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).

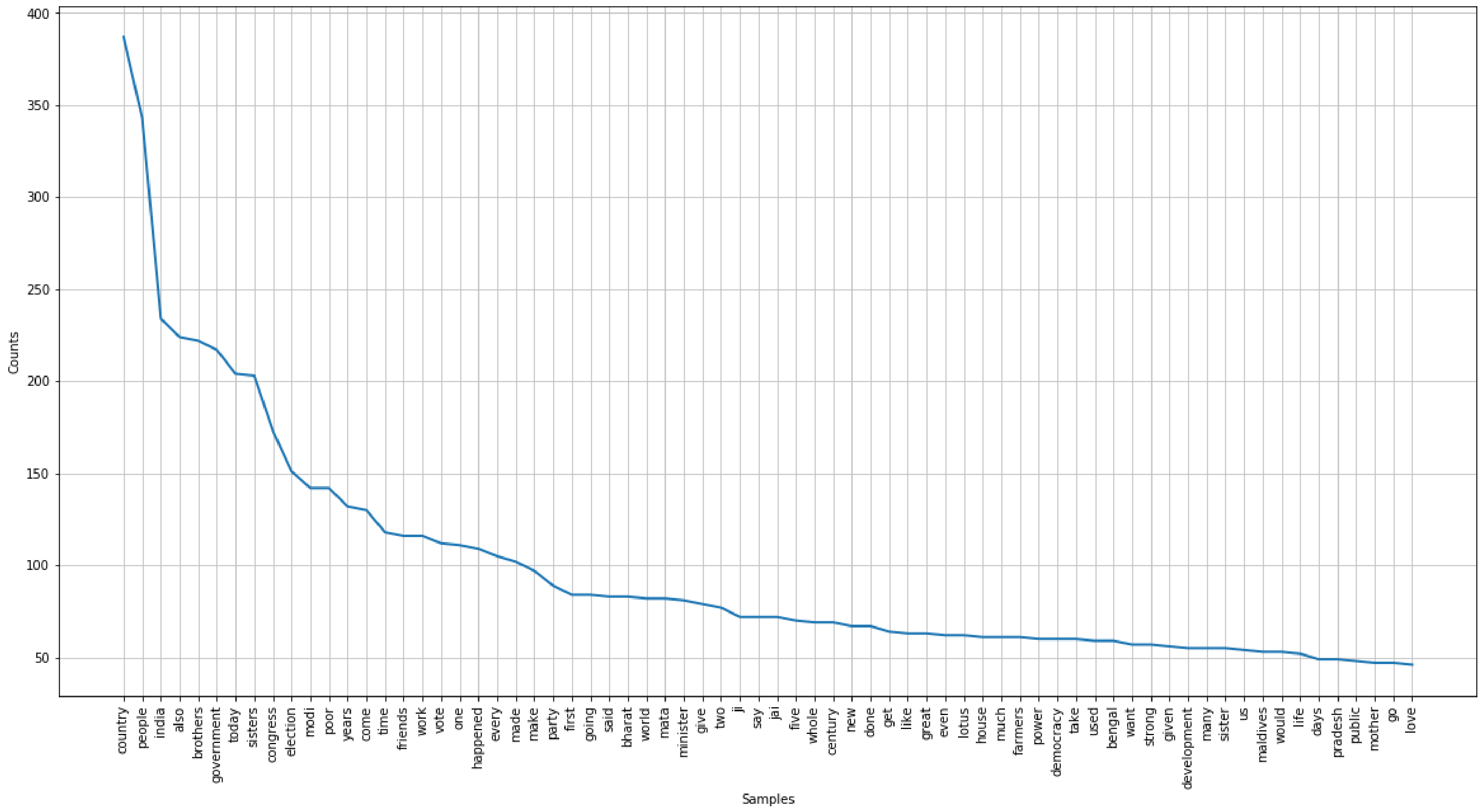
This rapidly improving area of artificial intelligence covers tasks such as speech recognition, natural-language understanding, and natural language generation.

