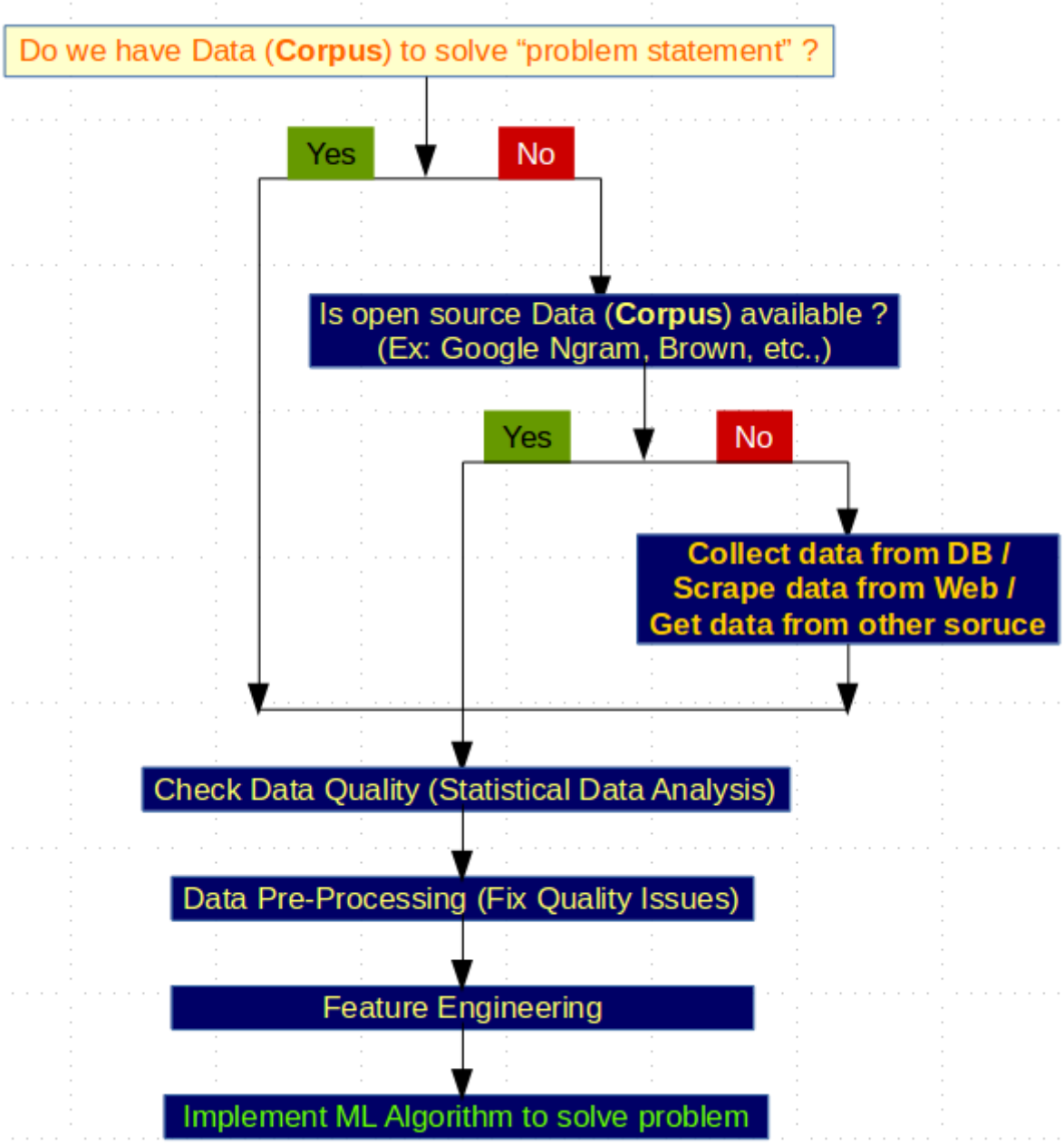
Corpus: is a collection of digitized "spoken" or "writen" content. The corpus is used for linguistic analysis or corpus analysis.

