# NLP Part 3 | Exploratory Data Analysis of Text Data

Let's glean some insights from our employee reviews

Kamil Mysiak Jul 26, 2019 · 14 min read ★


Photo by Luke Chesser on Unsplash

This is a continuation of a three part series on NLP using python. Feel free to check out my other articles. (Part 1, Part 2)

Let's get a better understanding of our newly cleansed dataset.
Exploratory Data Analysis (EDA) is the process by which the data analyst becomes acquainted with their data to drive intuition and begin to formulate testable hypotheses. This process typically makes use of descriptive statistics and visualizations.

Let's begin, as always, by importing the necessary libraries and opening our dataset.

```
import pandas as pd
import numpy as np
import nltk
import pickle
import pyLDAvis.sklearn
from collections import Counter
from textblob import TextBlob
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation, NMF
from wordcloud import WordCloud, ImageColorGenerator
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
pd.options.mode.chained_assignment = None
pd.set_option('display.max_colwidth', 100)

with open('indeed_scrape_clean.pkl', 'rb') as pickle_file:
    df = pickle.load(pickle_file)
```

| | rating | rating_description | no_contract | rating_description_str | langs | tokenized | lower | no_punc | stopwords_removed | pos_tags | wordnet_po |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4.0 | Contracted to design custom pitot test adapters. Involved multiple design phases and prototypes.... | [Contracted, to, design, custom, pitot, test, adapters., Involved, multiple, design, phases, and... | Contracted to design custom pitot test adapters. Involved multiple design phases and prototypes.... | en | [Contracted, to, design, custom, pitot, test, adapters, ., Involved, multiple, design, phases, a... | [contracted, to, design, custom, pitot, test, adapters, ., involved, multiple, design, phases, a... | [contracted, to, design, custom, pitot, test, adapters, involved, multiple, design, phases, and.... | [contracted, design, custom, pitot, test, adapters, involved, multiple, design, phases, prototyp... | [(contracted, VBN), (design, NN), (custom, NN), (pitot, JJ), (test, NN), (adapters, NNS), (invol... | [(contracte v), (design, (custom, (pitot, (test, (adapters, (involve v), |
| 1 | 5.0 | Lots of support and collaboration across many engaging projects. You are given an opportunity to... | [Lots, of, support, and, collaboration, across, many, engaging, projects., You, are, given, an, ... | Lots of support and collaboration across many engaging projects. You are given an opportunity to... | en | [Lots, of, support, and, collaboration, across, many, engaging, projects, ., You, are, given, an... | [lots, of, support, and, collaboration, across, many, engaging, projects, ., you, are, given, an... | [lots, of, support, and, collaboration, across, many, engaging, projects, you, are, given, an, o... | [lots, support, collaboration, across, many, engaging, projects, given, opportunity, grow, ideas... | [(lots, NNS), (support, NN), (collaboration, NN), (across, IN), (many, JJ), (engaging, VBG), (pr... | [(lots, (support, (collaboratio n), (across, (many, (engaging, (projects, |

If you recall from our previous tutorial, we went through a series of pre-processing steps to clean and prepare our data for analysis. Our final dataset contains numerous columns but the last column “lemmatized”, contained our final cleansed list of words. We are going to overwrite our existing dataframe because we are only interested in the “rating” and “lemmatized” columns.

```
df = df[['rating', 'lemmatized']]
df.head()
```

| | rating | lemmatized |
|---|---|---|
| 0 | 4.0 | [contract, design, custom, pitot, test, adapter, involve, multiple, design, phase, prototypes, f... |
| 1 | 5.0 | [lot, support, collaboration, across, many, engage, project, give, opportunity, grow, idea, resp... |
| 2 | 5.0 | [work, responsibility, culture, great, hard, part, job, hectic, management, good, google, ad, le... |
| 4 | 5.0 | [amaze, work, environment, everyone, smart, friendly, learn, lot, great, coworkers, amaze, manag... |
| 5 | 5.0 | [productive, innovative, culture, environment, foster, creativity, limit, potential, positive, t... |