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()
```

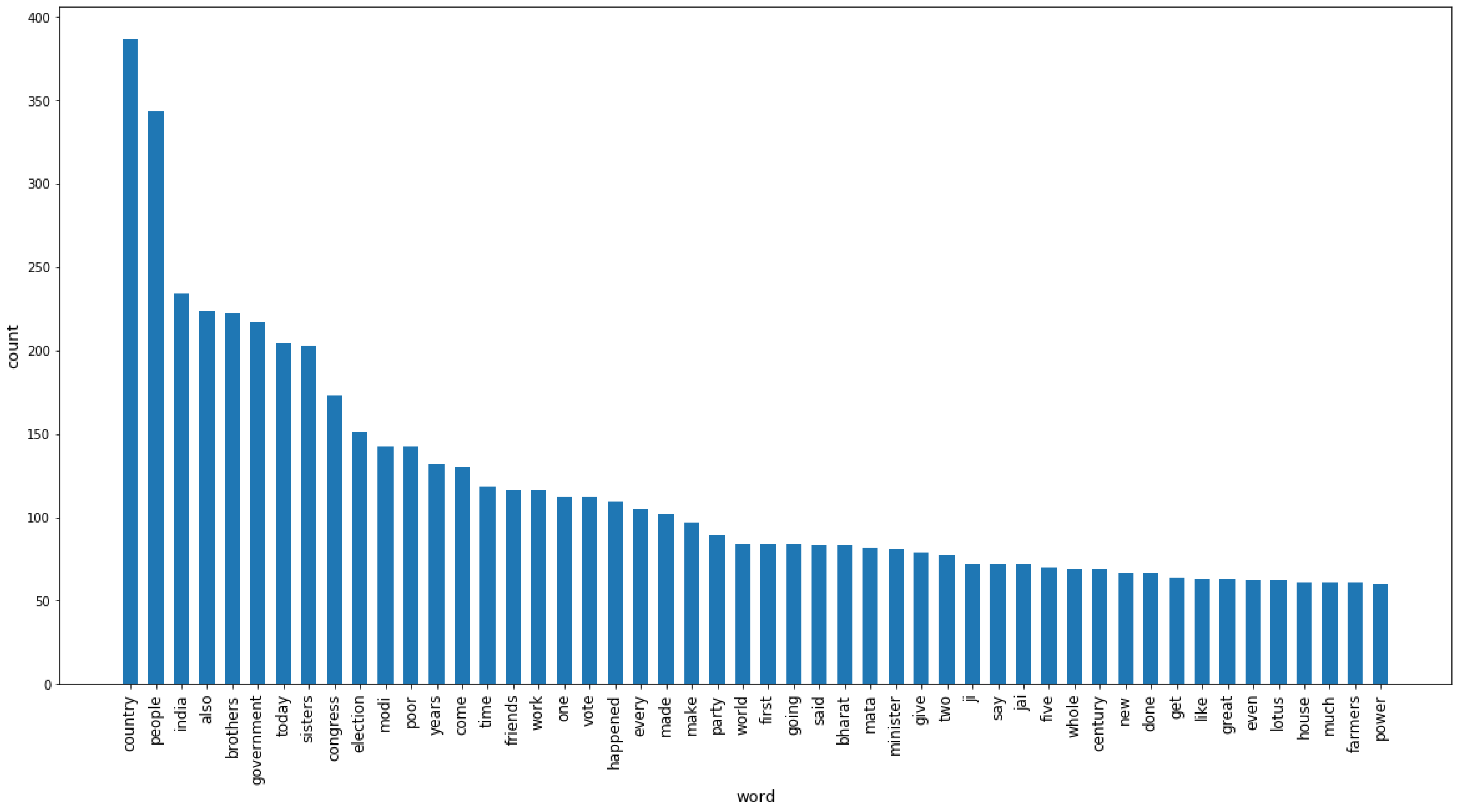


In [3]:
```
data = ' '.join(filtered_sentence)

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()

