# Company Reviews Dashboard – Process and Instructions

Niharika Chunduru

2024-09-03

## I. Data Collection

**Companies:** Airbnb, Booking.com, Tripadvisor

**Primary Data Sources:** Glassdoor, Indeed

**Mode of data collection:** JSON data from API Calls[1].

**High-level themes of data:**

- Numerical: Overall and categorical ratings of company from employees
- Textual: Review title, text, pros and cons
- Boolean: Employment status of reviewer, Reviewer's approval of the CEO

The rating scales of the numerical data were aligned in both websites, as both used a five-star scale rating. The issue appeared to be with the review text data, as Indeed contained an aggregated text of the review, while Glassdoor offered separate features of pros and cons in form of text reviews. For data collection, API calls were used to extract review data as they offered best quality as compared to web-scraping. The latter approach would have compromised the quantity and quality of the data set, as these data sources are well equipped will firewalls to prevent web crawlers.

## II. Data Preprocessing and Normalization

The most recent 400 reviews of each company (200 from each data source) were collected for the scope of this project.

Data cleaning, preprocessing and normalization was performed in Python using data science modules like `pandas`, `nltk`, `matplotlib` and `json`.

Fortunately, the numerical ratings were already standardized in both sources with the use of a five-star scale for the ratings. There was, however, standardization needed for the names of the categorical features before they could be aggregated into a single standard data set. For example, one category was named as `culture_and_values_rating` in Glassdoor response, while the same was named `job_culture_rating` in Indeed response. Parallels were drawn manually between these feature names and then they were corroborated into a single feature for the aggregated data set for the ratings. Similarly, the text reviews were split into `pros` and `cons` in the Glassdoor response, while there was only a single review `text` feature in the Indeed response. So, to maintain consistency for aggregation, the `pros` and `cons` were combined into a single `text` feature for further textual analyses. Apart from these naming incosistencies, there were no other issues with the data sets in terms of duplicacy or missing data, which is mostly attributed to the usage of API calls for data collection, instead of scraping.

[1] 1 https://wextractor.com

# III. Interactive Dashboard Development

## Dashboard Platform

For the development of the interactive dashboard, *Shiny* was chosen as the programming tool of choice. *Shiny* makes dashboard development easier than Python because it's part of the R ecosystem, which is built for data analysis and visualization. *Shiny's* concise, declarative code style and its built-in, interactive UI components streamline the process. It also uses reactive programming to automatically update the dashboard based on user inputs or data changes, reducing manual coding. In contrast, Python often requires managing multiple libraries and more complex code to achieve similar results. Dashboards developed using *Shiny* can also be deployed to *ShinyApps*[2], which further cuts down on deployment time and search for domains.

## Dashboard Layout and Usage

Given the objective of the project, an A vs. B layout was chosen for this minimal dashboard to make it easy to do comparative analysis.

The dashboard is split into two 'sides', one for *Company-A*, and the other for its competitor, *Company-B*, both appearing based on the user selection from the multi-select drop down menu located on the top of the dashboard.

Each side begins with a qualitative and quantitative summary of the nature of reviews. While the number of ratings and the reviews are obtained in a straightforward manner, the percentage of positive and negative reviews was calculated by performing sentiment analysis using text polarity scores on every single review.

Below these statistical summaries, a plot of the company ratings is displayed, which contains the normalized categorical ratings of the company in addition to the overall rating provided to the company by the reviewers. In this plot, the origin is shifted to `2.5`, leading to a diverging bar plot of 'low' (`rating<2.5`), and 'high' (`rating>=2.5`) ratings for easier comparison.

The last section is based out of clustering analysis on text-based features of the reviews data sets. It starts off with a short summary of the key themes mentioned by the employees of the company in their reviews. This is followed by a visual depiction of the frequencies in which said themes occurred in the review titles and the text, in the form of word clouds that display the top 15-20 theme keywords discussed in reviews.

# IV. Data Updation

## Scenario-1: Adding more review records to existing companies

It is fairly simple to update the dashboard if more reviews data needs to be added to the companies. All that needs to be done is to run the Juptyter notebooks in this order, and they will automatically update the data sets, which the dashboard code will take in for its updated analysis.

1. **`data_preprocess.ipynb`**: Update the `SOURCE_N_FILES` global parameter to reflect how many json files need to be aggregated for the reviews.
2. **`text_preprocess.ipynb`**: Run as is. The word clouds will get re-generated with the updated reviews data and automatically stored in the desired location for dashboard input.
3. **`sentiment_analysis.ipynb`**: Run as is, once for every company whose data was updated. Update the `company` variable to point to the correct company name.

[2] 2 https://www.shinyapps.io

## Scenario-2: Adding more competitor companies

In this scenario, there will be a little more changes to be performed in the Jupyter notebooks, though the process still remains decently simple.

1. Add the json files for the new company in the `data` folder by following the existing directory structure (`data/<data_source>/<company_name>/<data_source_company_name_[0-n].json>`).
2. **`data_preprocess.ipynb`**: Update the `COMPANIES`, and SOURCE_N_FILES‘ global parameters to reflect the company name, and how many json files need to be aggregated for the reviews.
3. **`text_preprocess.ipynb`**: Update the `COMPANIES` parameter to reflect the updated list of companies, and then run the code as is. If all the files were generated correctly in the previous step, this step will run smoothly too.
4. **`sentiment_analysis.ipynb`**: Update the `company` variable to point to the correct company name, and run the code as is.

## Scenario-3: Adding more data scources companies

While not impossible, this scenario will demand the most amount or work, as the data from the new source will need to be preprocessed and normalized a format that is consistent with the existing data.

1. Add the json files for the new company in the `data` folder by following the existing directory structure (`data/<data_source>/<company_name>/<data_source_company_name_[0-n].json>`).
2. **`data_preprocess.ipynb`**: Update the global parameters and add in new preprocessing and normalizing code to generate the ratings data set.
3. **`text_preprocess.ipynb`**: Update the global parameters and add in new preprocessing and normalizing code to generate the review text word clouds.
4. **`sentiment_analysis.ipynb`**: Update the `company` variable and make sure the data set has `title` and `text` columns to perform sentiment analysis on.