## Sentiment Analysis

Sentiment analysis is the process of determining the writer’s attitude or opinion ranging from -1 (negative attitude) to 1 (positive attitude). We’ll be using the TextBlob library to analyze sentiment. TextBlob’s Sentiment() function requires a string but our “lemmatized” column is currently a list. Let’s convert the list into a string.

```
df['lemma_str'] = [' '.join(map(str,l)) for l in df['lemmatized']]
df.head()
```

| | rating | lemmatized | lemma_str |
|---|---|---|---|
| 0 | 4.0 | [contract, design, custom, pitot, test, adapter, involve, multiple, design, phase, prototypes, f... | contract design custom pitot test adapter involve multiple design phase prototypes final design ... |
| 1 | 5.0 | [lot, support, collaboration, across, many, engage, project, give, opportunity, grow, idea, resp... | lot support collaboration across many engage project give opportunity grow idea respect addition... |
| 2 | 5.0 | [work, responsibility, culture, great, hard, part, job, hectic, management, good, google, ad, le... | work responsibility culture great hard part job hectic management good google ad learn environme... |
| 4 | 5.0 | [amaze, work, environment, everyone, smart, friendly, learn, lot, great, coworkers, amaze, manag... | amaze work environment everyone smart friendly learn lot great coworkers amaze manager director ... |
| 5 | 5.0 | [productive, innovative, culture, environment, foster, creativity, limit, potential, positive, t... | productive innovative culture environment foster creativity limit potential positive topic wish ... |

Now we can pass the “lemma_str” column into the Sentiment() function to calculate sentiment. Since we have the “rating” column, we can validate how well the sentiment

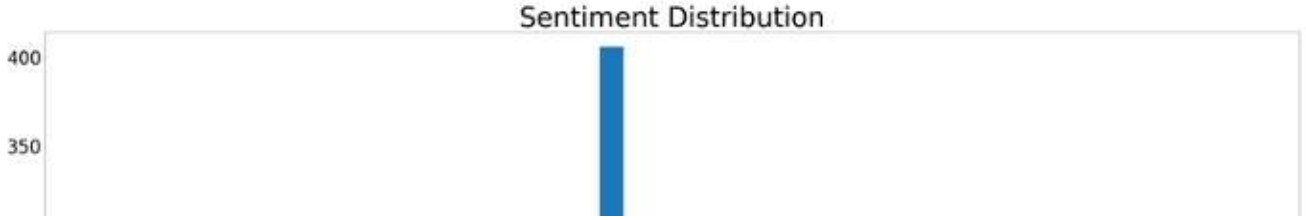
analysis was able to determine the writer's attitude. That said, we do see obvious errors as rating #5 has a rating of 5but a fairly low sentiment.

