Preprocessing: Cleaning Data

Preprocessing: Cleaning Data

There are numerous osteps that can be taken to help put all text on equal footing, many of which involve the comparatively simple ideas of substitution or removal. They are, however, no less important to the overall process. These include:

- set all characters to lowercase
- remove punctuation (generally part of tokenization, but still worth keeping in mind at this stage, even as confirmation)
- remove numbers (or convert numbers to textual representations)
- strip white space (also generally part of tokenization)
- remove default stop words (general English stop words)

Import Data

I've included an excerpt from Amazon Fine Food Reviews in the Data Folder as well! This file is called `Amazon Reviews.csv`.

I have reduced it into a smaller one called `Food_Review.csv`

In [1]:
```
import pandas as pd
df = pd.read_csv('Food_Review.csv')
```

jupyter and pandas display is a good resource to help use jupyters display with pandas to the fullest.

In [2]:
```
df.head(2)
```

Out[2]:

	Summary	Text
0	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...

In [3]:
```
df['Text'].head(3)
```

Out[3]:
```
0    I have bought several of the Vitality canned d...
1    Product arrived labeled as Jumbo Salted Peanut...
2    This is a confection that has been around a fe...
Name: Text, dtype: object
```

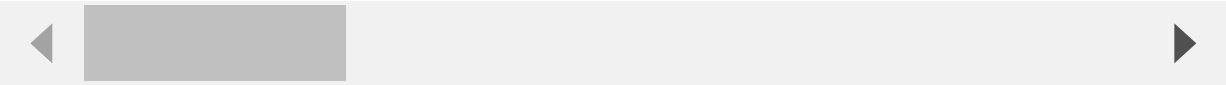
In [4]:
```
#for automatic linebreaks and multi-line cells.
pd.set_option('display.max_colwidth', -1)
```

```
C:\Users\rebec\AppData\Local\Temp\ipykernel_25740\1423554688.py:2: FutureWarning: Pas
sing a negative integer is deprecated in version 1.0 and will not be supported in fut
ure version. Instead, use None to not limit the column width.
  pd.set_option('display.max_colwidth', -1)
```

In [5]:
```
#suppress all warnings with this
import warnings
warnings.filterwarnings("ignore")
```

In [6]:
```
df['Text'].head(3)
```

Out[6]:
```
0    I have bought several of the Vitality canned dog food products and have found th
em all to be of good quality. The product looks more like a stew than a processed mea
t and it smells better. My Labrador is finicky and she appreciates this product bette
r than  most.
1    Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually smal
l sized unsalted. Not sure if this was an error or if the vendor intended to represen
t the product as "Jumbo".
2    This is a confection that has been around a few centuries.  It is a light, pillo
wy citrus gelatin with nuts - in this case Filberts. And it is cut into tiny squares
and then liberally coated with powdered sugar.  And it is a tiny mouthful of heaven.
Not too chewy, and very flavorful.  I highly recommend this yummy treat.  If you are
familiar with the story of C.S. Lewis' "The Lion, The Witch, and The Wardrobe" - this
is the treat that seduces Edmund into selling out his Brother and Sisters to the Witc
h.
Name: Text, dtype: object
```



Breaking a Large String Into Smaller Strings

A big task for preparing string data is breaking the string into smaller substrings. In ths notebook we'll focus on breaking our Amazon Fine Food Reviews excerpt into individual words, then we'll look into trying to make individual sentences. Our goal by the end of this notebook is to be able to take in our excerpt and return a word count pandas dataframe.

Individual Words

`str.split()`.

The `split` function inherent to all `str` objects in python allows you to take a string and break it into a list of substrings based on the input it is given.

In [7]:
```
df['Text'].head(2).str.split()
```

Out[7]:
```
0    [I, have, bought, several, of, the, Vitality, canned, dog, food, products, and,
have, found, them, all, to, be, of, good, quality., The, product, looks, more, like,
a, stew, than, a, processed, meat, and, it, smells, better., My, Labrador, is, finick
y, and, she, appreciates, this, product, better, than, most.]
1    [Product, arrived, labeled, as, Jumbo, Salted, Peanuts...the, peanuts, were, act
ually, small, sized, unsalted., Not, sure, if, this, was, an, error, or, if, the, ven
dor, intended, to, represent, the, product, as, "Jumbo".]
Name: Text, dtype: object
```

Since we want words, let's first lower ervery word in our dataframe.
this is accomplished by using `.str.lower()`

The `str.lower()` method will take all `A-Z` characters in the string and turn them into their corresponding `a-z` form.

In [8]:
```
"THE Ohio State University".lower()
```

Out[8]:
```
'the ohio state university'
```

In [10]:
```
# We lower all srings
df['Text_clean'] = df['Text'].str.lower()
```

In [11]:
```
df['Text_clean'].head(1)
```

Out[11]:
```
0    i have bought several of the vitality canned dog food products and have found th
em all to be of good quality. the product looks more like a stew than a processed mea
t and it smells better. my labrador is finicky and she appreciates this product bette
r than  most.
Name: Text_clean, dtype: object
```

`str.replace()` We can replace any specified substring within a string with another specified substring using `str.replace()`. This can help us eliminate the pesky punctuation.