- Example: If we are trying to build a grammar correction system, we use text corpus to find out and correct grammarly incorrect instances.
- Monolingual Corpus : The corpus will have one language.
- Bilingual : The corpus will have two languages.
- Multilingual : The corpus will have two languages.
    - Examples: Google Ngram, Brown corpus, American National corpus.

Corpus and its Significance: A corpus is most critical and basic building block in an NLP application. Corpus provides the qualitative data useful in building NLP appication. Below are the things we need to consider while corpus selection (or collect a corpus).

- Type of the Data (What kind of Data?)
- Availability of Data
- Quality of the Data
- Adequacy of the Data



Corpus Analysis (Statistical Data Analysis): Corpus analysis helps understand data and fix noice in it. The general analysis involves,

- How many differet words are present in Corpus
- What is the frequency of a certain words.
- etc.,

Types of Corpora:

- **Isolated Corpus**: Collection of natural language or text.
- **Categorized Corpus**: Collection of text grouped into different categories.
- **Overlaping Corpus**: Collection of categorized text, but overlap with each other.
- **Temporal Corpus**: Collection of usages of natural language over a period of time.
- Basic Corpus Functionality defined in NLTK: more documentation can be found using **help(nltk.corpus.reader)** and by reading the online Corpus **HOWTO at** http://nltk.org/howto (http://nltk.org/howto).

Method Name	Description
fileids()	the files of the corpus
fileids([categories])	the files of the corpus corresponding to these categories
categories()	the categories of the corpus
categories([fileids])	the categories of the corpus corresponding to these files
raw()	the raw content of the corpus
raw(fileids=[f1,f2,f3])	the raw content of the specified files
raw(categories=[c1,c2])	the raw content of the specified categories
words()	the words of the whole corpus
words(fileids=[f1,f2,f3])	the words of the specified fileids
words(categories=[c1,c2])	the words of the specified categories
sents()	the sentences of the whole corpus
sents(fileids=[f1,f2,f3])	the sentences of the specified fileids
sents(categories=[c1,c2])	the sentences of the specified categories
abspath(fileid)	the location of the given file on disk
encoding(fileid)	the encoding of the file (if known)
open(fileid)	open a stream for reading the given corpus file
root	if the path to the root of locally installed corpus
readme()	the contents of the README file of the corpus

- **Isolated Corpus**:Collection of natural language or text.
  - Example 1: **"Established Literature - Gutenberg"**; NLTK's **Gutenberg** electronic text archive, which contains some 25,000 free electronic books, hosted at http://www.gutenberg.org/ (http://www.gutenberg.org/).
  - Example 2: **"Less Formal Language - web text"** - NLTK's small collection of **web text** includes content from a Firefox discussion forum, conversations overheard in New York, the movie script of Pirates of the Carribean, personal advertisements, and wine reviews

In [5]:

```
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/inde
x.xml (https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml)
```

Out[5]:

```
True
```

In [4]:

```
import nltk
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [4]:

```
nltk.corpus.gutenberg.fileids()
```

Out[4]:

```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

In [7]:

```
guten_emma_words = nltk.corpus.gutenberg.words(['austen-emma.txt'])
print(guten_emma_words[:])
```

```
['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']', ...]
```

In [8]:

```
len(guten_emma_words)
```

Out[8]:

```
192427
```

In [9]:

```
nltk.corpus.webtext.fileids()
```

Out[9]:

```
['firefox.txt',
 'grail.txt',
 'overheard.txt',
 'pirates.txt',
 'singles.txt',
 'wine.txt']
```

In [10]:

```
webtext_firefox_words = nltk.corpus.webtext.words(['firefox.txt'])
print(webtext_firefox_words)
```

```
['Cookie', 'Manager', ':', '"', 'Don', "'", 't', ...]
```