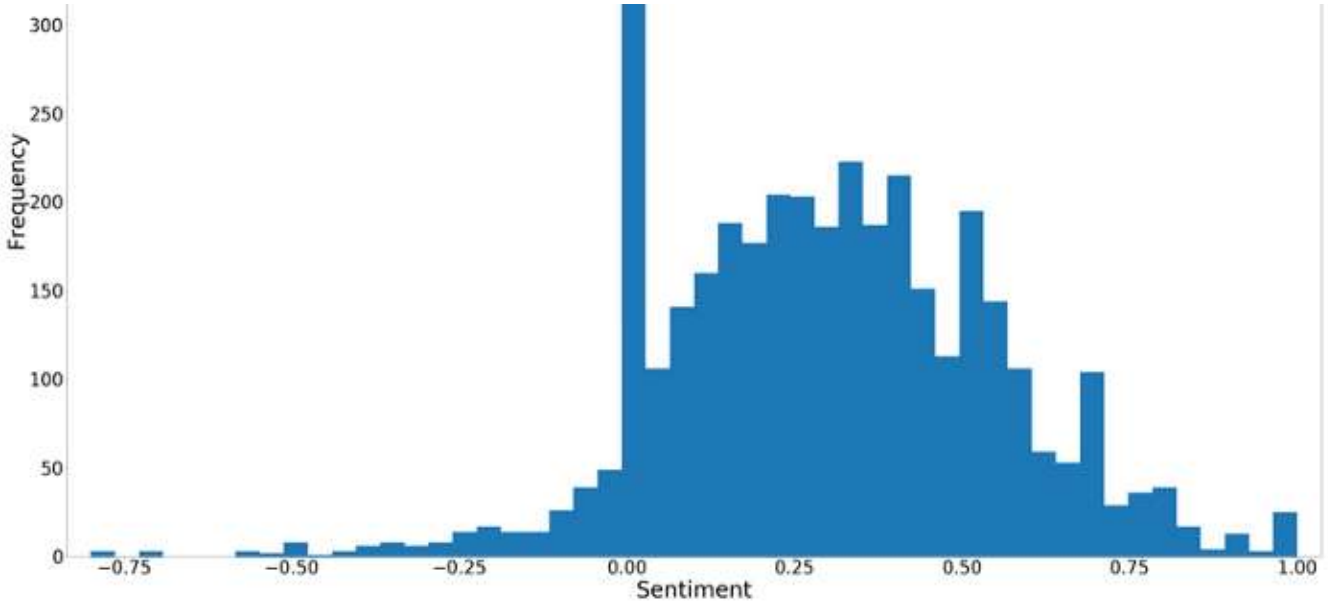
```
df['sentiment'] = df['lemma_str'].apply(lambda x:
TextBlob(x).sentiment.polarity)
df.head()
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 5 | 5.0 | [productive, innovative, culture, environment, foster, creativity, limit, potential, positive, t... | productive innovative culture environment foster creativity limit potential positive topic wish ... | 0.181818 | 144 | 18 |
| 6 | 5.0 | [technically, strong, people, google, client, directly, work, google.., portal, application, sup... | technically strong people google client directly work google.. portal application support insura... | 0.266667 | 113 | 14 |
| 7 | 5.0 | [contract, google, 7, month, love, every, minute, people, great, loved, manager, work, perk, eve... | contract google 7 month love every minute people great loved manager work perk ever see boulder ... | 0.666667 | 107 | 18 |
| 8 | 5.0 | [great, experience, great, perspective, google, fiber, optic, great, place, get, cash, check, ne... | great experience great perspective google fiber optic great place get cash check next time get b... | 0.514286 | 138 | 21 |
| 9 | 4.0 | [really, enjoyed, work, great, environment, good, food, free, lunch, make, good, money.., good, ... | really enjoyed work great environment good food free lunch make good money.. good people work al... | 0.614286 | 103 | 17 |

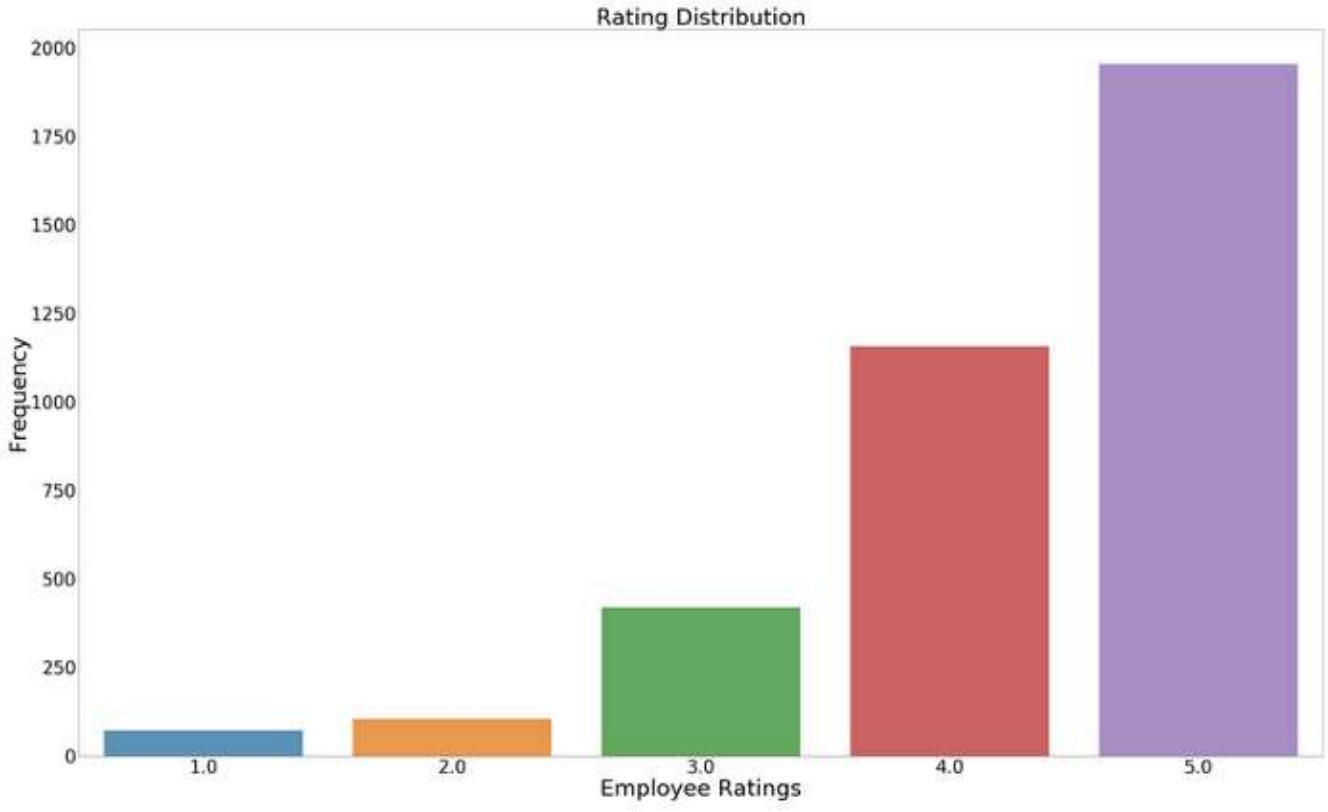
When comparing a histogram of our sentiment, we can see that the vast majority of our derived sentiment rating is overwhelmingly positive. When we compare this against the "ratings" column, we can see a similar pattern emerge. Not only do we feel comfortable in the accuracy of the sentiment analysis but we can see that the overall employee attitude about the company is very positive. It is no wonder Google regularly makes the Forbes's Best Places to Work list.

