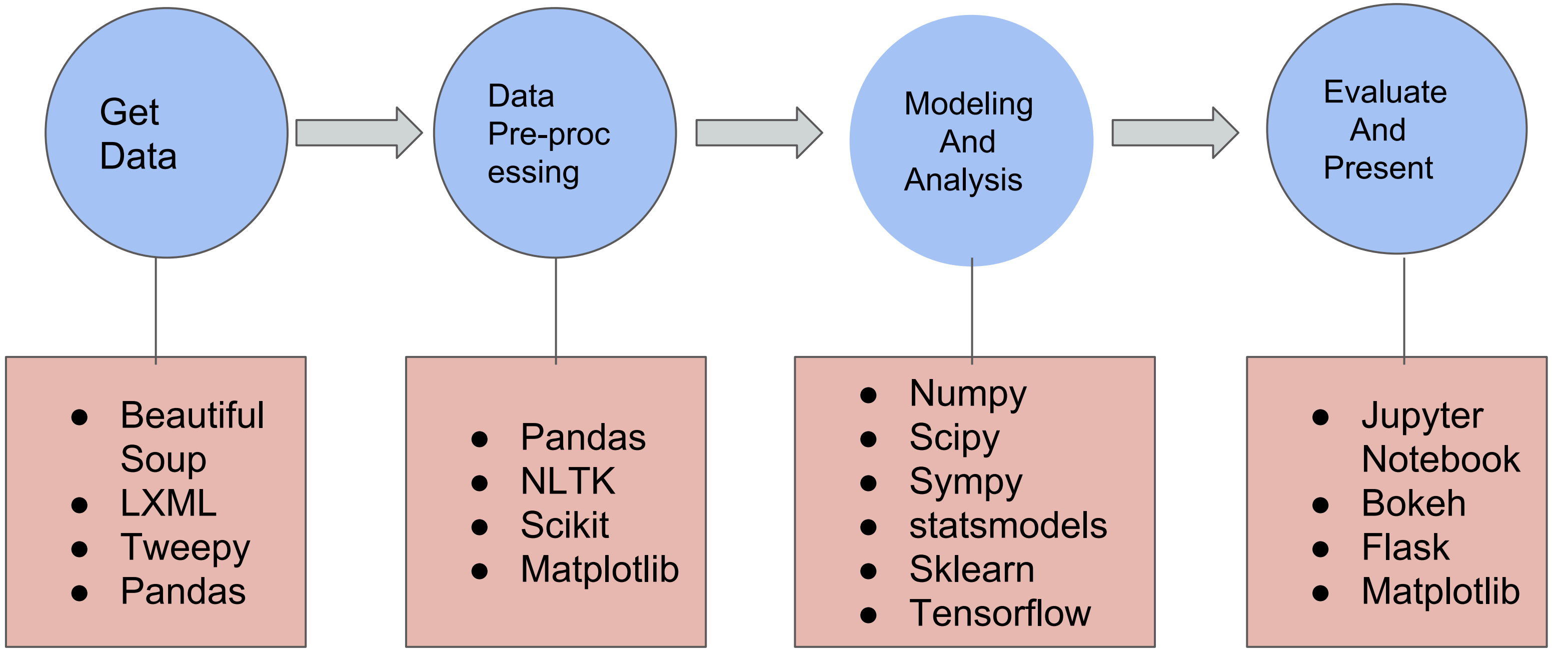
CSE 519: Data Science
Steven Skiena
Stony Brook University

Lecture 4: Python for Data Science II

Recap: Data Science with Python



Lecture Goals

- A real research project that uses Python for Data Science
- Learn more about each step by showcasing code and examples

Restaurant Ratings Across Sites

- Customers rely on online reviews and ratings to decide where to eat
- But the same restaurant seems to get different ratings and reviews across sites
- Is this true? What might be the reasons?

