Lecture: Extracting simple statistics from datasets
Learning objectives

In this lecture we will...

• Introduce data structures that help us to compile statistics (like "defaultdict")
• Compute simple statistics like counts, sums, and averages from data
Let's try to compute the following from the Amazon data:

- What is the average star rating?
- What is the distribution of star ratings?
- What fraction of purchases are verified?
- Which products are the most popular (purchases)?
- Which products have the highest average ratings?
First let's read the Amazon data into a list, exactly as we did in the previous lecture:

```python
In [1]:
import gzip
path = "datasets/amazon/amazon_reviews_us_Gift_Card_v1_00.tsv.gz"
f = gzip.open(path, 'rt')

In [2]:
import csv
reader = csv.reader(f, delimiter = '\t')

In [3]:
header = next(reader)

In [4]:
dataset = []
for line in reader:
    d = dict(zip(header, line))
    for field in ['helpful_votes', 'star_rating', 'total_votes']:
        d[field] = int(d[field])
    for field in ['verified_purchase', 'vine']:
        if d[field] == 'Y':
            d[field] = True
        else:
            d[field] = False
    dataset.append(d)
```
Code: Average rating and rating distribution

- Average rating can be computed straightforwardly with a list comprehension:

```
In [5]: ratings = [d['star_rating'] for d in dataset]
In [6]: sum(ratings) / len(ratings)
Out[6]: 4.731333018677096
```

- Rating distribution can be computed by using a dictionary to store counts:

```
In [7]: ratingCounts = {1: 0, 2: 0, 3: 0, 4: 0, 5: 0}
In [8]: for d in dataset:
   ...:     ratingCounts[d['star_rating']] += 1
In [9]: ratingCounts
Out[9]: {1: 4766, 2: 1560, 3: 3147, 4: 9808, 5: 129029}
```
Note that we counted ratings by initializing a dictionary with all zero counts:

```
In [7]: ratingCounts = {1: 0, 2: 0, 3: 0, 4: 0, 5: 0}
```

The "defaultdict" structure from the "collections" library allows us to automate this functionality, which is useful for counting different types of object.

Let's compute the rating distribution using defaultdict:

```
In [10]: from collections import defaultdict

In [11]: ratingCounts = defaultdict(int)

In [12]: for d in dataset:
   ratingCounts[d['star_rating']] += 1

In [13]: ratingCounts
```

```
Out[13]: defaultdict(int, {1: 4766, 2: 1560, 3: 3147, 4: 9808, 5: 129029})
```
Similarly we can use the defaultdict function to count verified vs. non-verified purchases.

```python
In [14]: verifiedCounts = defaultdict(int)

In [15]: for d in dataset:
   ...:     verifiedCounts[d['verified_purchase']] += 1

In [16]: verifiedCounts
Out[16]: defaultdict(<class 'int'>, {False: 13021, True: 135289})
```
• Again we can use defaultdict to determine product popularity (here we just want to count which products appear most in the dataset)

```python
In [17]: productCounts = defaultdict(int)

In [18]: for d in dataset:
    productCounts[d['product_id']] += 1

In [19]: counts = [(productCounts[p], p) for p in productCounts]

In [20]: counts.sort()

In [21]: counts[-10:]
```

```
Out[21]: [(2638, 'B064KNW000'),
         (2173, 'B0066AZG04'),
         (2630, 'BT06DDC7CE'),
         (2643, 'B004LLIKY2'),
         (3467, 'BT06DDC78K'),
         (3440, 'BT06CTOUNS'),
         (4283, 'B00IX11366'),
         (5034, 'BT06DDVMV0'),
         (6037, 'B00A4BG9D4'),
         (28705, 'B064LLIKVU')]
```

• Following this, we build a list of counts followed by product IDs, which we can sort to get the most popular
Here we need to compute the average rating for each product, which requires that we first construct the list of ratings for each product. This can also be done using defaultdict, with the "list" subclass:

```python
In [22]: ratingsPerProduct = defaultdict(list)
In [23]: for d in dataset:
   ratingsPerProduct[d['product_id']].append(d['star_rating'])
In [24]: averageRatingPerProduct = {}
   for p in ratingsPerProduct:
       averageRatingPerProduct[p] = sum(ratingsPerProduct[p]) / len(ratingsPerProduct[p])
```

We now have two data structures: one which stores the list of ratings for each product, and one which stores the average rating for each product.
• Now we can sort by ratings, and also filter to only include reasonably popular products:
Summary of concepts

- Saw how to compute simple statistics from datasets
- Introduced the "defaultdict" structure

On your own...

Try computing other statistics, e.g.
- Who are the most active users?
- What are the most commonly used words?
- What is the different in average rating between verified versus non-verified purchases?