```
plt.figure(figsize=(50,30))
plt.margins(0.02)
plt.xlabel('Sentiment', fontsize=50)
plt.xticks(fontsize=40)
plt.ylabel('Frequency', fontsize=50)
plt.yticks(fontsize=40)
plt.hist(df['sentiment'], bins=50)
plt.title('Sentiment Distribution', fontsize=60)
plt.show()
```

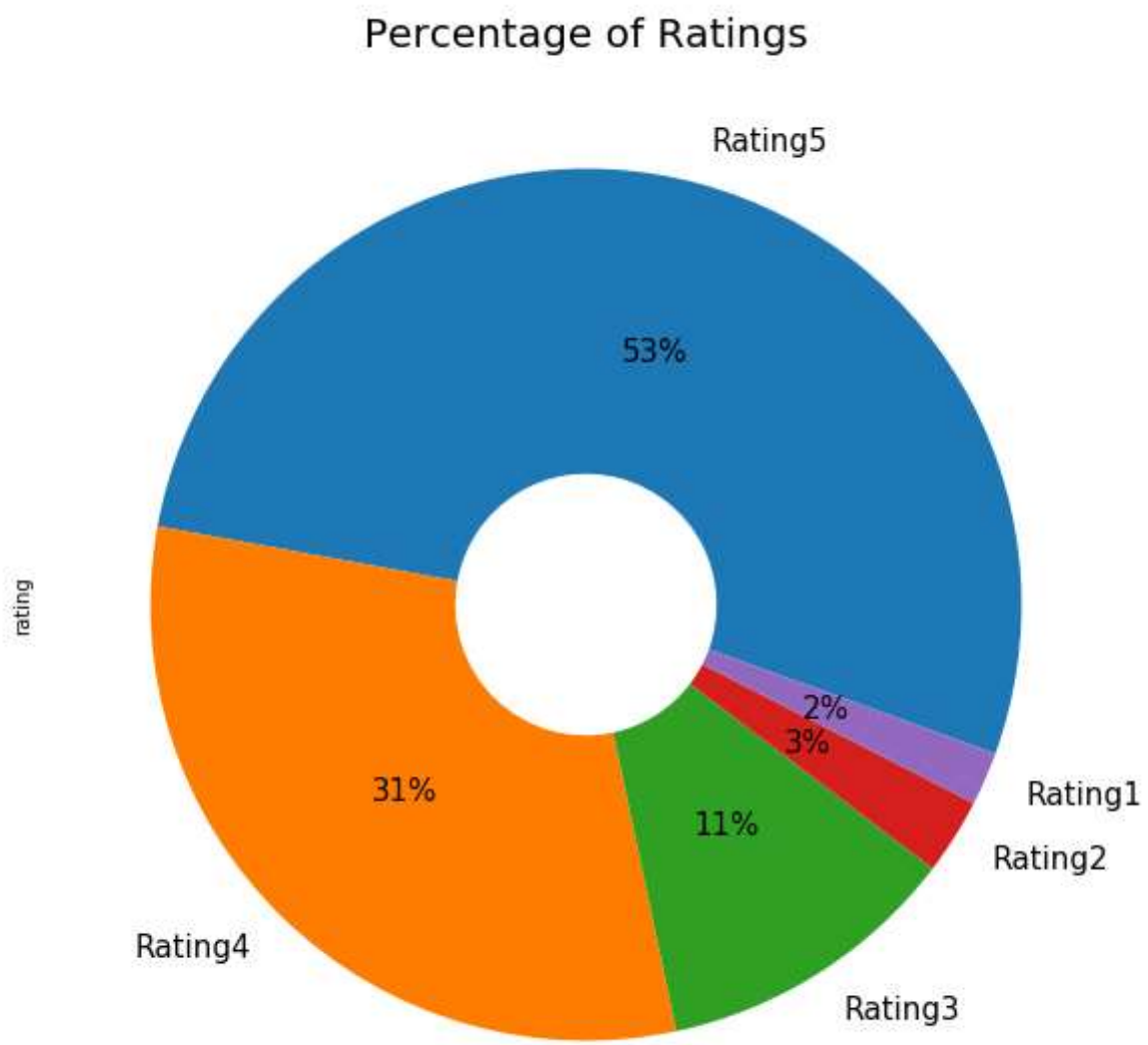




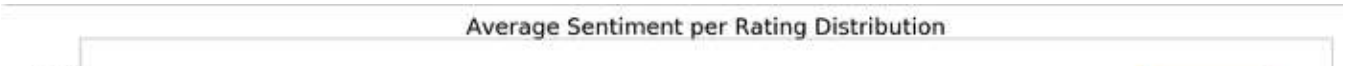