In [11]:

```
len(webtext_firefox_words)
```

Out[11]:

```
102457
```

In [12]:

```
nltk.corpus.webtext.raw(['firefox.txt'])[:500]
```

Out[12]:

```
'Cookie Manager: "Don\'t allow sites that set removed cookies to set future
cookies" should stay checked\r\nWhen in full screen mode\r\nPressing Ctrl-N
should open a new browser when only download dialog is left open\r\nadd icon
s to context menu\r\nSo called "tab bar" should be made a proper toolbar or
given the ability collapse / expand.\r\n[XUL] Implement Cocoa-style toolbar
customization.\r\n#ifdefs for MOZ_PHOENIX\r\ncustomize dialog\'s toolbar has
small icons when small icons is not checked\r\nnightly builds '
```

In [13]:

```
nltk.corpus.webtext.sents(['firefox.txt'])
```

Out[13]:

```
[['Cookie', 'Manager', ':', '"', 'Don', "'", 't', 'allow', 'sites', 'that',
'set', 'removed', 'cookies', 'to', 'set', 'future', 'cookies', '"', 'shoul
d', 'stay', 'checked', 'When', 'in', 'full', 'screen', 'mode', 'Pressing',
'Ctrl', '-', 'N', 'should', 'open', 'a', 'new', 'browser', 'when', 'only',
'download', 'dialog', 'is', 'left', 'open', 'add', 'icons', 'to', 'context',
'menu', 'So', 'called', '"', 'tab', 'bar', '"', 'should', 'be', 'made', 'a',
'proper', 'toolbar', 'or', 'given', 'the', 'ability', 'collapse', '/', 'expa
nd', '.'], ['[', 'XUL', ']', 'Implement', 'Cocoa', '-', 'style', 'toolbar',
'customization', '.'], ...]
```

- Categorized Corpus: Collection of text grouped into different categories.
  - Example 1: The **Brown Corpus** was the first million-word electronic corpus of English, created in 1961 at Brown University. This corpus contains text from 500 sources, and the sources have been categorized by genre, such as news, editorial, and so on (http://icame.uib.no/brown/bcm-los.html (http://icame.uib.no/brown/bcm-los.html)).

In [14]:

```
type(nltk.corpus.brown.fileids())
```

Out[14]:

```
list
```

In [15]:

```
nltk.corpus.brown.fileids()[:10]
```

Out[15]:

```
['ca01',
 'ca02',
 'ca03',
 'ca04',
 'ca05',
 'ca06',
 'ca07',
 'ca08',
 'ca09',
 'ca10']
```

In [16]:

```
nltk.corpus.brown.categories(['ca01', 'ca02','ca10', 'cd09','cc14'])
```

Out[16]:

```
['news', 'religion', 'reviews']
```

In [17]:

```
nltk.corpus.brown.raw(fileids=['cd09'])[:700]
```

Out[17]:

```
"Few/ap persons/nns who/wps join/vb the/at Church/nn-tl are/ber insincere/jj
./.\nThey/ppss earnestly/rb desire/vb to/to do/do the/at will/nn of/in God/n
p ./.\nWhen/wrb they/ppss fall/vb by/in the/at wayside/nn and/cc fail/vb to/
to achieve/vb Christian/jj stature/nn ,/, it/pps is/bez an/at indictment/nn
of/in the/at Church/nn-tl ./.\nThese/dts fatalities/nns are/ber dramatic/jj
evidence/nn of/in ``/`` halfway/jj evangelism/nn ''/'' ,/, a/at failure/nn t
o/to follow/vb through/rp ./.\nA/at program/nn of/in Lay/jj-tl Visitation/nn
-tl Evangelism/nn-tl can/md end/vb in/in dismal/jj defeat/nn with/in half/ab
n the/at new/jj members/nns drifting/vbg away/rb unless/cs practical/jj plan
s/nns and/cc strenuo"
```

In [18]:

```
nltk.corpus.brown.categories()
```

Out[18]:

```
['adventure',
 'belles_lettres',
 'editorial',
 'fiction',
 'government',
 'hobbies',
 'humor',
 'learned',
 'lore',
 'mystery',
 'news',
 'religion',
 'reviews',
 'romance',
 'science_fiction']
```

