Leadx: Yelp Web Scrape

Summer 2024

Poorva Patel

“Nothing in *data analysis* makes sense except in the light of *context*”

—Dobzhansky Template

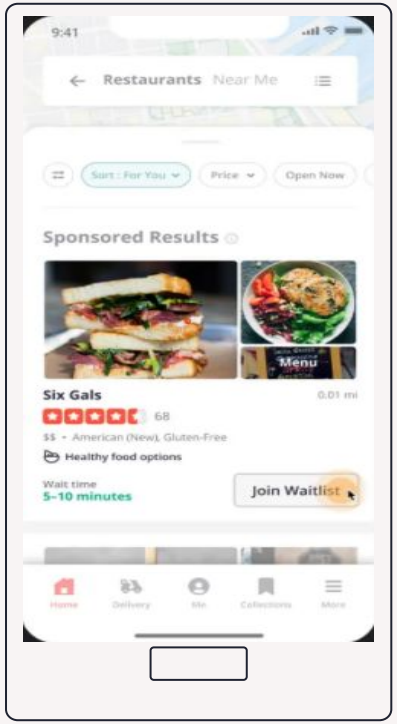
Leadx

Leadx is a consulting company that uses artificial intelligence and machine learning to boost sales for small/medium businesses

… And

Online reviews have become a vital indicator of a business's success

- User-generated content that can be analyzed to gain insights into customer satisfaction, areas needing improvement, and overall business performance.



Yelp

A popular review platform that offers a wealth of information about customer experiences and business operations.

… But

Activity on Yelp

32M
App Unique Devices
(monthly average in 2023)

287M
Cumulative Reviews
(as of Dec. 31, 2023)