```
x_rating = df.rating.value_counts()
y_rating = x_rating.sort_index()
plt.figure(figsize=(50,30))
sns.barplot(x_rating.index, x_rating.values, alpha=0.8)
plt.title("Rating Distribution", fontsize=50)
plt.ylabel('Frequency', fontsize=50)
plt.yticks(fontsize=40)
plt.xlabel('Employee Ratings', fontsize=50)
plt.xticks(fontsize=40)
```

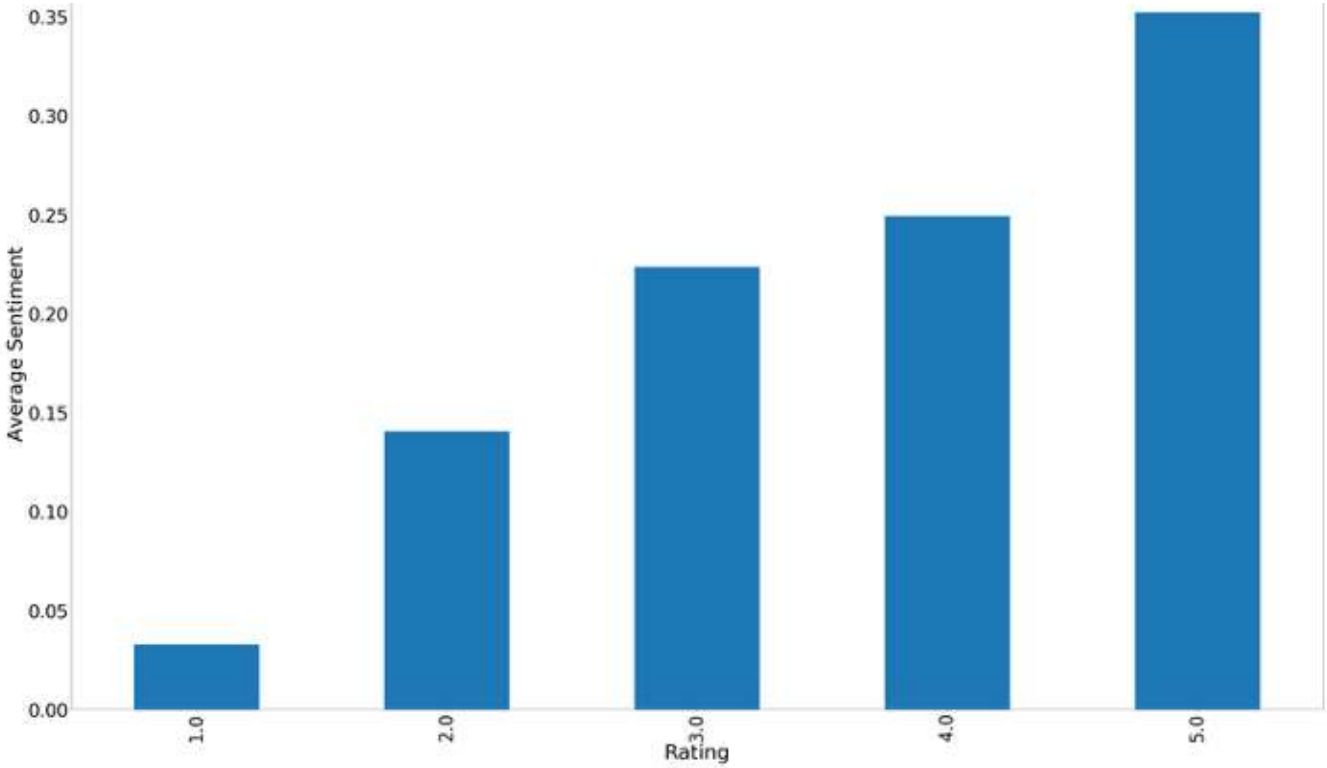


