In [1]:

```
# Visualizing Word Frequency With Pandas

# Let's visualize Romeo and Juliet's top 20 words that are not stop words. To do this, w
# use features from TextBlob, NLTK and pandas. Pandas visualization capabilities are bas
# on Matplotlib.

# First, let's load Romeo and Juliet.

from pathlib import Path
from textblob import TextBlob
blob = TextBlob(Path('romeoAndjuliet.txt').read_text())
```

In [2]:

```
# Next, load the NLTK stopwords:

from nltk.corpus import stopwords
stop_words = stopwords.words('english')
```

In [3]:

```
# To visualize the top 20 words, we need each word and its frequency. Let's call the
# blob.word_counts dictionary's items method to get a list of word-frequency tuples:

items = blob.word_counts.items()
```

In [4]:

```
# Next, let's use a list comprehension to eliminate any tuples containing stop words:

items = [item for item in items if item[0] not in stop_words]

# The expression item[0] gets the word from each tuple so we can check whether it's in
# stop_words.
```

In [5]:

```
# To determine the top 20 words, let's sort the tuples in items in descending order by f
# We can use built-in function sorted with a key argument to sort the tuples by
# the frequency element in each tuple. To specify the tuple element to sort by, use the
# itemgetter function from the Python Standard Library's operator module:

from operator import itemgetter
sorted_items = sorted(items, key=itemgetter(1), reverse=True)
```

In [6]:

```
# As sorted orders items' elements, it accesses the element at index 1 in each tuple via
# expression itemgetter(1). The reverse=True keyword argument indicates that the
# tuples should be sorted in descending order.
```

In [7]:

```
# Next, we use a slice to get the top 20 words from sorted_items. When TextBlob
# tokenizes a corpus, it splits all contractions at their apostrophes and counts
# the total numberof apostrophes as one of the “words.” Romeo and Juliet has many
# contractions.
# If you display sorted_items[0], you’ll see that they are the most frequently occurring
# “word” with 867 of them.22 We want to display only words, so we ignore element 0 and
# get a slice containing elements 1 through 20 of sorted_items:

top20 = sorted_items[1:21]
```

In [8]:

```
# Next, let’s convert the top20 list of tuples to a pandas DataFrame
# so we can visualize it conveniently:

import pandas as pd
df = pd.DataFrame(top20, columns=['word', 'count'])
df
```

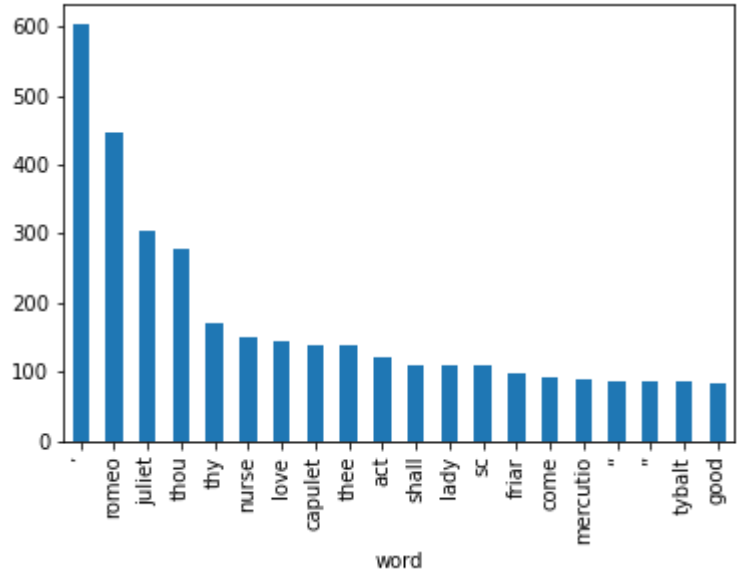
Out[8]:

	word	count
0	’	602
1	romeo	445
2	juliet	303
3	thou	277
4	thy	169
5	nurse	150
6	love	145
7	capulet	137
8	thee	137
9	act	121
10	shall	110
11	lady	109
12	sc	109
13	friar	97
14	come	91
15	mercutio	90
16	“	87
17	”	87
18	tybalt	86
19	good	84

In [9]:

```
# To visualize the data, we'll use the bar method of the DataFrame's plot property. The
# arguments indicate which column's data should be displayed along the x- and y-axes, ar
# that we do not want to display a legend on the graph:

axes = df.plot.bar(x='word', y='count', legend=False)
```



In [16]:

```
# The bar method creates and displays a Matplotlib bar chart.
# When you look at the initial bar chart that appears, you'll notice that some of the
# words are truncated. To fix that, use Matplotlib's gcf (get current figure) function t
# the Matplotlib figure that pandas displayed, then call the figure's tight_layout metho
# This compresses the bar chart to ensure all its components fit:

import matplotlib.pyplot as plt
plt.gcf().tight_layout()
```

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