Advertising Revenue by Category

Manually analyzing reviews is impractical

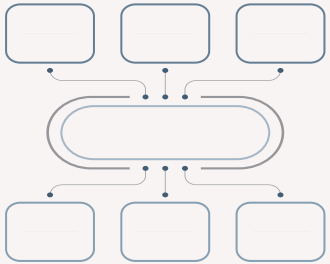
... Therefore

ETL

01 → **Extract**

02 → **Transform**

03 → **Load**



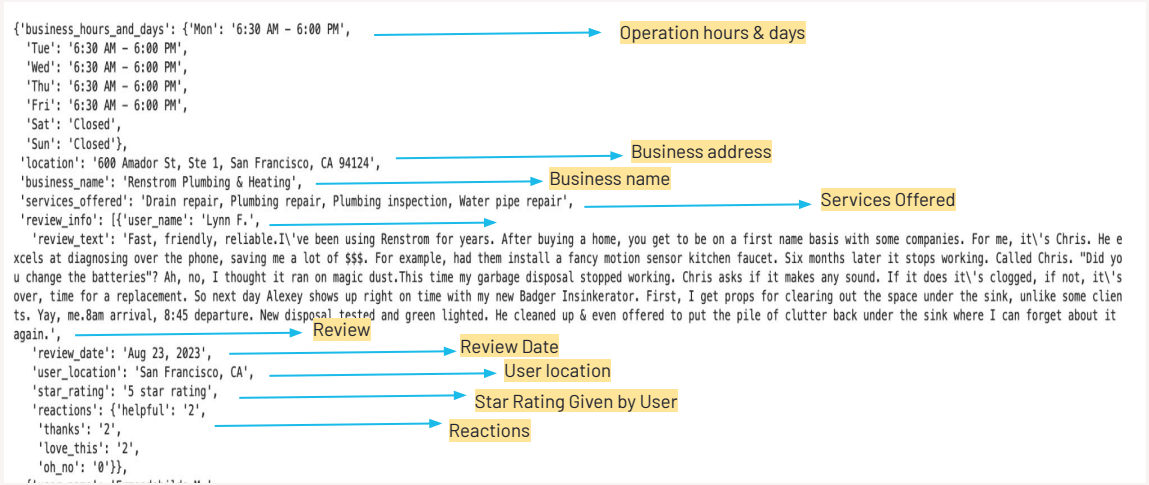
01 Extract

Extracting needed information:

```
# Function to convert business hours and days to a dictionary
def convert_business_hours_to_dict(days: list, times: list) -> dict:
    if len(days) != len(times):
        raise ValueError("The length of days and times lists must be equal")
    opening_hours = dict(zip(days, times))
    return opening_hours

# Function to clean and format address from parsed HTML
def address_clean(parsed_html):
    # search for address tags
    address_tags = parsed_html.find_all('address')

    # search for specific address tag
    address_lines = address_tags[0].find_all(class_='raw__09f24__T4Ezm')

    # join address tag text together
    address_str = "".join([f"{address.text}, " for address in address_lines]).strip()

    # clean address by removing unnecessary characters
    address_str = address_str[:-1] if address_str[-1] == ',' else address_str
    return address_str

# Function to extract services offered from parsed HTML
def extract_services_offered(parsed_html) -> str:
    services_section = parsed_html.find_all('section', {'aria-label': 'Services Offered'})
    if services_section:
        # Find all paragraph tags within the services section
        services_offered = services_section[0].find_all('p', class_='y-css-t1npoe')
        # Join service texts together, separated by commas
        services_text = ", ".join(service.text.strip() for service in services_offered)
        return services_text if services_text else "Services Offered section not available"
    return "Services Offered section not available"

# Function to extract review information from parsed HTML
def extract_review_info(parsed_html):
    # Extract business hours and store operation days
    time_elements = parsed_html.find_all(class_='y-css-29kerx')
    business_hours_list = []
    store_operation_days_list = []

    for i in range(1, 14, 2):  # Loop through from 1 to 13 from the hour table
        if i < len(time_elements):
            business_hours_element = time_elements[i].find(class_='y-css-1iy1dwt')
            business_hours_list.append(business_hours_element.get_text() if business_hours_element else "Business hours not available" + str(i))
            operation_days_element = time_elements[i].find('p')
            store_operation_days_list.append(operation_days_element.get_text() if operation_days_element else "Operation days not available" + str(i))
        else:
            business_hours_list.append("Index out of range: " + str(i))
            store_operation_days_list.append("Index out of range: " + str(i))

    # Combine business hours and operation days into a dictionary
    combined_hours = {day: hour for hour, day in zip(business_hours_list, store_operation_days_list)}

    # Extract the name of the business
    title_elements = parsed_html.find_all(class_='y-css-1j7tr06')
    business_name = title_elements[0].find(class_='y-css-olzveb').text if title_elements and title_elements[0].find(class_='y-css-olzveb') else "Business title not available"

    # Extract location
    location = address_clean(parsed_html) if parsed_html.find('address') else None

    # Extract services offered
    services_offered = extract_services_offered(parsed_html)

    # Extract review elements
    review_elements = parsed_html.find_all('li', class_='y-css-1jp2syp')

    # Initialize dictionary to store extracted information
    reviews = []
    yelp_json = {}
    yelp_json['business_hours_and_days'] = combined_hours
    yelp_json['location'] = location
    yelp_json['business_name'] = business_name
    yelp_json['services_offered'] = services_offered

    # Extract reviews
    for review_element in review_elements:
        review_info = {
            'user_name': review_element.find(class_='y-css-12ly5yx').text if review_element.find(class_='y-css-12ly5yx') else "Name not available",
            'review_text': review_element.find(class_='raw__09f24__T4Ezm').text if review_element.find(class_='raw__09f24__T4Ezm') else "Review text not available",
            'review_date': review_element.find(class_='y-css-19pbem2').text if review_element.find(class_='y-css-19pbem2') else "Date not available",
            'user_location': review_element.find(class_='y-css-12kfwpw').text if review_element.find(class_='y-css-12kfwpw') else "User location not available",
            'star_rating': review_element.find(class_='y-css-9tnml4').get('aria-label') if review_element.find(class_='y-css-9tnml4') else "Rating not available",
            'reactions': {
                reaction.lower().replace(' ', '_'): (review_element.find("span", string=reaction).find_next_sibling("span").text if review_element.find("span", string=reaction) else "0"
                for reaction in ['Helpful', 'Thanks', 'Love this', 'Oh no']
            }
        }
        reviews.append(review_info)

    yelp_json['review_info'] = reviews

    return yelp_json

# Select specific HTML and extract information
yelp_json = extract_review_info(parsed_html8[0])
```

In [48]:
```
# To show all extracted information
yelp_json
```

The extraction process involves using Python to retrieve and decompress HTML content from Yelp. Key data such as business names, addresses, hours, services, and user reviews are parsed and extracted with libraries like aiohttp, BeautifulSoup, and gzip.



02 Transform

Addresses are standardized, business hours are formatted into dictionaries, and user reviews are organized. Sentiment analysis determines sentiment scores and extracts keywords.

```python
# Load spacy model
nlp = spacy.load("en_core_web_sm")

# Initialize sentiment analyzer
nltk.download('vader_lexicon')
sid = SentimentIntensityAnalyzer()

# Parse decompressed HTML with BeautifulSoup
parsed_htmls = yelp_scrape['yelp_scrape_decompressed'].apply(BeautifulSoup, features="html.parser")

# Extract reviews from all parsed HTMLs
all_reviews = []
total_htmls = len(parsed_htmls)       # Get the total number of parsed HTML documents
total_reviews = 0                     # Initialize a counter for the total number of reviews

# Loop through each parsed HTML document
for i, soup in enumerate(parsed_htmls):
    reviews = []
    for review in soup.find_all('li', class_='y-css-1jp2syp'):
        review_text = review.find(class_='raw__09f24__T4Ezm')
        if review_text:
            reviews.append(review_text.text)
    all_reviews.extend(reviews)
    total_reviews += len(reviews)
    print(f"Processed {len(reviews)} reviews from HTML {i+1}/{total_htmls}")
```