```
plt.figure(figsize=(30,10))
plt.title('Percentage of Ratings', fontsize=20)
df.rating.value_counts().plot(kind='pie', labels=['Rating5',
'Rating4', 'Rating3', 'Rating2', 'Rating1'],
                              wedgeprops=dict(width=.7),
autopct='%1.0f%%', startangle= -20,
                              textprops={'fontsize': 15})
```




```
polarity_avg = df.groupby('rating')
['sentiment'].mean().plot(kind='bar', figsize=(50,30))
plt.xlabel('Rating', fontsize=45)
plt.ylabel('Average Sentiment', fontsize=45)
plt.xticks(fontsize=40)
plt.yticks(fontsize=40)
plt.title('Average Sentiment per Rating Distribution', fontsize=50)
plt.show()
```






Let's create two additional features of "word_count" to determine the number of words per review and "review_len" to determine the number of letters per review.

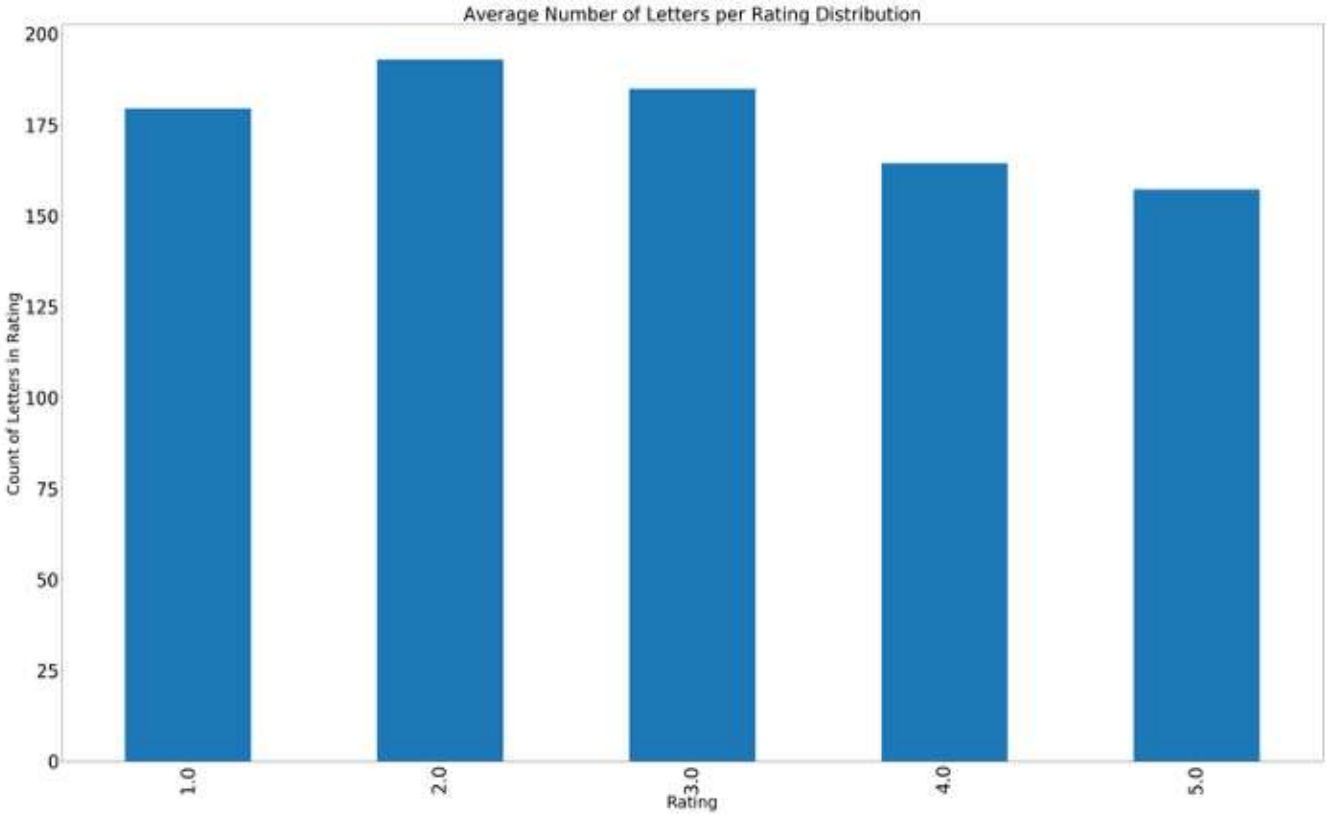
```
df['word_count'] = df['lemmatized'].apply(lambda x:
len(str(x).split()))