Motivating Example

- The Curry Club on Yelp, Google and FB

Yelp:
3.5/5

Google:
4.2/5

FB:
4.5/5

Motivating Example

- So I manually examined several restaurants...
  - For most of them, the rating is highest on Yelp and lowest on Facebook
  - Hypothesis: ratings on Yelp < Google < Facebook
- How can we verify this hypothesis?
  - We need to get more data in a systematic manner
- How should we interpret the discrepancy in ratings? What could be the reasons?
  - Perform data analysis to get more insights

Step 1: Get Data

- What data and how much data do we need?
  - We need ratings and reviews on Yelp, Google and Facebook for at least 1000 restaurants
  - Important: make sure the dataset is aligned – we should use the same set of restaurants for all review sites
- How to get these ratings and reviews?

Step 1: Get Data

- Always try to get data in the easiest way
  - Best: use an existing dataset published online
    - Google Dataset Search (new!): https://toolbox.google.com/datasetsearch
  - Good: using APIs or SDKs to download data
    - A lot of websites provide web APIs: Google, Facebook, Yelp, Reddit, Twitter, etc.
    - Typically they also come with rate limiting
  - Try to avoid: write your own Web crawler

Get Data (cont.)

- Yelp released datasets of reviews and ratings in their data challenge: https://www.yelp.com/dataset/challenge
  - We can sample some restaurants from Yelp as a starting point
- Find the corresponding review pages on Google and FB
  - Both sites provide APIs for this

Search for Places on Google

- https://developers.google.com/places/web-service/search#find-place-examples

Find Place examples

The following example shows a Find Place request for "Museum of Contemporary Art Australia", including the `photos`, `formatted_address`, `name`, `rating`, `opening_hours`, and `geometry` fields:

```
https://maps.googleapis.com/maps/api/place/findplacefromtext/json?input=Museum%20of%20Contemporary%20
```

The following example shows a Find Place request for "Mongolian Grill", using the `locationbias` parameter to prefer results within 2000 meters of the specified coordinates:

```
https://maps.googleapis.com/maps/api/place/findplacefromtext/json?input=mongolian%20grill&inputtype=t
```

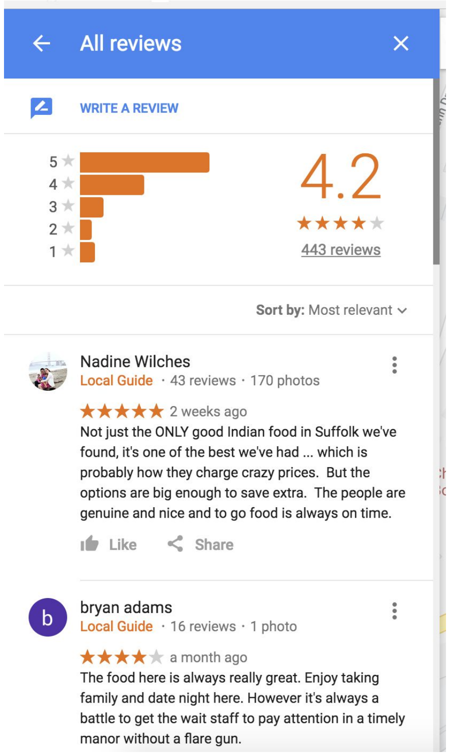
The following example shows a Find Place request for a phone number. Note that the international call prefix "+" has been encoded to %2B so that this request is a compliant URL. Left unencoded, the + prefix would be decoded to a space on the server, resulting in an invalid phone number lookup.

```
https://maps.googleapis.com/maps/api/place/findplacefromtext/json?input=%2B61293744000&inputtype=phon
```

Get Data (cont.)

