NLP Assignment 1- Text processing of a book

Team Members

1. Himanshu Daga (16ucc037)
2. Prabhat Sharma (16ucc067)
3. Ujjwal Madan (16ucs204)

To Do

```
Reference Book: Pride and Prejudice (2nd most popular book on Gutenberg)
Reference URL: http://www.gutenberg.org/files/1342/1342-0.txt
Targets:
1. Import the text
2. Perform simple text pre-processing steps and tokenize the text
3. Analyze the frequency distribution of tokens
4. Create  a Word Cloud including stopwords
5. Remove the stopwords and then create a word cloud - what's the difference it giv
es?
6. Evaluate the relationship between word length and frequency - what's your resul
t?
7. Do PoS Tagging using some known corpus and get the distribution of various tags
```

1. Import the text book

In [14]:
```
# Get the text file as a string.
import urllib.request #, urllib.parse, urllib.error
import pandas as pd
import re
import string
import matplotlib.pyplot as plt
url='http://www.gutenberg.org/files/1342/1342-0.txt'
fhand = urllib.request.urlopen(url)

data=''
# read data line by line and decode it to utf8 format
for line in fhand:
    data += line.decode().strip() + ' '
```

In [15]:
```
data1=data.split('.')
```

2 Pre-processing & Tokenisation

2.1 Get content of the book from complete text

There are extra headers and footers by Gutenberg which are not part of actual book and hence they should be removed

Content of the book starts with-

"*** START OF THIS PROJECT GUTENBERG EBOOK PRIDE AND PREJUDICE ***"

Content of the book ends with-

"* END OF THIS PROJECT GUTENBERG EBOOK PRIDE AND PREJUDICE *"

In [16]:
```
# finding book's content
start_phrase = "*** START OF THIS PROJECT GUTENBERG EBOOK PRIDE AND PREJUDICE
***"
start_idx = data.find(start_phrase)
end_phrase = "*** END OF THIS PROJECT GUTENBERG EBOOK PRIDE AND PREJUDICE ***"
end_idx = data.find(end_phrase)
book = data[start_idx + len(start_phrase): end_idx ]
```

In [17]:
```
book = book[len("    Produced by Anonymous Volunteers    "):].strip()
book[:1000]
```

Out[17]:
```
'PRIDE AND PREJUDICE  By Jane Austen    Chapter 1   It is a truth universally
acknowledged, that a single man in possession of a good fortune, must be in wa
nt of a wife.  However little known the feelings or views of such a man may be
on his first entering a neighbourhood, this truth is so well fixed in the mind
s of the surrounding families, that he is considered the rightful property of
some one or other of their daughters.  “My dear Mr. Bennet,” said his lady to
him one day, “have you heard that Netherfield Park is let at last?”  Mr. Benne
t replied that he had not.  “But it is,” returned she; “for Mrs. Long has just
been here, and she told me all about it.”  Mr. Bennet made no answer.  “Do you
not want to know who has taken it?” cried his wife impatiently.  “_You_ want t
o tell me, and I have no objection to hearing it.”  This was invitation enoug
h.  “Why, my dear, you must know, Mrs. Long says that Netherfield is taken by
a young man of large fortune from the north of England; that he'
```

In [18]:
```
# Convert the file to all lower case
book = book.lower()
book[:1000]
```

Out[18]:
```
'pride and prejudice  by jane austen    chapter 1   it is a truth universally
acknowledged, that a single man in possession of a good fortune, must be in wa
nt of a wife.  however little known the feelings or views of such a man may be
on his first entering a neighbourhood, this truth is so well fixed in the mind
s of the surrounding families, that he is considered the rightful property of
some one or other of their daughters.  “my dear mr. bennet,” said his lady to
him one day, “have you heard that netherfield park is let at last?”  mr. benne
t replied that he had not.  “but it is,” returned she; “for mrs. long has just
been here, and she told me all about it.”  mr. bennet made no answer.  “do you
not want to know who has taken it?” cried his wife impatiently.  “_you_ want t
o tell me, and i have no objection to hearing it.”  this was invitation enoug
h.  “why, my dear, you must know, mrs. long says that netherfield is taken by
a young man of large fortune from the north of england; that he'
```