- Overlaping Corpus: Collection of categorized text, but overlap with each other.
  - Example 1: The **Reuters Corpus** contains 10,788 news documents totaling 1.3 million words. The documents have been classified into 90 topics, and grouped into two sets, called "training" and "test"; thus, the text with fileid 'test/14826' is a document drawn from the test set.
    - Categories in the Reuters corpus overlap with each other, simply because a news story often covers multiple topics. We can ask for the topics covered by one or more documents, or for the documents included in one or more categories. For convenience, the corpus methods accept a single fileid or a list of fileids.

In [19]:

```
nltk.corpus.reuters.fileids()[:10]
```

Out[19]:

```
['test/14826',
 'test/14828',
 'test/14829',
 'test/14832',
 'test/14833',
 'test/14839',
 'test/14840',
 'test/14841',
 'test/14842',
 'test/14843']
```

In [20]:

```
nltk.corpus.reuters.categories()[:10]
```

Out[20]:

```
['acq',
 'alum',
 'barley',
 'bop',
 'carcass',
 'castor-oil',
 'cocoa',
 'coconut',
 'coconut-oil',
 'coffee']
```

In [21]:

```
nltk.corpus.reuters.fileids(categories=['acq'])[:10]
```

Out[21]:

```
['test/14843',
 'test/14852',
 'test/14865',
 'test/14888',
 'test/14900',
 'test/14907',
 'test/14909',
 'test/14921',
 'test/14932',
 'test/14941']
```

In [22]:

```
nltk.corpus.reuters.words(fileids=['training/11920'])
```

Out[22]:

```
['CYCLOPS', '&', 'lt', ';', 'CYL', '>', 'NAMES', ...]
```

In [23]:

```
nltk.corpus.reuters.sents(fileids=['training/11920'])
```

Out[23]:

```
[['CYCLOPS', '&', 'lt', ';', 'CYL', '>', 'NAMES', 'DIXONS', 'OFFICIALS', 'T
O', 'BOARD', 'Cyclops', 'corp', 'said', 'it', 'has', 'reconstituted', 'its',
'board', 'to', 'include', 'three', '&', 'lt', ';', 'Dixons', 'Group', 'PLC',
'>', 'executives', 'following', 'Dixons', "'", 'acquisition', 'of', '83', 'p
ct', 'of', 'Cyclops', "'", '4', ',', '061', ',', '000', 'shares', 'in', 'a',
'95', 'dlr', 'per', 'share', 'tender', 'offer', '.'], ['Cyclops', 'said', 'r
emaining', 'on', 'the', 'six', '-', 'member', 'board', 'are', 'chairman', 'a
nd', 'chief', 'executive', 'W', '.', 'H', '.'], ...]
```

In [24]:

```
nltk.corpus.reuters.raw(fileids=['training/11920'])
```

Out[24]:

```
"CYCLOPS &lt;CYL> NAMES DIXONS OFFICIALS TO BOARD\n  Cyclops corp said it ha
s\n  reconstituted its board to include three &lt;Dixons Group PLC>\n  execu
tives following Dixons' acquisition of 83 pct of Cyclops'\n  4,061,000 share
s in a 95 dlr per share tender offer.\n      Cyclops said remaining on the s
ix-member board are chairman\n  and chief executive W.H. Knoell, president a
nd chief operating\n  officer James F. Will and senior vice president Willia
m D.\n  Dickey.\n  \n\n"
```

- Temporal Corpus: Collection of usages of natural language over a period of time.
  - Example 1: The **Inaugural Corpus** is actually a collection of 55 texts, one for each presidential address. An interesting property of this collection is its time dimension.