- But how to get reviews from Google and FB?
  - Both sites have some data APIs, but no API for accessing the content or ratings of reviews
  - We have no choice but to crawl Web pages
- Sometimes this could be the most challenging and time-consuming part
  - so try your best to avoid crawling data

Crawl Web Pages



Crawl Web Pages: Typical Workflow

- Construct the URL that contains your data
  - This could be hard, you may need to read the source code of the Web pages to figure out the source of data
- Finally we have the following magical code for constructing the URL of review data on Google

Construct the URL

```python
def get_review_json_url(url):
    r = get_html(url, {})
    if r is None:
        logging.error('Cannot get review page for URL %s.' % url)
        return None
    content = r.content.decode('utf-8')
    # print (content)
    # get review count
    regex = re.compile('\d+ reviews')
    m = regex.search(content)
    try:
        review_count = int(re.compile('\d+').search(m.group()).group())
    except:
        logging.error('Cannot retrieve review count for URL %s.' % url)
        return None
    # construct paginated review url
    regex = re.compile('0x\w{16}:0x\w{16}')
    m = regex.search(content)
    try:
        secret_str = m.group().replace(':', '%3A')
        review_page0_url = 'https://www.google.com/maps/preview/reviews?authuser=0&hl=en&pb=' + \
            '!1s' + secret_str + '!2i0!3i10!4e6!7m4!2b1!3b1!5b1!6b1'
    except:
        logging.error('Cannot retrieve secret string for URL %s.' % url)
        return None
    return review_count, review_page0_url
```

Crawl Web Pages: Get HTML

- We use the requests library to get HTML

```
def get_html(url, header):
    retry_count = 5
    while retry_count > 0:
        try:
            proxy = get_proxy()
            html = requests.get(url, headers=header, proxies=proxy, timeout=20)
            return html
        except Exception as e:
            logging.error(e)
            retry_count -= 1
            logging.debug('Remove proxy %s' % proxy)
            delete_proxy(list(proxy.values())[0][7:])
    return None
```

Crawl Web Pages: Get HTML (cont.)

- Important: use HTTP proxies to avoid being blocked by the site
  - You will almost certainly be blocked without using proxies
  - Ideally, you should create a pool of available proxies
  - For each page to crawl, choose one of these proxies

IP Address	Port	Code	Country	Anonymity	Google	Https	Last Checked
178.128.103.83	3128	SG	Singapore	transparent	no	no	1 minute ago
194.87.148.222	1080	RU	Russian Federation	transparent	no	no	1 minute ago
181.129.53.106	8080	CO	Colombia	transparent	no	no	1 minute ago
186.96.104.18	32334	CO	Colombia	elite proxy	no	no	1 minute ago

Crawl Web Pages: Parsing HTML

- Use Beautiful Soup and/or regular expression
- Save the structured data to a .csv file

```
def extract_fields(review):
    try:
        reviewer_url = review.find("a", {"class": "_5pcq"}).get_attribute_list('href')[0]
    except:
        try:
            reviewer_url = review.find("a", {"class":"_5pb8 _1yz2 _8o _8s lfloat _ohe"}).get_attribute_list('href')[0]
        except:
            return None

    try:
        unwanted_periods = review.find("span", {"class": "text_exposed_hide"})
        if unwanted_periods:
            unwanted_periods.extract()
        regex = re.compile('_5pbx userContent')
        review_text_raw = review.find("div", {"class":regex})
        review_text = 'null' if review_text_raw is None else review_text_raw.get_text()
        reviewer_name = review.find("img", {"class":"_s0 _4ooo _5xib _5sq7 _44ma _rw img"}).get('aria-label')
        date = review.find("span", {"class": "timestampContent"}).get_text()
        stars = int(review.find("u").get_text()[0])
        return {"date": date, "stars": stars, "text": review_text,
                "user_name": reviewer_name, "reviewer_url": reviewer_url}
    except:
        return None
```

Crawl Web Pages

- This could take several days / months to finish
- For this project, it took two weeks to crawl all reviews for about 5,000 restaurants on Yelp, Google and FB

Step 2: Preprocessing

- Raw data needs to be pre-processed
- For this project, we mostly use Pandas for preprocessing
  - Also use NLTK for some easy text preprocessing

Look at Your Data

- Important: always do data spot checks
  - What is the format of your data?
  - What fields (features) are in your dataset?
  - Do you see any obvious problems with your data?