In [12]:
```
### Some substrings we'll want to remove are:
## , ",", ".", "!", "?", "\'", '\"', "-", "(", ")"

df['Text_cleaned'] = df['Text_clean'].replace(",","")
df['Text_cleaned'] = df['Text_cleaned'].replace(".","")
df['Text_cleaned'] = df['Text_cleaned'].replace("!","")
df['Text_cleaned'] = df['Text_cleaned'].replace("?","")
df['Text_cleaned'] = df['Text_cleaned'].replace("\'","")
df['Text_cleaned'] = df['Text_cleaned'].replace('\"',"")
df['Text_cleaned'] = df['Text_cleaned'].replace("-"," ")
df['Text_cleaned'] = df['Text_cleaned'].replace("(","")
df['Text_cleaned'] = df['Text_cleaned'].replace(")","")
```

In [13]:
```
#Here we clean the content by removing all the  punctuation,
df['Text_clean'] = df['Text_clean'].str.replace('[^\w\s]','')
```

In [14]:
```
df['Text_clean'].head(1)
```

Out[14]:
```
0    i have bought several of the vitality canned dog food products and have found th
em all to be of good quality the product looks more like a stew than a processed meat
and it smells better my labrador is finicky and she appreciates this product better t
han  most
Name: Text_clean, dtype: object
```

To convert Digit into numbers

Import `re` library, make sure your column is of type `string`, and use `(?<!\S)\d+(?!\S)` to match sequences of digits that are between start/end of string and whitespace chars. If you want to only match whole entries that are all digits, you may use `^\d+$` regex.

In [16]:
```
def f(row):
    return num2words(row['Text_clean'])
```

In [17]:
```
import re
import num2words
import inflect
p = inflect.engine()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Input In [17], in <cell line: 2>()
      1 import re
----> 2 import num2words
      3 import inflect
      4 p = inflect.engine()

ModuleNotFoundError: No module named 'num2words'
```

In [18]:
```
#Here we clean the content by removing all the  numbers
df['Text_nonumber'] = df['Text_clean'].str.replace('\d+', '')
```

Here we clean the content convert Digit into numbers

df['Text_convnumber'] = df.iloc[:,3].astype(str).apply(lambda row: re.sub(r'(^\d+$)', lambda x: p.number_to_words(x.group()), row))

df['Text_convnumber'] = df['Text_clean'].apply(num2words)

In [19]:
```
# picked some arbitrary rows to review.
df[['Text_clean','Text_nonumber']][16:20]
```

Out[19]:

	Text_clean	Text_nonumber
16	i love eating them and they are good for watching tv and looking at movies it is not too sweet i like to transfer them to a zip lock baggie so they stay fresh so i can take my time eating them	i love eating them and they are good for watching tv and looking at movies it is not too sweet i like to transfer them to a zip lock baggie so they stay fresh so i can take my time eating them
17	i am very satisfied with my twizzler purchase i shared these with others and we have all enjoyed them i will definitely be ordering more	i am very satisfied with my twizzler purchase i shared these with others and we have all enjoyed them i will definitely be ordering more
18	twizzlers strawberry my childhood favorite candy made in lancaster pennsylvania by y s candies inc one of the oldest confectionery firms in the united states now a subsidiary of the hershey company the company was established in 1845 as young and smylie they also make apple licorice twists green color and blue raspberry licorice twists i like them allbr br i keep it in a dry cool place because is not recommended it to put it in the fridge according to the guinness book of records the longest licorice twist ever made measured 1200 feet 370 m and weighted 100 pounds 45 kg and was made by y s candies inc this recordbreaking twist became a guinness world record on july 19 1998 this product is kosher thank you	twizzlers strawberry my childhood favorite candy made in lancaster pennsylvania by y s candies inc one of the oldest confectionery firms in the united states now a subsidiary of the hershey company the company was established in as young and smylie they also make apple licorice twists green color and blue raspberry licorice twists i like them allbr br i keep it in a dry cool place because is not recommended it to put it in the fridge according to the guinness book of records the longest licorice twist ever made measured feet m and weighted pounds kg and was made by y s candies inc this recordbreaking twist became a guinness world record on july this product is kosher thank you
19	candy was delivered very fast and was purchased at a reasonable price i was home bound and unable to get to a store so this was perfect for me	candy was delivered very fast and was purchased at a reasonable price i was home bound and unable to get to a store so this was perfect for me

In [20]:
```
df['Text_clean'].head(1)
```

Out[20]:
```
0    i have bought several of the vitality canned dog food products and have found th
em all to be of good quality the product looks more like a stew than a processed meat
and it smells better my labrador is finicky and she appreciates this product better t
han  most
Name: Text_clean, dtype: object
```

In [21]:
```
#Here we clean the content by removing all the  white space,
df['Text_clean'] = df['Text_clean'].str.strip()
```

In [22]:
```
df['Text_clean'].head(1)
```

Out[22]:
```
0    i have bought several of the vitality canned dog food products and have found th
em all to be of good quality the product looks more like a stew than a processed meat
and it smells better my labrador is finicky and she appreciates this product better t
han  most
Name: Text_clean, dtype: object
```