In [25]:

```
nltk.corpus.inaugural.fileids()[:10]
```

Out[25]:

```
['1789-Washington.txt',
 '1793-Washington.txt',
 '1797-Adams.txt',
 '1801-Jefferson.txt',
 '1805-Jefferson.txt',
 '1809-Madison.txt',
 '1813-Madison.txt',
 '1817-Monroe.txt',
 '1821-Monroe.txt',
 '1825-Adams.txt']
```

In [26]:

```
nltk.corpus.inaugural.categories()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-26-07c3138233d4> in <module>
----> 1 nltk.corpus.inaugural.categories()

AttributeError: 'PlaintextCorpusReader' object has no attribute 'categories'
```

In [27]:

```
help(nltk.corpus.inaugural)
```

```
class PlaintextCorpusReader(nltk.corpus.reader.api.CorpusReader)
 |  PlaintextCorpusReader(root, fileids, word_tokenizer=WordPunctTokenizer
(pattern='\\w+|[^\\w\\s]+', gaps=False, discard_empty=True, flags=<RegexFl
ag.UNICODE|DOTALL|MULTILINE: 56>), sent_tokenizer=<nltk.tokenize.punkt.Pun
ktSentenceTokenizer object at 0x000002B7824597C8>, para_block_reader=<func
tion read_blankline_block at 0x000002B782463948>, encoding='utf8')
 |
 |  Reader for corpora that consist of plaintext documents.  Paragraphs
 |  are assumed to be split using blank lines.  Sentences and words can
 |  be tokenized using the default tokenizers, or by custom tokenizers
 |  specificed as parameters to the constructor.
 |
 |  This corpus reader can be customized (e.g., to skip preface
 |  sections of specific document formats) by creating a subclass and
 |  overriding the ``CorpusView`` class variable.
 |
 |  Method resolution order:
 |      PlaintextCorpusReader
 |      nltk.corpus.reader.api.CorpusReader
 |      builtins.object
 |
```

In [28]:

```
nltk.corpus.inaugural.raw(fileids=['1789-Washington.txt'])[:600]
```

Out[28]:

```
'Fellow-Citizens of the Senate and of the House of Representatives:\n\nAmong
the vicissitudes incident to life no event could have filled me with greater
anxieties than that of which the notification was transmitted by your order,
and received on the 14th day of the present month. On the one hand, I was su
mmoned by my Country, whose voice I can never hear but with veneration and l
ove, from a retreat which I had chosen with the fondest predilection, and, i
n my flattering hopes, with an immutable decision, as the asylum of my decli
ning years -- a retreat which was rendered every day more necessary '
```

In [29]:

```
nltk.corpus.inaugural.abspath('1789-Washington.txt')
```

Out[29]:

```
FileSystemPathPointer('C:\\Users\\thisi\\AppData\\Roaming\\nltk_data\\corpor
a\\inaugural\\1789-Washington.txt')
```

FreqDist

In [30]:

```
words = nltk.corpus.inaugural.words(fileids=['1789-Washington.txt'])
```

In [33]:

```
words
```

Out[33]:

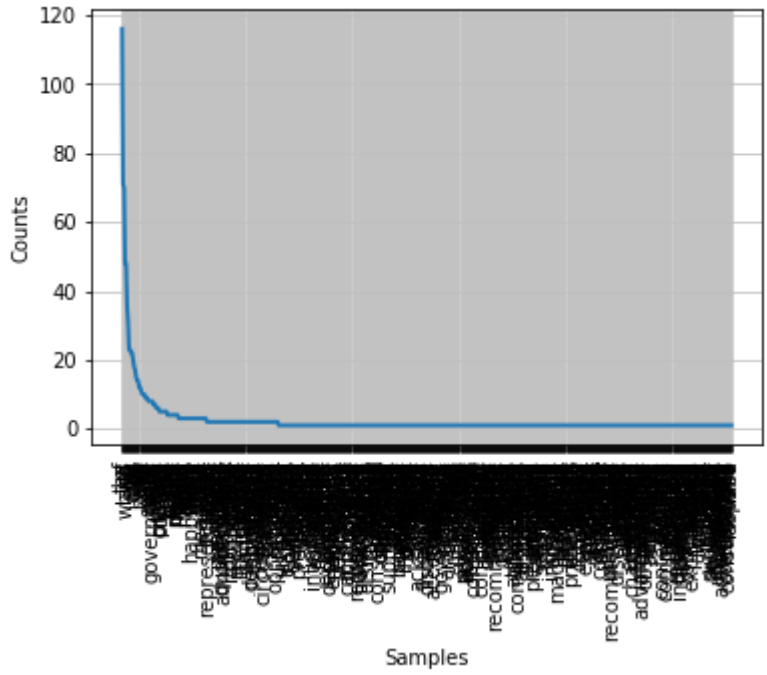
```
['Fellow', '-', 'Citizens', 'of', 'the', 'Senate', ...]
```