Look at Your Data

- Find potential problems in data: spot check + general experience with a specific type of data
- Do we have restaurants that have a lot of reviews on one site but very few (or even zero) on another site?
- If we do, should we filter them out? (yes!)
  - But does this filtering introduce bias?

Look at Your Data

- Since we are dealing with textual data, there are several things to pay attention to
- Does every review has textual content?
- Are the reviews always written in English?
- Are there special characters in reviews (such as emojis)? – this causes decoding problems

Look at Your Data

- Load the Google dataset

```
google_review_fname = '../data/google_reviews_2k_withnull.csv'
google_review = pd.read_csv(google_review_fname, sep='\t', index_col=0)
google_review['date'] = pd.to_datetime(google_review['date'])
google_review = google_review.sort_values(by=['date'])
len(google_review)
```

```
203344
```

Look at Your Data

```
In [121]: google_review
```

	business_id	date	stars	text	user_name	user_review_url	user_
0	zt9RLUIU32fZYOBh2L0NNQ	1990-12-31 00:00:00.000	2	The only thing peruvian about this place (I am...	Luis Patron	https://www.google.com/maps/contrib/1071361760...	10713617605959403730
1	BjH8Xepc10i6OhCDQdX6og	2002-10-15 00:00:00.000	4	NaN	Greg Stoddard	https://www.google.com/maps/contrib/1026703284...	10267032845644118959
2	MqYYYNA-ZYvV-1w5qcmMoA	2002-10-15 00:00:00.000	4	NaN	Greg Stoddard	https://www.google.com/maps/contrib/1026703284...	10267032845644118959
3	OPxWcHK96_cbmiF7legDnA	2003-04-04 00:00:00.000	4	NaN	George Chen	https://www.google.com/maps/contrib/1165535784...	11655357843668812205

We see reviews without textual content

Look at Your Data

- What portion of reviews do not have text?

Filter out Google reviews without text:

```
google_notnull = google_review[google_review['text'].notnull()]
len(google_notnull)
```

```
142132
```

- Only 70% of Google reviews have textual content

Look at Your Data

- We noticed that Yelp reviews seems to be longer
- What is the average review length (in words and sentences) on Yelp, Google and FB?

```
yelp_notnull['text_len'] = yelp_notnull['text'].apply(lambda x: len(x.split()))
```

```
count_sent = lambda doc: len(nltk.tokenize.sent_tokenize(doc))
yelp_notnull['sent_count'] = yelp_notnull.text.apply(count_sent)
google_notnull['sent_count'] = google_notnull.text.apply(count_sent)
fb_notnull['sent_count'] = fb_notnull.text.apply(count_sent)
```

Look at Your Data

- Check the distribution of # of sentences

```
In [334]: yelp_notnull['text_len'].describe()

Out[334]: count    483207.000000
          mean        113.226708
          std         107.333654
          min           1.000000
          25%          43.000000
          50%          79.000000
          75%         146.000000
          max        1021.000000
          Name: text_len, dtype: float64
```

Look at Your Data

- Basic statistics of our dataset

	Yelp	Google	Facebook
Total # of Reviews	469,642	203,344	240,238
# of Reviews with Text	469,642	142,132	82,270
Avg # of Reviews	233.0	98.0	115.8
Avg Length (sentences)	8.64	2.90	3.18
Avg Length (words)	113.2	30.6	31.6

Table 1: Statistics of the dataset used in our experiments.