In [23]:
```
df['words'] = df.Text_clean.str.strip().str.split('[\W_]+')
```

In [24]:
```
df['words'].head(1)
```

Out[24]:
```
0    [i, have, bought, several, of, the, vitality, canned, dog, food, products, and,
have, found, them, all, to, be, of, good, quality, the, product, looks, more, like,
a, stew, than, a, processed, meat, and, it, smells, better, my, labrador, is, finick
y, and, she, appreciates, this, product, better, than, most]
Name: words, dtype: object
```

In [25]:
```
#pd.set_option('display.max_colwidth', -1) # Setting this so we can see the full conte

# picked some arbitrary rows to review.
df[['Text_clean','words']][16:20]
```

Out[25]:

	Text_clean	words
16	i love eating them and they are good for watching tv and looking at movies it is not too sweet i like to transfer them to a zip lock baggie so they stay fresh so i can take my time eating them	[i, love, eating, them, and, they, are, good, for, watching, tv, and, looking, at, movies, it, is, not, too, sweet, i, like, to, transfer, them, to, a, zip, lock, baggie, so, they, stay, fresh, so, i, can, take, my, time, eating, them]
17	i am very satisfied with my twizzler purchase i shared these with others and we have all enjoyed them i will definitely be ordering more	[i, am, very, satisfied, with, my, twizzler, purchase, i, shared, these, with, others, and, we, have, all, enjoyed, them, i, will, definitely, be, ordering, more]
18	twizzlers strawberry my childhood favorite candy made in lancaster pennsylvania by y s candies inc one of the oldest confectionery firms in the united states now a subsidiary of the hershey company the company was established in 1845 as young and smylie they also make apple licorice twists green color and blue raspberry licorice twists i like them allbr br i keep it in a dry cool place because is not recommended it to put it in the fridge according to the guinness book of records the longest licorice twist ever made measured 1200 feet 370 m and weighted 100 pounds 45 kg and was made by y s candies inc this recordbreaking twist became a guinness world record on july 19 1998 this product is kosher thank you	[twizzlers, strawberry, my, childhood, favorite, candy, made, in, lancaster, pennsylvania, by, y, s, candies, inc, one, of, the, oldest, confectionery, firms, in, the, united, states, now, a, subsidiary, of, the, hershey, company, the, company, was, established, in, 1845, as, young, and, smylie, they, also, make, apple, licorice, twists, green, color, and, blue, raspberry, licorice, twists, i, like, them, allbr, br, i, keep, it, in, a, dry, cool, place, because, is, not, recommended, it, to, put, it, in, the, fridge, according, to, the, guinness, book, of, records, the, longest, licorice, twist, ever, made, measured, 1200, feet, 370, m, and, weighted, 100, ...]
19	candy was delivered very fast and was purchased at a reasonable price i was home bound and unable to get to a store so this was perfect for me	[candy, was, delivered, very, fast, and, was, purchased, at, a, reasonable, price, i, was, home, bound, and, unable, to, get, to, a, store, so, this, was, perfect, for, me]

Getting Word Counts

Now that we have a list of the words used in the text we can write a quick loop to make a word count dataframe.

In [26]:
```
words_list = df['Text_clean'].tolist()
raw_text = ''.join(words_list)
```

In [27]:
```
all_words = raw_text.split()
```

In [28]:
```
type(words_list)
```

Out[28]:
```
list
```

In [29]:
```
all_words[:10]
```

Out[29]:
```
['i',
 'have',
 'bought',
 'several',
 'of',
 'the',
 'vitality',
 'canned',
 'dog',
 'food']
```

In [30]:
```
### We'll make a temporary dictionary to hold the words
### Dictionaries are quite useful for word counts
word_dict = {}

## For each word in the text
for word in all_words:
    # if the word wasn't already in the dictionary
    if word not in word_dict.keys():
        # add it
        word_dict[word] = 1
    # otherwise
    else:
        # add 1 to the existing count
        word_dict[word] = word_dict[word] + 1

## NOTE In the future we could write this as a function
## then anytime we want a word count we just need to call the
## function!


# Let's examine the dictionary
word_dict
```