2.2 Remove all things except alphabets and tokenize

In [19]:
```
# Remove all things except alphabets and count the words
tokens= re.findall("[a-z]+", book)
counts={}
for w in tokens:
    counts[w]= counts.get(w,0) + 1
d=list(counts.items())
df=pd.DataFrame(d, columns=['words','count'])
print(df.info())
df.head(6)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6261 entries, 0 to 6260
Data columns (total 2 columns):
words    6261 non-null object
count    6261 non-null int64
dtypes: int64(1), object(1)
memory usage: 97.9+ KB
None
```

Out[19]:

	words	count
0	pride	49
1	and	3586
2	prejudice	7
3	by	637
4	jane	293
5	austen	2

Above code splitted words like "Earth's" into 2 words- "Earth" & "s".

To avoid such things, let's do some step-by-step preprocessing where we remove all the numbers and all the punctuations.

We'll use nltk corpus for this processing.

2.3 Remove numbers and punctuations explicitly for tokenisation

In [20]:
```
book
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('punkt')
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package punkt to /Users/ujjwal/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/ujjwal/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [21]:
```
# remove numbers
text= re.sub(r'\d+', '', book)
# remove punctuations
text=text.translate(text.maketrans("","", string.punctuation))
# total no. of words
tokens_nltk = text.split()
print("No. of tokens earlier: ", len(tokens))
print("No. of tokens now: ", len(tokens_nltk))
```