Rating Distribution

- Recall that our goal is to see if the distribution of restaurant ratings is different across sites
    - We should plot the distribution of ratings at per review level and per restaurant level and see the difference

Ratings Distribution: Per Review

```
tmp = yelp_review['stars'].value_counts()
review_score_count_yelp = pd.DataFrame({'stars': tmp.index, 'count': tmp.values / len(yelp_review)})
review_score_count_yelp['stars'] = review_score_count_yelp['stars'].apply(int)
```

```
tmp = google_review['stars'].value_counts()
review_score_count_google = pd.DataFrame({'stars': tmp.index, 'count': tmp.values / len(google_review)})
```

```
tmp = google_review['stars'].value_counts()
notnull_review_score_count_google = pd.DataFrame({'stars': tmp.index, 'count': tmp.values / len(google_review)})
```

```
tmp = fb_review['stars'].value_counts()
review_score_count_fb = pd.DataFrame({'stars': tmp.index, 'count': tmp.values / len(fb_review)})
```

```
tmp = pd.merge(review_score_count_yelp, review_score_count_google, on='stars', suffixes=('_yelp', '_google'))
review_score_count = pd.merge(tmp, review_score_count_fb, on='stars')
review_score_count = review_score_count.rename(columns={'count_yelp': 'Yelp', 'count_google': 'Google', 'count': 'FB',
                                                         'stars': 'Star Rating'})
review_score_count = review_score_count.sort_values(by=['Star Rating'])
```

```
review_score_count
```

	Yelp	Star Rating	Google	FB
3	0.102715	1	0.092051	0.045642
4	0.086511	2	0.045775	0.028863
2	0.130757	3	0.096079	0.072586
1	0.259863	4	0.217380	0.147483
0	0.420149	5	0.548715	0.705425

Ratings Distribution: Per Review

```
ax = review_score_count.plot(x='Star Rating', y=['Yelp', 'Google', 'FB'], kind='bar')
plt.setp(ax.get_legend().get_texts(), fontsize='14')
plt.xlabel('Review Star Rating', fontsize=14)
plt.ylabel('', fontsize=14)
plt.xticks(fontsize=12, rotation='horizontal')
plt.yticks(fontsize=12)
plt.savefig('../paper/figure/review-star-rating.pdf')
```

Ratings Distribution: Per Review

Obtain the Ratings of Restaurants

- To have similar plot at per restaurant level, we need to obtain the ratings of the restaurants

```
average_stars = google_review_groups[['business_id', 'stars']].mean().join(
                yelp_review_groups[['business_id', 'stars']].mean(),
                                   how='outer', lsuffix='_google', rsuffix='_yelp')
average_stars = average_stars.join(fb_review_groups[['business_id', 'stars']].mean(), how='outer')
average_stars = average_stars.rename(columns={'stars': 'stars_fb'}).reset_index()
average_stars
```

	business_id	stars_google	stars_yelp	stars_fb
0	--9e1ONYQuAa-CB_Rrw7Tw	4.238372	4.087113	4.000000
1	--q7kSBRb0vWC8lSkXFByA	3.937500	4.000000	3.944444
2	-8R_-EkGpUhBk55K9Dd4mg	4.269231	3.544444	4.571429
3	-AD5PiuJHgdUcAK-Vxao2A	3.936709	3.667925	4.470588
4	-BS4aZAQm9u41YnB9MUASA	4.187500	4.657534	4.909091
5	-Bf8BQ3yMk8U2f45r2DRKw	4.606264	3.921429	4.200000

Ratings Distribution: Per Restaurant

- But we have some problem with plotting the rating distribution per restaurant
  - The rating of a restaurant is continuous, so bar chart is not directly applicable here
  - However, we can still make a bar chart here by bucketing the restaurant ratings
  - Even better: plot its cumulative distribution function (CDF)
- Important: choose the right visualization type