Out[30]:

```
{'i': 1978,
 'have': 571,
 'bought': 83,
 'several': 28,
 'of': 1329,
 'the': 3099,
 'vitality': 1,
 'canned': 9,
 'dog': 46,
 'food': 208,
 'products': 41,
 'and': 2096,
 'found': 92,
 'them': 378,
 'all': 271,
 'to': 1517,
 'be': 279,
 'good': 303,
 'quality': 71,
 'product': 189,
 'looks': 16,
 'more': 176,
 'like': 407,
 'a': 1901,
 'stew': 2,
 'than': 199,
 'processed': 4,
 'meat': 16,
 'it': 1229,
 'smells': 4,
 'better': 116,
 'my': 603,
 'labrador': 1,
 'is': 1138,
 'finicky': 3,
 'she': 69,
 'appreciates': 1,
 'this': 859,
 'mostproduct': 1,
 'arrived': 29,
 'labeled': 2,
 'as': 433,
 'jumbo': 1,
 'salted': 10,
 'peanutsthe': 1,
 'peanuts': 11,
 'were': 197,
 'actually': 48,
 'small': 56,
 'sized': 12,
 'unsalted': 10,
 'not': 471,
 'sure': 53,
 'if': 234,
 'was': 467,
 'an': 137,
 'error': 2,
 'or': 290,
 'vendor': 4,
 'intended': 1,
```

```
'represent': 1,
'jumbothis': 1,
'confection': 1,
'that': 609,
'has': 209,
'been': 103,
'around': 35,
'few': 48,
'centuries': 1,
'light': 27,
'pillowy': 1,
'citrus': 4,
'gelatin': 1,
'with': 564,
'nuts': 5,
'in': 888,
'case': 53,
'filberts': 1,
'cut': 28,
'into': 35,
'tiny': 15,
'squares': 3,
'then': 81,
'liberally': 1,
'coated': 4,
'powdered': 5,
'sugar': 125,
'mouthful': 2,
'heaven': 2,
'too': 164,
'chewy': 7,
'very': 267,
'flavorful': 20,
'highly': 37,
'recommend': 67,
'yummy': 14,
'treat': 18,
'you': 546,
'are': 690,
'familiar': 3,
'story': 3,
'cs': 1,
'lewis': 1,
'lion': 1,
'witch': 1,
'wardrobe': 2,
'seduces': 1,
'edmund': 1,
'selling': 9,
'out': 169,
'his': 38,
'brother': 3,
'sisters': 3,
'witchif': 1,
'looking': 57,
'for': 858,
'secret': 3,
'ingredient': 11,
'robitussin': 1,
'believe': 23,
```

```
'got': 56,
'addition': 9,
'root': 4,
'beer': 5,
'extract': 9,
'ordered': 50,
'which': 117,
'made': 64,
'some': 167,
'cherry': 10,
'soda': 18,
'flavor': 259,
'medicinalgreat': 1,
'taffy': 9,
'at': 300,
'great': 260,
'price': 124,
'there': 125,
'wide': 2,
'assortment': 2,
'delivery': 19,
'quick': 17,
'your': 123,
'lover': 5,
'deali': 1,
'wild': 2,
'hair': 2,
'five': 8,
'pound': 8,
'bag': 178,
'enjoyable': 4,
'many': 82,
'flavors': 87,
'watermelon': 6,
'melon': 1,
'peppermint': 4,
'grape': 7,
'etc': 13,
'only': 137,
'complaint': 3,
'bit': 68,
'much': 142,
'redblack': 1,
'licoriceflavored': 1,
'pieces': 29,
'just': 248,
'particular': 8,
'favorites': 9,
'between': 10,
'me': 166,
'kids': 16,
'husband': 26,
'lasted': 4,
'two': 56,
'weeks': 13,
'would': 149,
'brand': 125,
'delightful': 5,
'treatthis': 1,
'saltwater': 1,
```

```
'had': 221,
'soft': 18,
'each': 41,
'candy': 22,
'individually': 4,
'wrapped': 8,
'well': 94,
'none': 12,
'candies': 6,
'stuck': 9,
'together': 18,
'did': 69,
'happen': 5,
'expensive': 34,
'version': 19,
'fralingers': 1,
'served': 7,
'beachthemed': 1,
'party': 10,
'everyone': 21,
'loved': 40,
'itthis': 8,
'so': 386,
'amazing': 26,
'definitely': 48,
'buying': 33,
'satisfyingright': 1,
'now': 112,
'im': 113,
'mostly': 11,
'sprouting': 1,
'cats': 33,
'can': 197,
'eat': 110,
'grass': 2,
'they': 554,
'love': 212,
'rotate': 1,
'wheatgrass': 1,
'rye': 1,
'toothis': 3,
'healthy': 30,
'their': 116,
'digestion': 2,
'also': 143,
'puppies': 3,
'eats': 10,
'her': 51,
'required': 1,
'amount': 38,
'every': 67,
'feedingi': 1,
'dont': 164,
'know': 84,
'its': 247,
'cactus': 2,
'tequila': 5,
'unique': 17,
'combination': 19,
'ingredients': 62,
```

```
'but': 546,
'flavour': 15,
'hot': 79,
'sauce': 45,
'makes': 42,
'one': 253,
'kind': 32,