df['review_len'] = df['lemma_str'].astype(str).apply(len)
```

| | rating | lemmatized | lemma_str | sentiment | review_len | word_count |
|---|---|---|---|---|---|---|
| 0 | 4.0 | [contract, design, custom, pitot, test, adapter, involve, multiple, design, phase, prototypes, f... | contract design custom pitot test adapter involve multiple design phase prototypes final design ... | 0.000000 | 137 | 18 |
| 1 | 5.0 | [lot, support, collaboration, across, many, engage, project, give, opportunity, grow, idea, resp... | lot support collaboration across many engage project give opportunity grow idea respect addition... | 0.675000 | 113 | 15 |
| 2 | 5.0 | [work, responsibility, culture, great, hard, part, job, hectic, management, good, google, ad, le... | work responsibility culture great hard part job hectic management good google ad learn environme... | 0.260417 | 115 | 17 |
| 4 | 5.0 | [amaze, work, environment, everyone, smart, friendly, learn, lot, great, coworkers, amaze, manag... | amaze work environment everyone smart friendly learn lot great coworkers amaze manager director ... | 0.284184 | 192 | 27 |
| 5 | 5.0 | [productive, innovative, culture, environment, foster, creativity, limit, potential, positive, t... | productive innovative culture environment foster creativity limit potential positive topic wish ... | 0.181818 | 144 | 18 |

Although the differences are not significantly large it seems the longest reviews based on the count of letters and words seem to be negative and neutral. It seems disgruntled employees typically provide significantly more detail in their reviews. This result is not

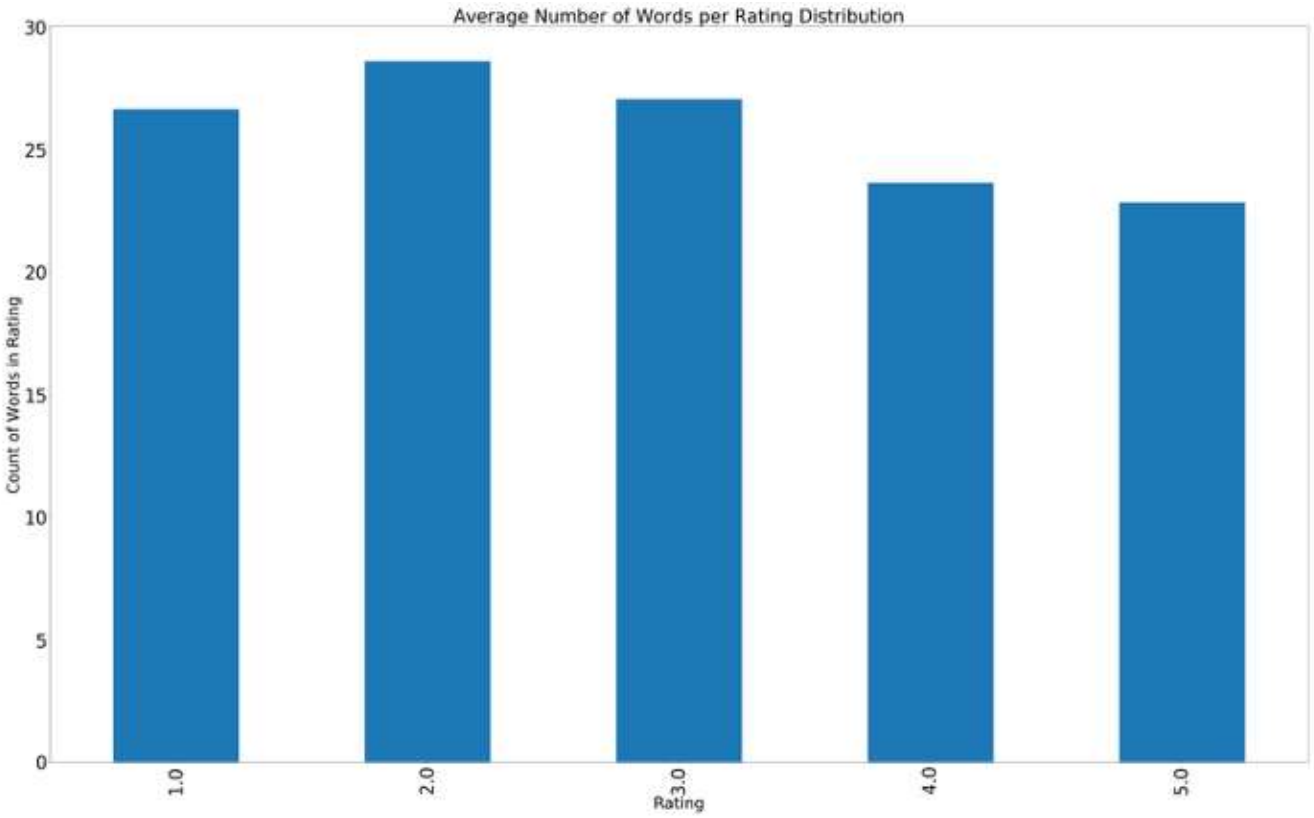
uncommon as humans have a tendency to complain in detail but praise in brief. This can be further confirmed by examining the correlation matrix below. Both ratings and sentiment have a negative correlation with “review_len” and “word_count”. This would explain the inverse relationship as the count of letters and words per review increases the overall rating and sentiment decreases. However, once again the correlation is rather small nevertheless negative.