Ratings Distribution: Per Restaurant

```
average_stars['stars_yelp'].hist(cumulative=True, normed=1, bins=200, histtype='step',
         linewidth=2, alpha=0.8, range=(1,5.009), label='Yelp')
average_stars['stars_google'].hist(cumulative=True, normed=1, bins=200, histtype='step',
         linewidth=2, alpha=0.8, range=(1,5.009), label='Google')
average_stars['stars_fb'].hist(cumulative=True, normed=1, bins=200, histtype='step',
         linewidth=2, alpha=0.8, range=(1,5.009), label='FB')
plt.xlabel('Restaurant Star Rating', fontsize=14)
plt.ylabel('CDF', fontsize=14)
plt.xticks(fontsize=12, rotation='horizontal')
plt.yticks(fontsize=12)
plt.xlim(1, 5.00)
plt.legend(bbox_to_anchor=(0., 0.98, 1., .100), loc=3,
           ncol=3, mode="expand", borderaxespad=0., fontsize=12)
plt.savefig('../paper/figure/star-rating-cdf.pdf')
```

Ratings Distribution: Per Restaurant

Ratings Distribution: Per Restaurant

- What do we learn from this CDF plot?

	From								
Site	Yelp			Google			FB		
Rating	Centile	To Google	To FB	Centile	To Yelp	To FB	Centile	To Yelp	To Google
2.5	12.0%	3.42	3.67	1.3%	1.62	2.43	1.40%	1.63	2.53
3.0	24.7%	3.76	4.05	4.8%	2.00	3.10	4.10%	1.93	2.96
3.5	43.0%	4.03	4.35	15.3%	2.64	3.82	10.1%	2.39	3.33
4.0	70.4%	4.38	4.65	41.6%	3.47	4.33	22.6%	2.95	3.70
4.5	95.0%	4.75	4.94	80.1%	4.17	4.75	56.4%	3.77	4.20
5.0	100.0%	5.00	5.00	100.0%	5.00	5.00	100.0%	5.00	5.00

Average Ratings

- Numbers are as important as plots
- Compute the average restaurant ratings

```
In [919]: average_stars.mean()

Out[919]: stars_google    4.041770
          stars_yelp      3.506936
          stars_fb        4.301911
          dtype: float64
```

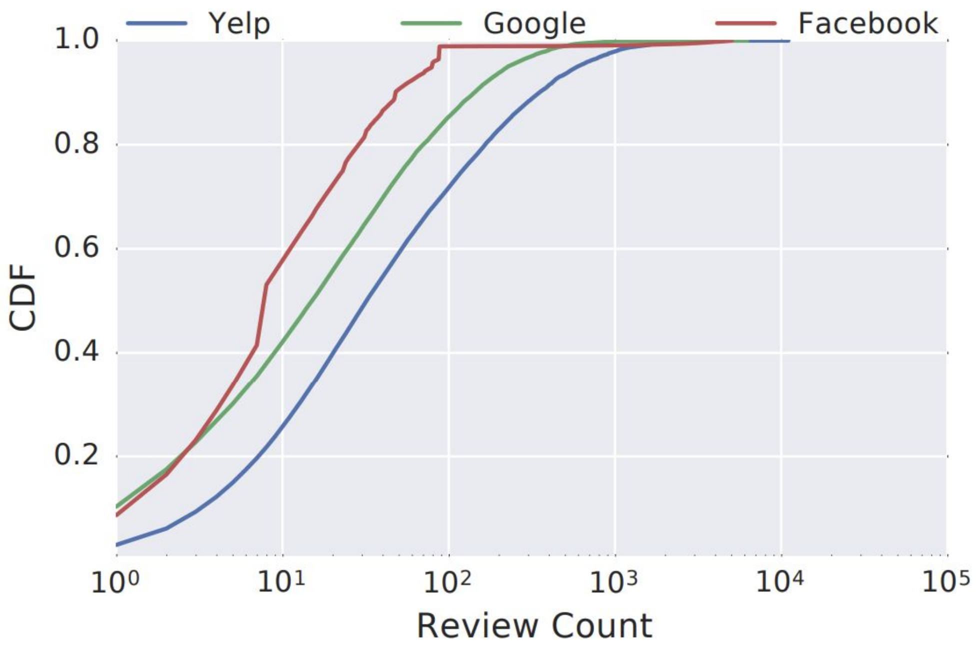
- Our hypothesis is correct!
- Ratings on Yelp < Google < Facebook

Why Different Ratings?