'we': 226,
'picked': 8,
'up': 131,
'bottle': 34,
'once': 36,
'on': 409,
'trip': 4,
'brought': 9,
'back': 44,
'home': 31,
'us': 35,
'totally': 9,
'blown': 2,
'away': 32,
'when': 201,
'realized': 7,
'simply': 8,
'couldnt': 27,
'find': 122,
'anywhere': 11,
'our': 100,
'city': 4,
'bummedbr': 2,
'br': 449,
'because': 137,
'magic': 3,
'internet': 5,
'ecstatic': 2,
'itbr': 15,
'saucei': 4,
'mean': 12,
'really': 160,
'want': 65,
'tastelessly': 2,
'burns': 3,
'throat': 2,
'grab': 7,
'picante': 2,
'gourmet': 13,
'de': 4,
'inclan': 2,
'realize': 6,
'taste': 269,
'will': 203,
'never': 62,
'use': 140,
'any': 115,
'other': 191,
'saucebr': 3,
'thank': 22,
'personal': 7,
'incredible': 4,
'serviceone': 1,
```

```
'boys': 1,
'needed': 14,
'lose': 9,
'weight': 14,
'didnt': 55,
'put': 46,
'floor': 2,
'chubby': 2,
'guy': 3,
'proteinrich': 1,
'no': 159,
'byproduct': 1,
'higher': 14,
'where': 31,
'skinny': 2,
'boy': 4,
'jump': 1,
'sits': 2,
'going': 43,
'stale': 13,
'both': 36,
'go': 61,
'losing': 1,
'about': 154,
'ounce': 16,
'weekmy': 1,
'happily': 2,
'eating': 61,
'felidae': 12,
'platinum': 3,
'years': 43,
'new': 45,
'shape': 8,
'different': 46,
'tried': 107,
'first': 86,
'bowls': 3,
'sit': 2,
'full': 36,
'kitties': 2,
'touch': 7,
'ive': 115,
'noticed': 9,
'similar': 12,
'reviews': 23,
'related': 3,
'formula': 10,
'changes': 4,
'past': 16,
'unfortunately': 21,
'need': 48,
'eatgood': 1,
'these': 486,
'came': 44,
'securely': 2,
'packed': 9,
'fresh': 65,
'delicious': 63,
'twizzlersthe': 1,
'strawberry': 23,
```

```
'twizzlers': 5,
'guilty': 3,
'pleasure': 4,
'six': 11,
'pounds': 9,
'while': 45,
'son': 33,
'imy': 1,
'daughter': 26,
'loves': 52,
'shipment': 9,
'hit': 12,
'spot': 3,
'exactly': 18,
'what': 109,
'expectsix': 1,
'packages': 2,
'twizzlersi': 1,
'watching': 3,
'tv': 2,
'movies': 1,
'sweet': 79,
'transfer': 1,
'zip': 3,
'lock': 1,
'baggie': 1,
'stay': 15,
'take': 47,
'time': 137,
'themi': 5,
'am': 109,
'satisfied': 6,
'twizzler': 1,
'purchase': 27,
'shared': 5,
'others': 21,
'enjoyed': 15,
'ordering': 26,
'moretwizzlers': 1,
'childhood': 2,
'favorite': 79,
'lancaster': 1,
'pennsylvania': 1,
'by': 120,
'y': 2,
's': 2,
'inc': 5,
'oldest': 3,
'confectionery': 2,
'firms': 1,
'united': 7,
'states': 8,
'subsidiary': 1,
'hershey': 4,
'company': 34,
'established': 1,
'1845': 1,
'young': 3,
'smylie': 1,
'make': 104,
```

```
'apple': 15,
'licorice': 11,
'twists': 2,
'green': 36,
'color': 23,
'blue': 11,
'raspberry': 3,
'allbr': 5,
'keep': 41,
'dry': 23,
'cool': 5,
'place': 14,
'recommended': 10,
'fridge': 7,
'according': 4,
'guinness': 2,
'book': 5,
'records': 1,
'longest': 1,
'twist': 5,
'ever': 81,
'measured': 1,
'1200': 1,
'feet': 1,
'370': 1,
'm': 2,
'weighted': 1,
'100': 19,
'45': 5,
'kg': 1,
'recordbreaking': 1,
'became': 7,
'world': 13,
'record': 2,
'july': 1,
'19': 3,
'1998': 1,
'kosher': 1,
'youcandy': 1,
'delivered': 12,
'fast': 23,
'purchased': 37,
'reasonable': 6,
'bound': 1,
'unable': 4,
'get': 148,
'store': 64,
'perfect': 52,
'memy': 1,
'addict': 5,
'weve': 12,
'times': 16,
'from': 212,
'amazon': 114,
'government': 1,
'employees': 2,
'living': 6,
'overseas': 2,
'cant': 52,
'country': 9,
```

```
'assigned': 1,
'theyve': 2,
'always': 53,
'tasty': 57,
'arrive': 6,
'timely': 5,
'manneri': 1,
'who': 69,
'currently': 2,
'he': 84,
'apparently': 3,
'staff': 2,
'likes': 23,
'alsobr': 2,
'generous': 2,
'amounts': 5,
'16ounce': 3,
'worth': 24,
'hrefhttpwwwamazoncomgpproductb001gvisjmtwizzlers': 2,
'bags': 118,
'pack': 52,
