [3]:
```
# adding thhh enew data in a new file

data = ' '.join(filtered_sentence)

fout = open('text_new.txt', "a")
fout.write(data)
fout.close()
```

In [4]:

```
def main():

    parser = argparse.ArgumentParser(description=__doc__, formatter_class=argparse.RawDescriptionHelpFormatter)
    parser.add_argument('-f','--filepath',dest='filepath',metavar='file path',help='Path to text input file to be anal
ysed.', required=True)
    parser.add_argument('-n','--number',dest='number',metavar='number',help='Most frequent n words will be displayed a
nd plotted.', required=False, default=50, type=int)
    args = parser.parse_args()

    # Path to text file to analyse
    rawfilepath = args.filepath

    # Print a histogram containing the top N words, and print them and their counts.
    top_n = args.number

    # Load the file
    filepath = os.path.normpath(os.path.join(rawfilepath))
    file = open('text_new.txt')

    # Parse as a list, removing lines
    content_sublists = [line.split(',') for line in file.readlines()]

    # Parse into a single list (from a list of lists)
    content_list = [item for sublist in content_sublists for item in sublist]

    # Remove whitespace so we can concatenate appropriately, and unify case
    content_list_strip = [str.strip().lower() for str in content_list]

    # Concatenate strings into a single string
    content_concat = ' '.join(content_list_strip)

    # Remove punctuation and new lines
    punct = set(string.punctuation)
    unpunct_content = ''.join(x for x in content_concat if x not in punct)

    # Split string into list of strings, again
    word_list = unpunct_content.split()


    counts_all = Counter(word_list)
    words, count_values = zip(*counts_all.items())
```

```
    values_sorted, words_sorted = zip(*sorted(zip(count_values, words), key=operator.itemgetter(0), reverse=True))

    # Top N
    words_sorted_top = words_sorted[0:top_n]
    values_sorted_top = values_sorted[0:top_n]


    # Histogram

    # Make xticklabels comprehensible by matplotlib
    xticklabels = str(list(words_sorted_top)).split()
    # Remove the single quotes, commas and enclosing square brackets
    xtlabs = [xstr.replace("'","").replace(",","").replace("]","").replace("[","") for xstr in xticklabels]

    indices = np.arange(len(words_sorted_top))
    width = 0.6
    fig = plt.figure(figsize=(20,10))
    fig.suptitle('Word frequency histogram, top {0}'.format(top_n), fontsize=25)
    plt.xlabel('word', fontsize=13)
    plt.ylabel('count', fontsize=13)
    plt.bar(indices, values_sorted_top, width)
    plt.xticks(indices, xtlabs, rotation='vertical', fontsize=12)
    plt.show()

if __name__ == '__main__':
    main()
```


Word frequency histogram, top 50
count
word
0
50
100
150
200
250
300
350
400
country
people
india
also
brothers
government
today
sisters
congress
election
modi
poor
years
come
time
friends
work
one
vote
happened
every
made
make
party
world
first
going
said
bharat
mata
minister
give
two
ji
say
jai
five
whole
century
new
done
get
like
great
even
lotus
house
much
farmers
power

In [5]:
```
# Generaitng a wordcloud for better understanding and for usage in political analysis:


wordcloud = WordCloud(max_font_size=70, max_words=150, background_color="white", width = 600, height=400).generate(' '
.join(filtered_sentence))
plt.figure(figsize=(12,8))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```