            # Perform count
            counts_all = Counter(word_list)
```

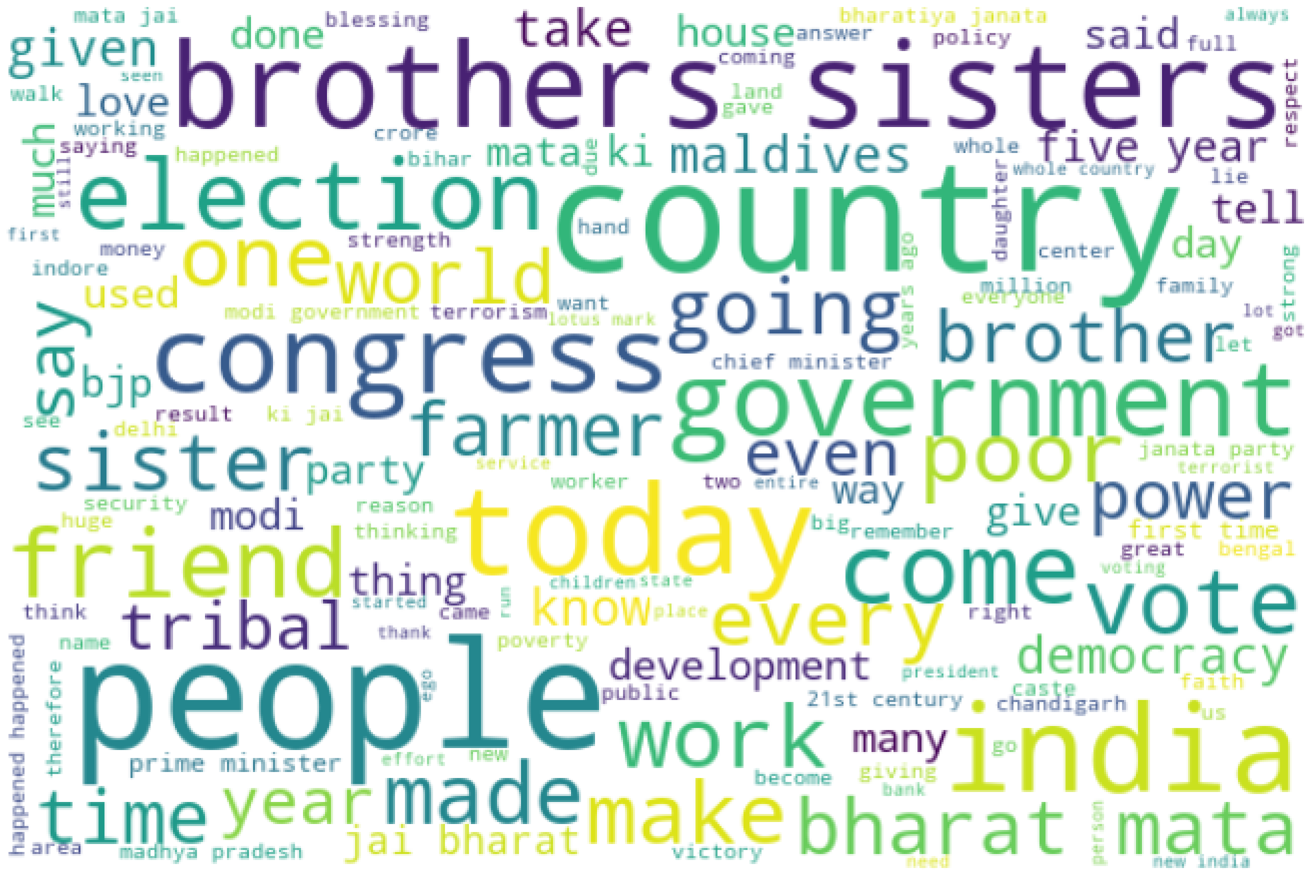
```python
    words, count_values = zip(*counts_all.items())

    # Sort both lists by frequency in values (Schwartzian transform) - thanks, http://stackoverflow.com/questions/9543
211/sorting-a-list-in-python-using-the-result-from-sorting-another-list
    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]
    '''
    print("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -")
    print("{0} unique words identified in the text file, {1}".format(len(values_sorted), filepath))
    print("The top {0} words are: \n{1}".format(top_n, words_sorted_top))
    print("... their respective frequencies: \n{0}".format(values_sorted_top))
    print("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -")
    # Pandas DataFrame just for visualisation
    df = pd.DataFrame({'count': values_sorted_top, 'word': words_sorted_top})
    print("{0}".format(df))
    sys.stdout.flush()
    '''
    # 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



In [5]:

```python
# Display the generated image:
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()
```


mata jai
blessing
take
house
bharatiya janata
always
given
done
said
answer
policy
full
brothers
sisters
coming
seen
walk
love
land
gave
respect
working
crore
saying
happened
bihar
mata
ki
maldives
whole
five year
whole country
much
still
election
country
lie
hand
tell
first
strength
ago
daughter
center
day
money
one
world
indore
used
million
everyone
family
strong
modi government
terrorism
want
lotus mark
going
years
lot
say
congress
brother
let
got
chief minister
bjp
government
result
ki jai
farmer
see
delhi
sister
even
poor
janata party
party
service
terrorist
worker
two
entire
way
power
security
modi
reason
give
huge
big
remember
first time
thinking
great
bengal
friend
today
come
voting
thing
children
state
started
run
think
tribal
came
know
place
every
right
vote
thank
name
poverty
development
president
democracy
faith
caste
ego
public
21st century
chandigarh
us
happened happened
therefore
people
work
many
india
go
prime minister
effort
new
giving
become
bank
year
made
make
time
bharat
mata
person
area
madhya pradesh
jai bharat
victory
need
new india