'6ai': 1,
'remember': 6,
'kid': 3,
'hasnt': 3,
'dropped': 3,
'still': 75,
'superb': 6,
'wont': 37,
'disappointed': 21,
'withi': 1,
'after': 84,
'watchers': 3,
'craving': 5,
'iti': 9,
'lived': 5,
'over': 54,
'7': 9,
'yrs': 2,
'miss': 7,
'visit': 3,
'someone': 13,
'visits': 1,
'stock': 15,
'say': 50,
'yumbr': 1,
'sell': 10,
'mexico': 1,
'faithful': 1,
'buyer': 3,
'often': 11,
'able': 21,
'buy': 139,
'right': 59,
'nowproduct': 1,
'received': 23,
'advertisedbr': 1,
'6athe': 2,
'red': 23,
'plan': 10,
```

```
'againi': 9,
'glad': 20,
'carried': 5,
'batteries': 1,
'hard': 55,
'finding': 11,
'elsewhere': 4,
'such': 22,
'size': 58,
'garage': 2,
'door': 7,
'openerbr': 1,
'deal': 28,
'pricei': 2,
'mum': 1,
'diabetic': 2,
'needs': 7,
'watch': 6,
'intake': 6,
'father': 4,
'chooses': 1,
'limit': 4,
'unnecessary': 1,
'shes': 7,
'tooth': 3,
'toffees': 1,
'guess': 9,
'theyre': 49,
'sugarfree': 8,
'pretty': 41,
'guilt': 2,
'free': 42,
'impressed': 6,
'myself': 21,
'w': 6,
'dark': 17,
'chocolate': 86,
'office': 7,
'ill': 27,
'instead': 29,
'snacking': 9,
'sugary': 3,
'sweetsbr': 1,
'excellenti': 1,
'servicei': 1,
'huge': 11,
'coffee': 130,
'fan': 23,
'however': 48,
'mother': 11,
'little': 104,
'machine': 2,
'talked': 1,
'trying': 23,
'latte': 1,
'macciato': 1,
'shop': 9,
'most': 62,
'usually': 30,
'noncoffee': 1,
```

```
'drinkerbr': 1,
'dolche': 1,
'guesto': 1,
'super': 19,
'easy': 45,
'prepares': 1,
'coffeelattecappuccinoetc': 1,
'less': 63,
'minute': 9,
'water': 73,
'heated': 1,
'dolce': 1,
'gusto': 1,
'anyone': 23,
'iam': 2,
'getting': 23,
'myselfthis': 1,
'offer': 4,
'thanks': 23,
'productbr': 4,
'staralmccanns': 1,
'instant': 29,
'oatmeal': 35,
'must': 22,
'scrape': 1,
'three': 23,
'minutes': 16,
'prepare': 7,
'escaping': 1,
'fact': 28,
'even': 94,
'best': 172,
'nowhere': 1,
'near': 7,
'requiring': 2,
'stovetop': 2,
'preparation': 3,
'mccanns': 16,
'gets': 13,
'organic': 90,
'allnatural': 2,
'brands': 50,
'varieties': 12,
'variety': 29,
'prepared': 1,
'microwave': 4,
'adding': 9,
'boiling': 3,
'convenient': 11,
'extreme': 5,
'issuebr': 1,
'actual': 10,
'cane': 9,
'high': 41,
'fructose': 2,
'corn': 21,
'syrup': 22,
'helped': 6,
'decide': 2,
'real': 37,
```

```
'tastes': 56,
'harmful': 1,
'stuff': 40,
'thing': 40,
'do': 117,
'though': 34,
'thickeners': 1,
'oats': 20,
'plus': 26,
'heat': 15,
'should': 36,
'creamy': 6,
'without': 67,
'guar': 3,
'gum': 11,
'convenience': 10,
'maybe': 19,
'why': 30,
'sitting': 6,
'bowl': 7,
'becomes': 3,
'thick': 29,
'glueythis': 1,
'uses': 6,
'fructouse': 1,
'does': 42,
'sweetness': 18,
'doctors': 3,
'form': 13,
'cold': 12,
'morning': 19,
'steel': 3,
'cinnamon': 23,
'maple': 12,
'brown': 42,
'regular': 47,
'require': 2,
'doctoring': 1,
'tell': 18,
'apartinstant': 1,
'become': 11,
'soggy': 2,
'hits': 2,
'holds': 6,
'texture': 41,
'excellent': 42,
'same': 51,
'oat': 1,
'meal': 19,
'may': 37,
'longer': 20,
'eaten': 28,
'close': 8,
'second': 15,
'noninstant': 3,
'varietybr': 1,
'irish': 6,
'apples': 12,
'10count': 3,
'boxes': 25,
```

```
'6mccanns': 1,
'6br': 2,
'steelcut': 1,
'thought': 37,
'id': 30,
'give': 53,
'try': 110,
'hardy': 2,
'folks': 6,
'postbariatric': 1,
'surgery': 2,
'palatable': 4,
'easily': 10,
'digestible': 2,
'fiber': 8,
'bloatfor': 1,
'those': 48,
'celiac': 5,
'disease': 4,
'lifesaver': 1,
'could': 75,
'almost': 43,
'half': 34,
'grocery': 46,
'health': 24,
'flavorsbr': 4,
'thanksbr': 1,
'abbywhat': 1,
'else': 19,
'cup': 39,
'lowfat': 6,
'milk': 38,
'add': 56,