In [34]:

```
freqDist = nltk.FreqDist(w.lower() for w in words )
```

In [35]:

```
freqDist.plot()
```



Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2b7885d8608>
```

In [37]:

```
list(freqDist.keys())[:10]
```

Out[37]:

```
['fellow',
 '-',
 'citizens',
 'of',
 'the',
 'senate',
 'and',
 'house',
 'representatives',
 ':']
```

In [38]:

```
list(freqDist.values())[:10]
```

Out[38]:

```
[3, 1, 5, 71, 116, 1, 48, 2, 2, 1]
```

In [36]:

```
#freqDist.items()
```

In [39]:

```
df_freq_dist_key_val = pd.DataFrame([(k,v) for (k,v) in freqDist.items() if v > 10], column
```

In [40]:

```
df_freq_dist_key_val.head()
```

Out[40]:

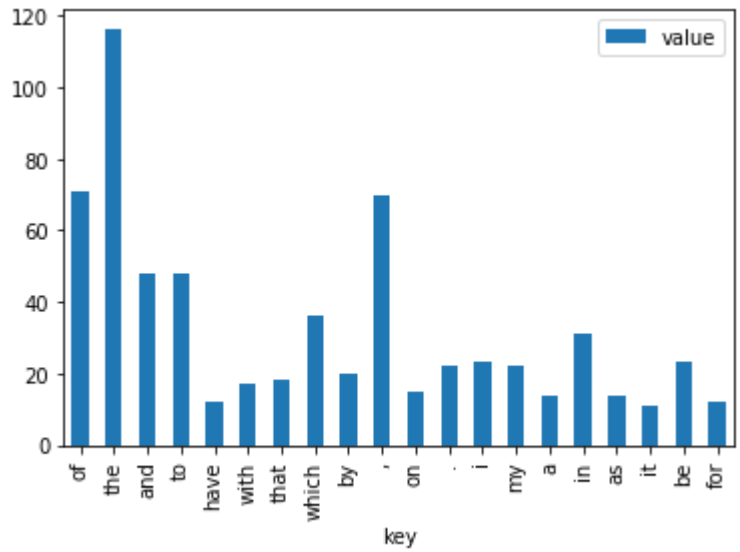
	key	value
0	of	71
1	the	116
2	and	48
3	to	48
4	have	12

In [41]:

```
df_freq_dist_key_val.plot(kind='bar', x='key', y='value')
```

Out[41]:

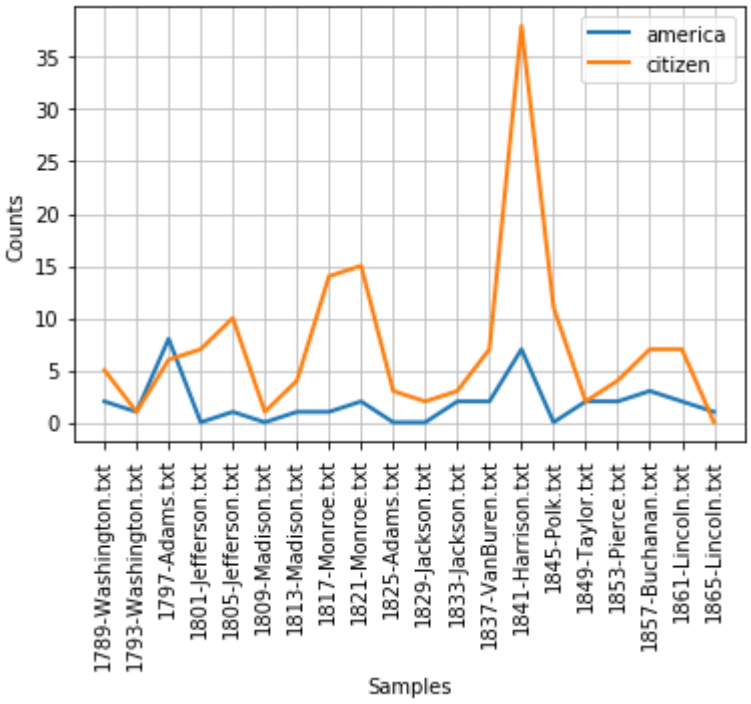
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b78a4b8a48>
```



ConditionalFreqDist:

In [40]:

```
cfd = nltk.ConditionalFreqDist(
           (target, fileid)
           for fileid in nltk.corpus.inaugural.fileids()[:20]
           for w in nltk.corpus.inaugural.words(fileid)
           for target in ['america', 'citizen']
           if w.lower().startswith(target))
cfd.plot()
```



In [41]:

```
cfd
```

Out[41]:

```
ConditionalFreqDist(nltk.probability.FreqDist,
                    {'america': FreqDist({'1789-Washington.txt': 2,
                              '1793-Washington.txt': 1,
                              '1797-Adams.txt': 8,
                              '1805-Jefferson.txt': 1,
                              '1813-Madison.txt': 1,
                              '1817-Monroe.txt': 1,
                              '1821-Monroe.txt': 2,
                              '1833-Jackson.txt': 2,
                              '1837-VanBuren.txt': 2,
                              '1841-Harrison.txt': 7,
                              '1849-Taylor.txt': 2,
                              '1853-Pierce.txt': 2,
                              '1857-Buchanan.txt': 3,
                              '1861-Lincoln.txt': 2,
                              '1865-Lincoln.txt': 1}),
                     'citizen': FreqDist({'1789-Washington.txt': 5,
                              '1793-Washington.txt': 1,
                              '1797-Adams.txt': 6,
                              '1801-Jefferson.txt': 7,
                              '1805-Jefferson.txt': 10,
                              '1809-Madison.txt': 1,
                              '1813-Madison.txt': 4,
                              '1817-Monroe.txt': 14,
                              '1821-Monroe.txt': 15,
                              '1825-Adams.txt': 3,
                              '1829-Jackson.txt': 2,
                              '1833-Jackson.txt': 3,
                              '1837-VanBuren.txt': 7,
                              '1841-Harrison.txt': 38,
                              '1845-Polk.txt': 11,
                              '1849-Taylor.txt': 2,
                              '1853-Pierce.txt': 4,
                              '1857-Buchanan.txt': 7,
                              '1861-Lincoln.txt': 7})})
```

In [44]:

```
cfd2 = nltk.ConditionalFreqDist(
           (target, fileid[:4])
           for fileid in nltk.corpus.inaugural.fileids()[:20]
           for w in nltk.corpus.inaugural.words(fileid)
           for target in ['america', 'citizen']
           if w.lower().startswith(target))
cfd2.tabulate()
```

```
        1789 1793 1797 1801 1805 1809 1813 1817 1821 1825 1829 1833 1837 184
1 1845 1849 1853 1857 1861 1865
america    2    1    8    0    1    0    1    1    2    0    0    2    2
7    0    2    2    3    2    1
citizen    5    1    6    7   10    1    4   14   15    3    2    3    7   3
8   11    2    4    7    7    0
```