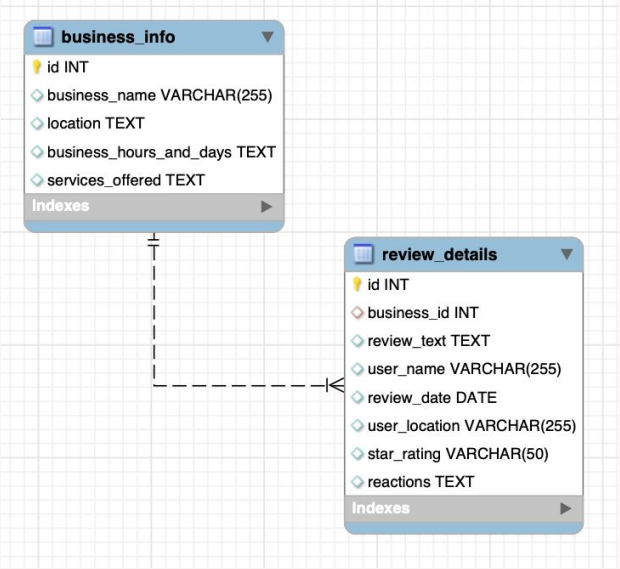
```python
# Print the total number of reviews processed
print(f"Total number of reviews processed: {total_reviews}")
```

```python
# Function to analyze sentiment of reviews
def analyze_sentiment(reviews):
    sentiments = []
    # Loop through each review in the list
    for review in reviews:
        sentiment_scores = sid.polarity_scores(review)
        # Append the review and its sentiment scores to the sentiments list
        sentiments.append({
            'review': review,
            'compound': sentiment_scores['compound'],
            'positive': sentiment_scores['pos'],
            'neutral': sentiment_scores['neu'],
            'negative': sentiment_scores['neg']
        })
    # Convert the sentiments list to a DataFrame
    return pd.DataFrame(sentiments)

# Analyze sentiments of all reviews
sentiment_df = analyze_sentiment(all_reviews)
print(f"Total number of reviews analyzed for sentiment: {len(sentiment_df)}")
print(sentiment_df.head(20))
```

```
Total number of reviews analyzed for sentiment: 50
                                               review  compound  positive  \
0   First psychiatrist I've seen in a LONG time th...    0.9813     0.184
1   Go somewhere else. They say they are accepting...    0.3612     0.169
2   Absolutely terrible. My first visit I loved th...    0.8128     0.105
3   We've been on the fence about changing pediatr...    0.8976     0.045
4   My children have been going there for years, &...   -0.8260     0.115
5   I have to agree with the other reviews on here...    0.9897     0.238
6   Great Doctors. We have been going here since o...    0.9168     0.194
7   As first time mother I was very worried about ...    0.9445     0.142
8   GREAT but be SURE to see the DOCTORS!!Wonderfu...    0.9704     0.113
9   Too many issues with this place to list.  Init...    0.5499     0.115
10  Worst practice ever being going there for more...   -0.9533     0.042
11  We love this dance school. My daughter is 3 ye...    0.9871     0.444
12  A lot of good memories here, my daughter atten...    0.9607     0.264
13  We went here for two years and left.We selecte...    0.9911     0.167
14  My niece was in the three year old baby dance ...   -0.9599     0.014
15  Both my daughters attended Chiampa Dance Cente...    0.9775     0.112
16  Angela started working with me and my husband ...    0.9762     0.157
17  Before working with Angela Ireland Interiors, ...    0.9093     0.153
18  Our office copier developed light blue horizon...   -0.9761     0.000
19  I have been with this insurance broker for man...    0.9112     0.178

    neutral  negative
0     0.816     0.000
1     0.708     0.123
2     0.820     0.075
3     0.928     0.026
4     0.728     0.157
5     0.706     0.056
6     0.727     0.079
7     0.838     0.020
8     0.854     0.034
9     0.821     0.064
10    0.735     0.223
11    0.492     0.064
12    0.715     0.020
13    0.818     0.015
14    0.877     0.109
15    0.879     0.009
16    0.835     0.008
17    0.822     0.025
18    0.849     0.151
19    0.822     0.000
```

03 Load

Example Query: ‘’’’’
SELECT *
FROM business_info
’’’’’



Future Improvements:

- Incorporate other Review Platforms
- Enhance Sentiment Analysis
- Real-Time Data Processing
- ML model for predictive analysis