'raisinsnuke': 1,
'90': 3,
'seconds': 5,
'kroger': 3,
'tastier': 4,
'something': 48,
'mmm': 2,
'convenienti': 1,
'visiting': 4,
'friend': 8,
'nate': 2,
'storage': 2,
'room': 8,
'packet': 6,
'suggested': 1,
'own': 27,
'stash': 3,
'sometimes': 20,
'dose': 1,
'chance': 5,
'ended': 9,
'cinn': 1,
'tastefull': 1,
'goes': 19,
'oj': 1,
'slice': 4,
'toast': 2,
```

```
'ready': 8,
'worldor': 1,
'day': 51,
'least': 30,
'jerry': 1,
'reithi': 1,
'wife': 5,
'reccomended': 1,
'happy': 37,
'happybr': 1,
'hrefhttpwwwamazoncomgpproductb001eo5qw8mccanns': 1,
'packs': 8,
'greatbr': 3,
'030': 1,
'cents': 3,
'per': 40,
'understand': 4,
'earth': 3,
'isnt': 27,
'upbr': 2,
'terrific': 7,
'followed': 4,
'tired': 15,
'ole': 3,
'boil': 3,
'pot': 5,
'empty': 8,
'2': 57,
'pour': 8,
'expand': 1,
'2x': 1,
'sizebr': 2,
'takes': 19,
'preparebr': 1,
'extremely': 12,
'cheapmccanns': 1,
'connoisseur': 1,
'whether': 2,
'raw': 19,
'pellet': 1,
'state': 9,
'cooks': 5,
'hour': 8,
'sloth': 1,
'addled': 1,
'done': 16,
'under': 13,
'thats': 38,
'beauty': 3,
'available': 26,
'regularbr': 1,
'allows': 2,
'explored': 1,
'giving': 10,
'experience': 15,
'difference': 9,
'wellknown': 1,
'oatmeals': 1,
'personally': 9,
'thicker': 10,
```

```
'body': 14,
'top': 24,
'here': 56,
'america': 2,
'tends': 1,
'liquidy': 1,
'experiment': 5,
'1300watt': 1,
'twentyseven': 1,
'handle': 2,
'how': 80,
'usebr': 1,
'bad': 57,
'consider': 4,
'offering': 3,
'lot': 44,
'youll': 26,
'end': 16,
'tencount': 1,
'whole': 56,
'family': 35,
'oatmealeaters': 1,
'youre': 24,
'single': 14,
'person': 12,
'alone': 2,
'oatmeali': 1,
'save': 32,
'300': 4,
'boxbr': 3,
'healthymccanns': 1,
'choice': 7,
'overly': 9,
'breakfast': 23,
'excellentwe': 1,
'cook': 12,
'oftenbr': 1,
'convenientbr': 1,
'anything': 32,
'keeps': 6,
'regularly': 4,
'thingthis': 1,
'seems': 27,
'wholesome': 2,
'supermarket': 10,
'somewhat': 15,
'mushy': 3,
'doesnt': 43,
'quite': 31,
'either': 27,
'pass': 3,
'muster': 1,
'probably': 30,
'againgood': 2,
'wouldnt': 16,
'follow': 2,
'directions': 3,
'package': 26,
'since': 74,
'comes': 18,
```

```
'soupy': 1,
'ofthe': 1,
'see': 38,
'differce': 1,
'oaker': 1,
'mushyi': 1,
'fine': 19,
'added': 27,
'ok': 20,
'satisfying': 10,
'order': 75,
'again': 67,
'sugarthis': 1,
'big': 36,
'box': 79,
'stores': 39,
'nothing': 29,
'carbs': 1,
'sugars': 6,
'money': 20,
'tastethis': 1,
'quaker': 2,
'way': 62,
'gogot': 1,
'bloody': 3,
'mary': 2,
'mix': 53,
'seller': 12,
'advertising': 3,
'workedlol': 1,
'buddies': 1,
'yet': 13,
'burn': 7,
'mouth': 28,
'forever': 3,
'nice': 56,
'temp': 2,
'usthis': 1,
'wasnt': 14,
'last': 30,
'looked': 15,
'vermont': 3,
'weston': 1,
'along': 14,
'jaw': 2,
'harp': 2,
'cranberry': 1,
'horseradish': 1,
'fartless': 1,
'black': 27,
'bean': 15,
'salsa': 21,
'cider': 1,
'jelly': 4,
'newtons': 1,
'cradle': 1,
'art': 1,
'motion': 1,
'staple': 3,
'syrupbr': 1,
```

```
 'ass': 2,
 'kickin': 1,
 'activate': 1,
 'perspiration': 1,
 'glands': 1,
 'behind': 3,
 'ears': 1,
 'arms': 2,
 'requires': 2,
 'beverage': 3,
 'advertised': 8,
 'glass': 20,
 'kleenex': 1,
 'nose': 4,
 'run': 14,
 'look': 26,
 'ordinary': 2,
 'already': 11,
 'ideas': 1,
 'work': 33,
 'suspect': 3,
 'people': 41,
 'hitting': 1,
 'goodies': 2,
 'absence': 2,
 'especially': 32,
 'colleague': 1,
 'greg': 2,
 'earliest': 1,
 'opportunity': 3,
 'contents': 5,
 'planters': 1,
 'whose': 5,
 'crying': 1,
 'running': 8,
 'returnbr': 1,
 'shaken': 1,
 'ensure': 3,
 'spices': 15,
 'evenly': 1,
 ...}
```