In [45]:

```
dir(nltk.corpus)
```

```
 'CategorizedTaggedCorpusReader',
 'ChasenCorpusReader',
 'ChunkedCorpusReader',
 'ComparativeSentencesCorpusReader',
 'ConllChunkCorpusReader',
 'ConllCorpusReader',
 'CorpusReader',
 'CrubadanCorpusReader',
 'DependencyCorpusReader',
 'EuroparlCorpusReader',
 'FramenetCorpusReader',
 'IEERCorpusReader',
 'IPIPANCorpusReader',
 'IndianCorpusReader',
 'KNBCorpusReader',
 'LazyCorpusLoader',
 'LinThesaurusCorpusReader',
 'MTECorpusReader',
 'MWAPPDBCorpusReader',
```

Opensource Data

- Amazon Reviews (Classification, Sentiment Analysis): https://www.kaggle.com/bittlingmayer/amazonreviews (https://www.kaggle.com/bittlingmayer/amazonreviews), https://registry.opendata.aws/amazon-reviews/ (https://registry.opendata.aws/amazon-reviews/)
- OpinRank Review Dataset Data Set (Snetiment Analysis, Clustering): This data set contains user reviews of cars and and hotels collected from Tripadvisor (~~259,000 reviews) and Edmunds (~~42,230 reviews) http://archive.ics.uci.edu/ml/datasets/opinrank+review+dataset (http://archive.ics.uci.edu/ml/datasets/opinrank+review+dataset)
- MovieLens (Regression, Clustering, Classification): GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org (http://movielens.org)). https://grouplens.org/datasets/movielens/ (https://grouplens.org/datasets/movielens/), https://www.kaggle.com/prajitdatta/movielens-100k-dataset (https://www.kaggle.com/prajitdatta/movielens-100k-dataset)
- Yahoo! Music User Ratings of Musical Artists (Clustering, Regression): https://webscope.sandbox.yahoo.com/catalog.php?datatype=r&guccounter=1 (https://webscope.sandbox.yahoo.com/catalog.php?datatype=r&guccounter=1)
- Car Evaluation Data Set (Classification):Derived from simple hierarchical decision model, this database may be useful for testing constructive induction and structure discovery methods. https://archive.ics.uci.edu/ml/datasets/car+evaluation (https://archive.ics.uci.edu/ml/datasets/car+evaluation), https://www.kaggle.com/elikplim/car-evaluation-data-set (https://www.kaggle.com/elikplim/car-evaluation-data-set)
- YouTube Comedy Slam Preference Data Set (Classfication): This dataset provides user vote data on which video from a pair of videos is funnier collected on YouTube Comedy Slam. The task is to automatically predict this preference based on video metadata. https://archive.ics.uci.edu/ml/datasets/YouTube+Comedy+Slam+Preference+Data (https://archive.ics.uci.edu/ml/datasets/YouTube+Comedy+Slam+Preference+Data), https://www.kaggle.com/uciml/youtube-comedy-slam (https://www.kaggle.com/uciml/youtube-comedy-slam)
- Skytrax User Reviews Dataset (Classification, Regression): A scraped dataset created from all user reviews found on Skytrax (www.airlinequality.com (http://www.airlinequality.com)).

https://github.com/quankiquanki/skytrax-reviews-dataset (https://github.com/quankiquanki/skytrax-reviews-dataset)

- Teaching Assistant Evaluation Data Set (Classification): The data consist of evaluations of teaching performance; scores are "low", "medium", or "high". https://archive.ics.uci.edu/ml/datasets/teaching+assistant+evaluation (https://archive.ics.uci.edu/ml/datasets/teaching+assistant+evaluation)

Twitter

- Sentiment140 dataset (Sentiment Analysis) : https://www.kaggle.com/kazanova/sentiment140 (https://www.kaggle.com/kazanova/sentiment140)
- Social Computing Data Repository (Clustering, Graph Analysis): http://socialcomputing.asu.edu/ (http://socialcomputing.asu.edu/)
- Dataset : Twitter - http://socialcomputing.asu.edu/datasets/Twitter (http://socialcomputing.asu.edu/datasets/Twitter)