```
No. of tokens earlier:  122830
No. of tokens now:  121483
```

In [22]:
```
# let's count all these words
counts={}
for w in tokens_nltk:
    counts[w]= counts.get(w,0) + 1
d = list(counts.items())
df_nltk = pd.DataFrame(d, columns=['words','count'])
print(df_nltk.info())
df_nltk.head(6)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7682 entries, 0 to 7681
Data columns (total 2 columns):
words    7682 non-null object
count    7682 non-null int64
dtypes: int64(1), object(1)
memory usage: 120.1+ KB
None
```

Out[22]:

	words	count
0	pride	41
1	and	3435
2	prejudice	6
3	by	628
4	jane	254
5	austen	2

Observe:

In our method 2, total number of tokens reduced from 122830 to 121483 but on comparing the info of word-frequency dataframe we find that the number of unique tokens has increased from 6261 to 7682.

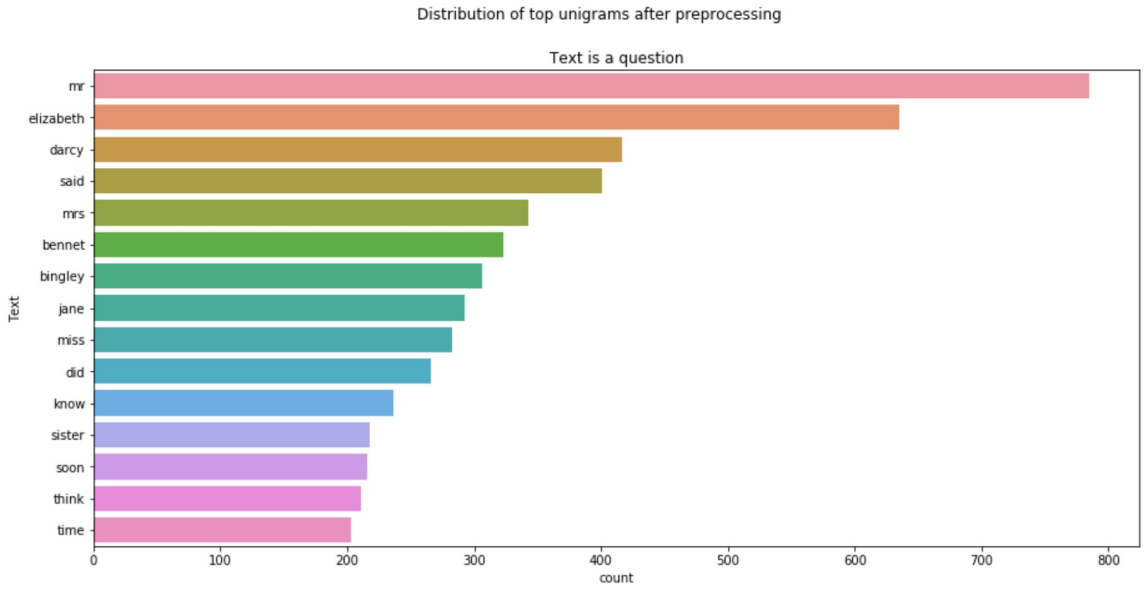
This is because earlier, "Earth" and "Earth's" both had a common token "Earth" and an undesired token "s" but now we have 2 different tokens "Earth" and "Earths" out of which none is undesired but the sense of 2nd token has been changed.

3. Analyze frequency distribution of tokens

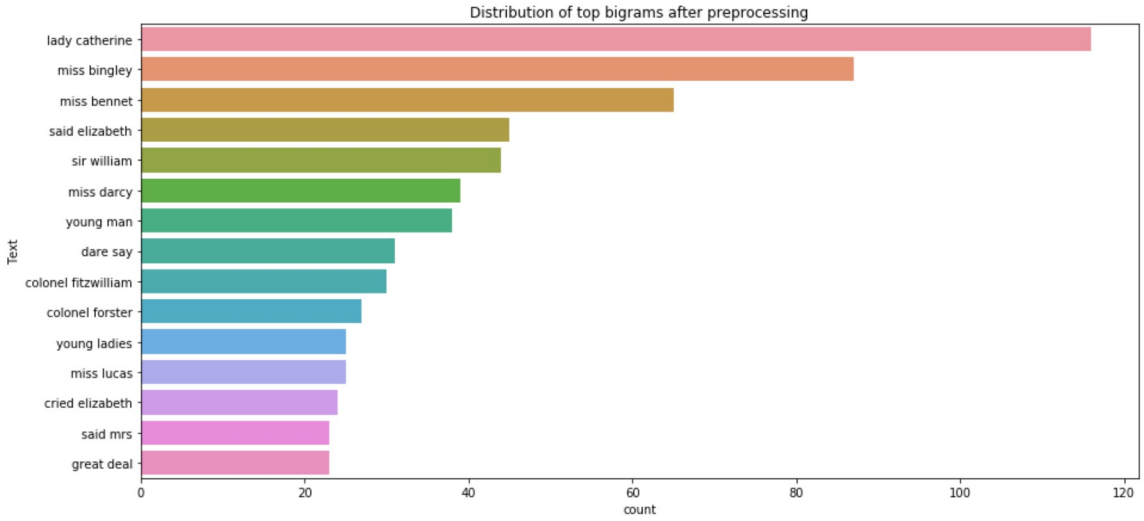
In [23]:
```
ngram = book.split('.')
```

In [24]:
```
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
import seaborn as sns
```

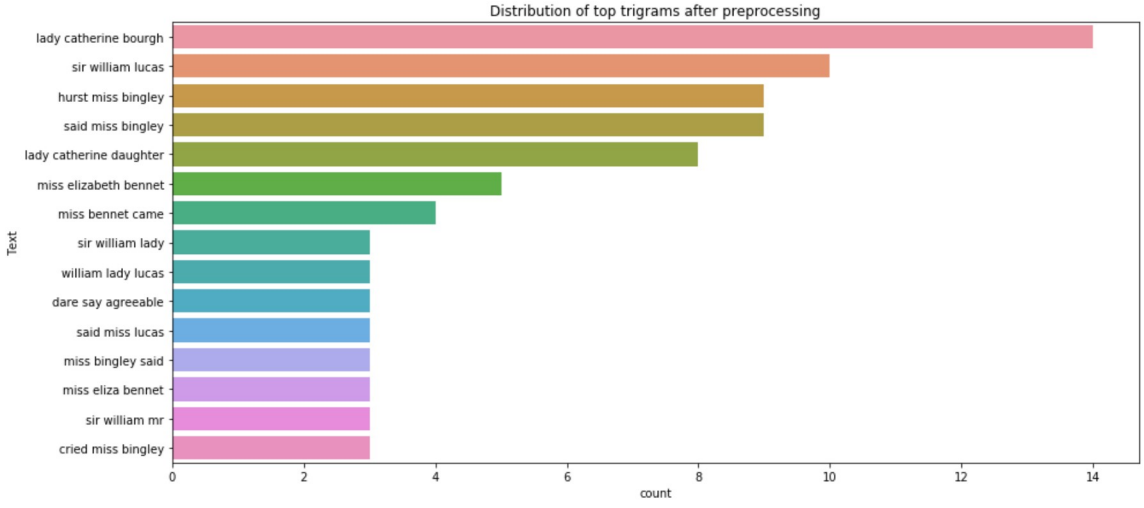
In [25]:
```
# let's look at frequency distribution of unigrams