In [31]:
```
# Now import pandas
import pandas as pd
```

In [32]:
```
print(pd.__version__)
```

```
1.4.2
```

In [33]:
```
# Now make the dataframe
# Note .count() is a native method for a dataframe object
# this is why I used times_used instead!
pa_word_counts = pd.DataFrame({'word':list(word_dict.keys()),
                               'times_used':list(word_dict.values())})
```

In [34]:
```
pa_word_counts.sort_values('times_used',ascending=False).head(25)
```

Out[34]:

	word	times_used
5	the	3099
11	and	2096
0	i	1978
23	a	1901
15	to	1517
4	of	1329
28	it	1229
33	is	1138
75	in	888
37	this	859
115	for	858
98	are	690
63	that	609
31	my	603
1	have	571
73	with	564
215	they	554
97	you	546
240	but	546
3401	chips	537
352	these	486
51	not	471
54	was	467
270	br	449
41	as	433

Great!

As a note, you might think it's silly that we care about how many times the word `the` is used. Hold onto that thought for the next notebook(s).

Practice

Okay I've been talking a lot, now is your time to practice. I've included an excerpt from IMDB Dataset of 50K Movie Reviews in the Data Folder as well! This file is called `IMDB Dataset.csv`.

I have reduced it into a smaller one called `Movie_Review.csv`

You're job is to produce a word count dataframe using what we learned above. This should take 5-10 minutes.

In [68]:
```
# import libraries, adjust settings
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
panda.set_option('display.max_colwidth', -1)
import re
```

In [70]:
```
# read in data
df = pd.read_csv('Movie_Review.csv')

# split strings
df['review'].head(2).str.split()

# make strings lowercase
df['Text_clean'] = df['review'].str.lower()

# remove punctuation strings
df['Text_cleaned'] = df['Text_clean'].replace(",","")
df['Text_cleaned'] = df['Text_cleaned'].replace(".","")
df['Text_cleaned'] = df['Text_cleaned'].replace("!","")
df['Text_cleaned'] = df['Text_cleaned'].replace("?","")
df['Text_cleaned'] = df['Text_cleaned'].replace("\'","")
df['Text_cleaned'] = df['Text_cleaned'].replace('\"',"")
df['Text_cleaned'] = df['Text_cleaned'].replace("-"," ")
df['Text_cleaned'] = df['Text_cleaned'].replace("(","")
df['Text_cleaned'] = df['Text_cleaned'].replace(")","")

# remove punctuation from strings
df['Text_clean'] = df['Text_clean'].str.replace('[^\w\s]','')

# convert digits to strings
def f(row):
    return num2words(row['Text_clean'])
df['Text_nonumber'] = df['Text_clean'].str.replace('\d+', '')

# remove whitespace
df['Text_clean'] = df['Text_clean'].str.strip()

# split words
df['words'] = df.Text_clean.str.strip().str.split('[\W_]+')
```

In [71]:
```
# constructing word count dataframe
words_list = df['Text_clean'].tolist()
raw_text = ''.join(words_list)
all_words = raw_text.split()

# constructing word count dictionary
word_dict = {}
for word in all_words:
    # if the word wasn't already in the dictionary
    if word not in word_dict.keys():
        # add it
        word_dict[word] = 1
    # otherwise
    else:
```

```
        # add 1 to the existing count
        word_dict[word] = word_dict[word] + 1
```

In [73]:
```
# test and display
###print(word_dict)
pa_word_counts = pd.DataFrame({'word':list(word_dict.keys()),
                               'times_used':list(word_dict.values())})
pa_word_counts.sort_values('times_used',ascending=False).head(25)
```

Out[73]:

	word	times_used
2	the	13302
37	and	6401
49	a	6306
1	of	5884
60	to	5290
22	is	4042
43	in	3675
117	i	2929
21	this	2916
67	it	2857
7	that	2607
28	br	2302
34	was	1916
20	as	1766
26	with	1761
51	for	1698
359	movie	1674
145	but	1601
371	film	1451
78	on	1326
179	you	1299
48	not	1209
238	his	1181
18	are	1171
88	have	1076

Clear Limitations of Built-In `str` Methods

Okay so we've seen how useful of the box str methods can be, but as was the case with punctuation clean up, they have their weaknesses as well.

For another example of why we might want fancier tools we'll do another quick practice.

Try to take the excerpt of Harry Potter and the Prisoner of Azkaban and break it into unique sentences. Let's take 5 minutes on this.

In [ ]:
```
## Code here
```

In [ ]:
```
## Code here
```

In [ ]:
```

```

- What Happened?
- What are some issues you ran into?

Conclusions

While some of you probably were already quite familiar with using str methods, it's good to review. Sometimes when cleaning data you'll want something quick and easy to code, and using some of the techniques we'll learn in the following notebooks may be a bit of overkill.

In [ ]:
```

```