- We need some hypotheses to explain the discrepancy in ratings across sites
- Hypothesis 1: Yelp has a larger portion of productive reviewers than Google or Facebook, who are less likely to give extreme ratings.
- Hypothesis 2: Yelp reviews are more likely to be longer than Google or Facebook reviews, which are associated with lower ratings.

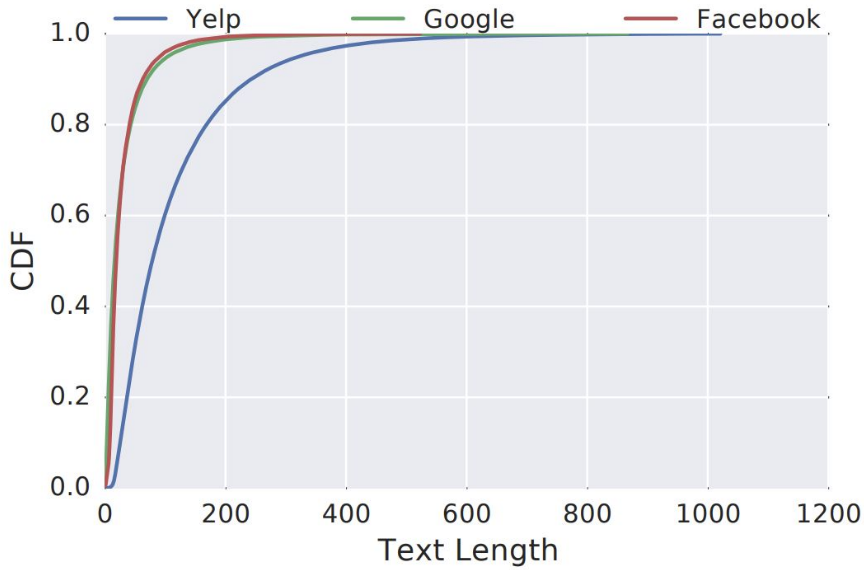
Verify our hypotheses

- CDF of review count



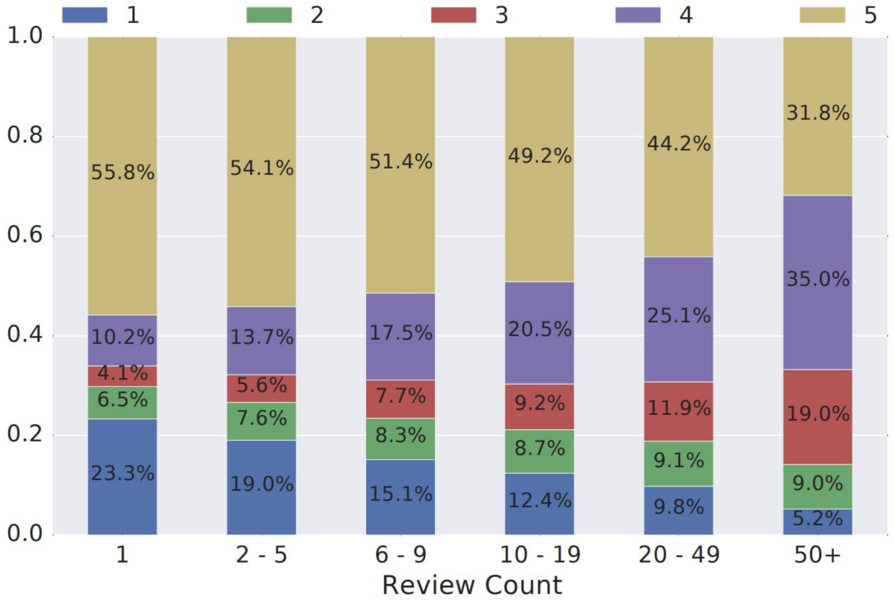
Verify our hypotheses

- CDF of review text length



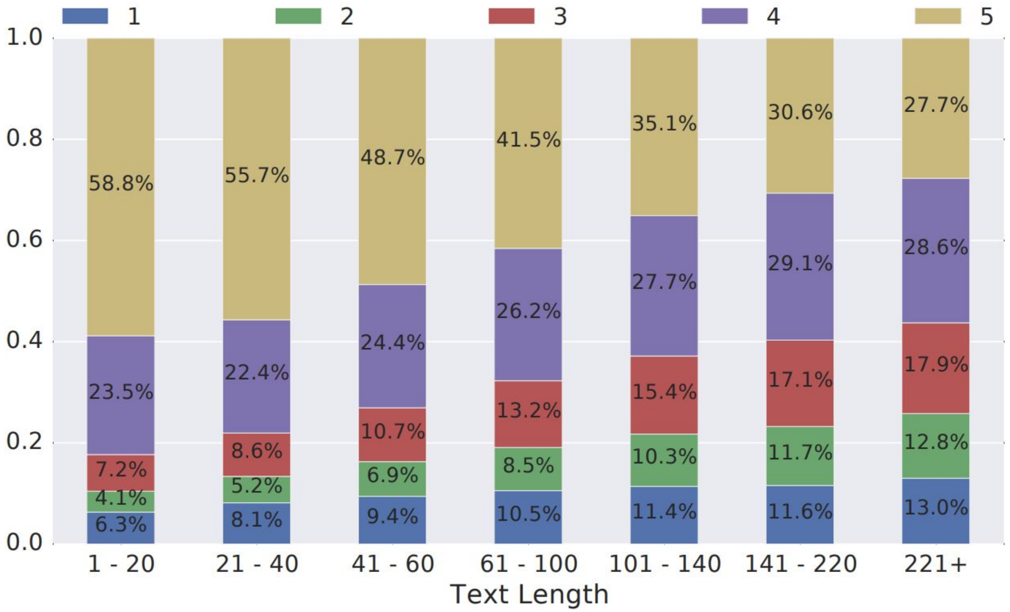
Verify our hypotheses

- What is the relationship between ratings and review count?



Verify our hypotheses

- What is the relationship between ratings and review text length?



Step 3: Modeling and Analysis

- To verify our hypotheses rigorously, we need a linear regression model that:
    - Uses the number of reviews written by the author of each review and the length of review as explanatory (independent) variables
    - Use the rating associated with each review as the dependent variable
    - Coefficients of the model help with verifying our hypothesis

Step 3: Modeling and Analysis

- Important: also control for the review site and the restaurant each review is written for
  - Concretely, we also add them as dependent variables in our model
- $Rating = \beta_1 * reviewCount + \beta_2 * textLen + \beta_3 * reviewSite + \beta_4 * RestaurantID + \beta_5$
- Fit the model and check the value of β_1 and β_2

Step 3: Modeling and Analysis

- Important: avoid reinventing the wheel
  - Very often, the modeling part requires you to write less than 20 lines of code
  - But tuning your model might takes some time
- Here we used the partial proportional odds model, which is a variation of linear regression
  - Designed for ordinal data (such as ratings)

Step 3: Modeling and Analysis

- Regression coefficients

Explanatory Variables		β
text_len		-0.49
review_count	Rating	
	1 vs 2, 3, 4, 5	1.10
	1, 2 vs 3, 4, 5	0.49
	1, 2, 3 vs 4, 5	0.017
	1, 2, 3, 4 vs 5	-0.36
site (ref. category=Yelp)	Google	-0.26
	Facebook	0.78

- Hypotheses verified!