def get_top_n_words(text, n=None):
    vec = CountVectorizer(stop_words='english').fit(text)
    bag_of_words = vec.transform(text)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.ite
ms()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq
common_words = get_top_n_words(ngram, 15)
df1 = pd.DataFrame(common_words, columns = ['Text' , 'count'])
fig, axs = plt.subplots(nrows=1, ncols=1, sharex=True);
fig.set_size_inches(15,7)
fig.suptitle('Distribution of top unigrams after preprocessing')
axs.set_title('Text is a question')
sns.barplot(y="Text", x="count",  data=df1[0:15],ax=axs);
```



In [26]:
```
# let's look at frequency distribution of bigrams
def get_top_n_words(text, n=None):
    vec = CountVectorizer(stop_words='english',ngram_range=(2, 2)).fit(text)
    bag_of_words = vec.transform(text)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.ite
ms()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq
common_words = get_top_n_words(ngram, 20)
df1 = pd.DataFrame(common_words, columns = ['Text' , 'count'])
fig, axs = plt.subplots(nrows=1, ncols=1, sharex=True);
fig.set_size_inches(15,7)
axs.set_title('Distribution of top bigrams after preprocessing')
sns.barplot(y="Text", x="count",  data=df1[0:15],ax=axs);
```



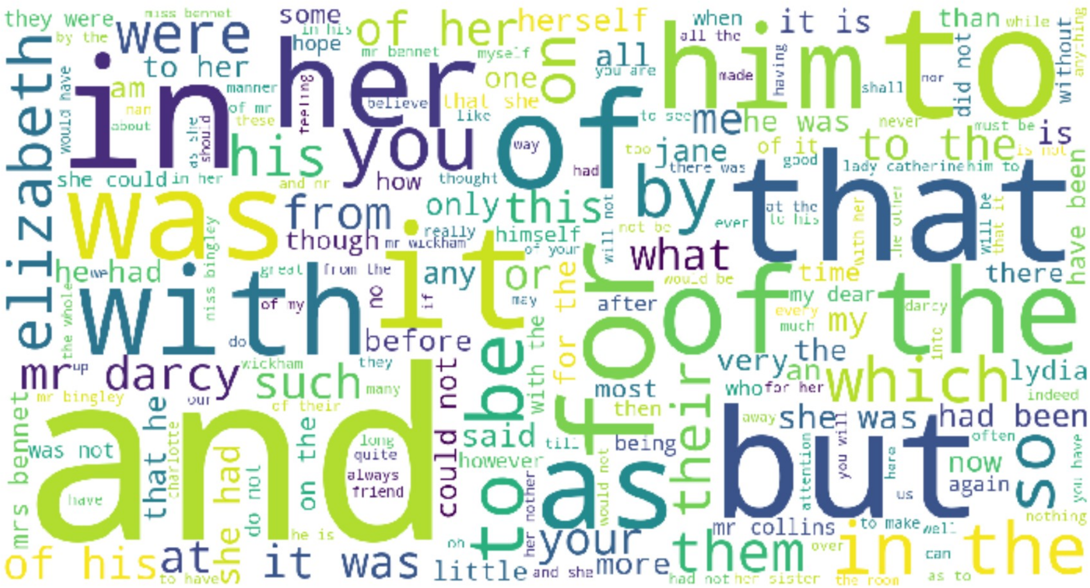
In [27]:
```
# let's look at frequency distribution of trigrams
def get_top_n_words(text, n=None):
    vec = CountVectorizer(stop_words='english',ngram_range=(3, 3)).fit(text)
    bag_of_words = vec.transform(text)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.ite
ms()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq
common_words = get_top_n_words(ngram, 15)
df1 = pd.DataFrame(common_words, columns = ['Text' , 'count'])
fig, axs = plt.subplots(nrows=1, ncols=1, sharex=True);
fig.set_size_inches(15,7)
axs.set_title('Distribution of top trigrams after preprocessing')
sns.barplot(y="Text", x="count",  data=df1[0:15],ax=axs);
```



From the different frequency distributions, we can observe that the average frequency of unigrams >> avg. frequency of bigrams >> avg. frequency of trigrams. Also from the plots we can infer that 'lady catherine', 'miss elizabeth' and 'sir william' are central characters of this text.

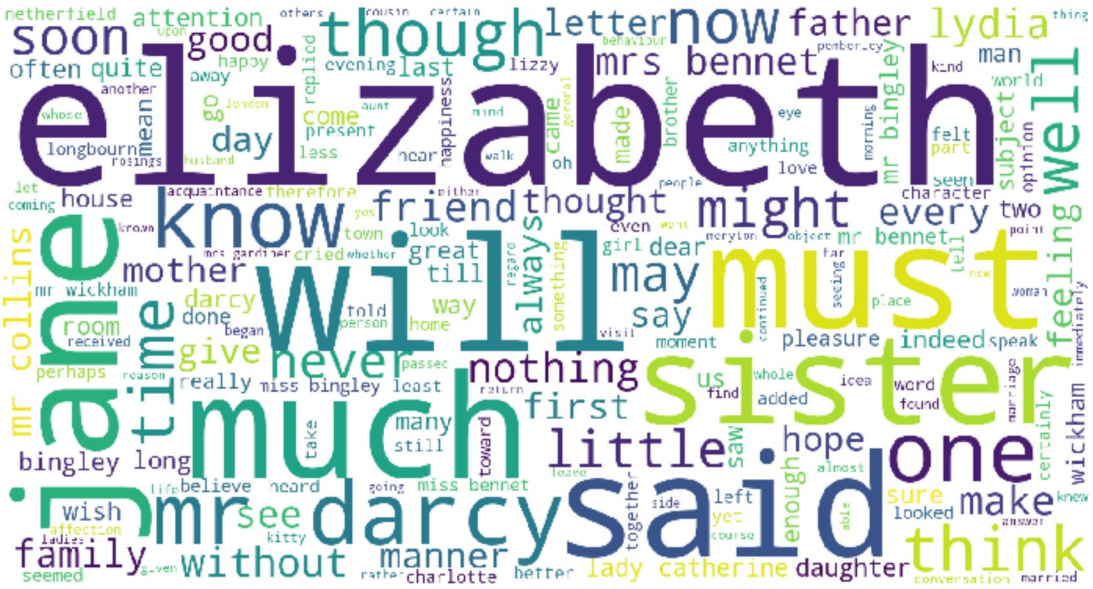
4. Create a Word Cloud of T

In [28]:
```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd

wordcloud = WordCloud(width = 1500,height = 800,stopwords = {},background_color
='white',min_font_size = 5).generate(text)
# plot the WordCloud image
plt.figure(figsize = (10, 25), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



5. Remove the stopwords and then create a word cloud - what's the difference it gives?

In [29]:
```
stopwords = set(STOPWORDS)
wordcloud = WordCloud(width = 1500,height = 800,stopwords = stopwords,background
_color ='white',min_font_size = 5).generate(text)

# plot the WordCloud image
plt.figure(figsize = (10, 25), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation

Clearly, the word cloud makes much more sense after removing the stop words because without removal of stop words we get - "and, but, that, the, to" etc. as most frequent words. This word cloud is also correct but it doesn't provide us with much useful information.

In [ ]:
```
```

6. Evaluate world length vs frequency

In [30]:
```
df_nltk['word_len']=df_nltk.words.str.len()
df_nltk.head()
```

Out[30]:

	words	count	word_len
0	pride	41	5
1	and	3435	3
2	prejudice	6	9
3	by	628	2
4	jane	254	4

In [31]:
```
df2 = df_nltk[['count','word_len']].groupby(['word_len']).sum()
df2.head(10)
```

Out[31]:

word_len	count
1	3687
2	23014
3	27928
4	21829
5	11974
6	9004
7	8402
8	5248
9	4945
10	2517

7. POS tagging and distribution of various tags

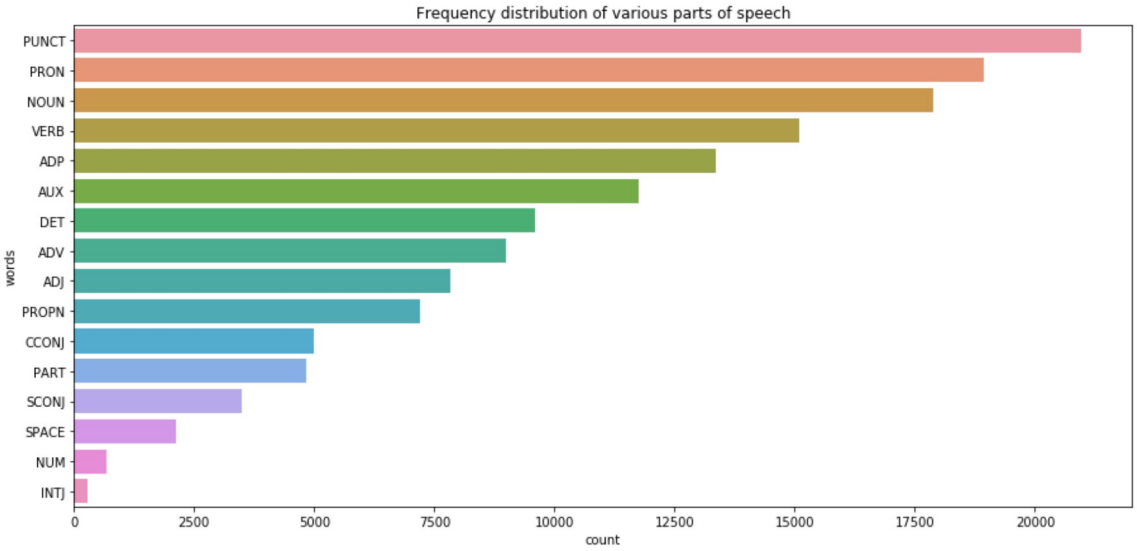
In [37]:
```
data[start_idx:end_idx].strip()[:2000]
```

Out[37]:
```
'*** START OF THIS PROJECT GUTENBERG EBOOK PRIDE AND PREJUDICE ***     Produce
d by Anonymous Volunteers      PRIDE AND PREJUDICE  By Jane Austen    Chapter
1   It is a truth universally acknowledged, that a single man in possession of
a good fortune, must be in want of a wife.  However little known the feelings
or views of such a man may be on his first entering a neighbourhood, this trut
h is so well fixed in the minds of the surrounding families, that he is consid
ered the rightful property of some one or other of their daughters.  "My dear
Mr. Bennet," said his lady to him one day, "have you heard that Netherfield Pa
rk is let at last?"  Mr. Bennet replied that he had not.  "But it is," returne
d she; "for Mrs. Long has just been here, and she told me all about it."  Mr.
Bennet made no answer.  "Do you not want to know who has taken it?" cried his
wife impatiently.  "_You_ want to tell me, and I have no objection to hearing
it."  This was invitation enough.  "Why, my dear, you must know, Mrs. Long say
s that Netherfield is taken by a young man of large fortune from the north of
England; that he came down on Monday in a chaise and four to see the place, an
d was so much delighted with it, that he agreed with Mr. Morris immediately; t
hat he is to take possession before Michaelmas, and some of his servants are t
o be in the house by the end of next week."  "What is his name?"  "Bingley."
"Is he married or single?"  "Oh! Single, my dear, to be sure! A single man of
large fortune; four or five thousand a year. What a fine thing for our girls!"
"How so? How can it affect them?"  "My dear Mr. Bennet," replied his wife, "ho
w can you be so tiresome! You must know that I am thinking of his marrying one
of them."  "Is that his design in settling here?"  "Design! Nonsense, how can
you talk so! But it is very likely that he _may_ fall in love with one of the
m, and therefore you must visit him as soon as he comes."  "I see no occasion
for that. You and the girls may go, or you may s'
```

In [38]:
```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp(book)
counts={}
for token in doc:
    counts[token.pos_]= counts.get(token.pos_,0) + 1
d=list(counts.items())
```

In [39]:
```
df_pos = pd.DataFrame(d, columns=['words','count'])
df_pos = df_pos.sort_values('count',ascending=False)
```

In [40]:
```
# frequency distribution of various POS
fig, axs = plt.subplots(nrows=1, ncols=1, sharex=True);
fig.set_size_inches(15,7)
axs.set_title('Frequency distribution of various parts of speech')
sns.barplot(y="words", x="count",  data=df_pos[0:16],ax=axs);
```



Based on the above graph we can see that Nouns and Verbs compose the major parts of speech in general in a text corpus and it aligns with our common knowledge too.