```
letter_avg = df.groupby('rating')
['review_len'].mean().plot(kind='bar', figsize=(50,30))
plt.xlabel('Rating', fontsize=35)
plt.ylabel('Count of Letters in Rating', fontsize=35)
plt.xticks(fontsize=40)
plt.yticks(fontsize=40)
plt.title('Average Number of Letters per Rating Distribution',
fontsize=40)
plt.show()
```

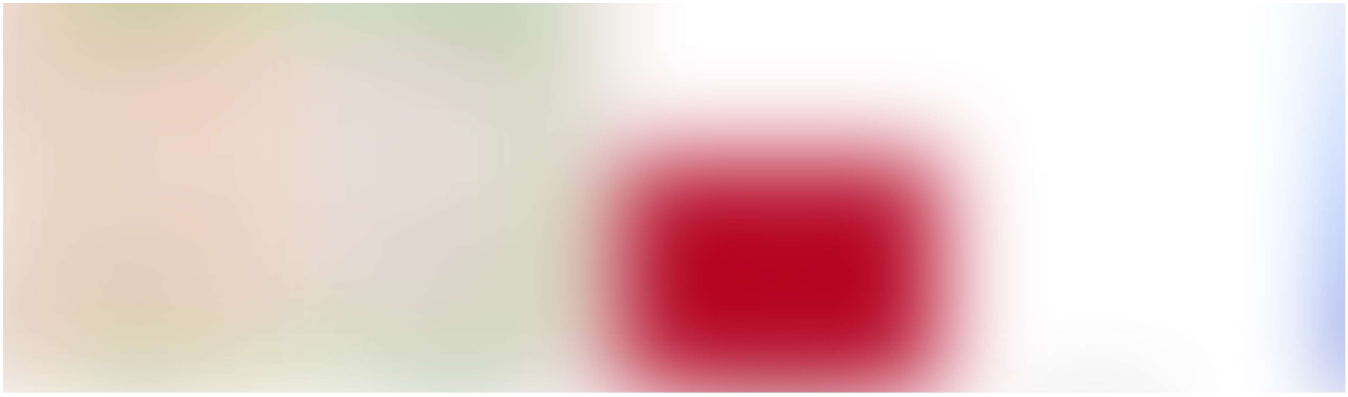



```
word_avg = df.groupby('rating')
['word_count'].mean().plot(kind='bar', figsize=(50,30))
plt.xlabel('Rating', fontsize=35)
plt.ylabel('Count of Words in Rating', fontsize=35)
plt.xticks(fontsize=40)
plt.yticks(fontsize=40)
plt.title('Average Number of Words per Rating Distribution',
```

```
fontsize=40)
plt.show()
```




```
correlation = df[['rating','sentiment', 'review_len',
'word_count']].corr()
mask = np.zeros_like(correlation, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
plt.figure(figsize=(50,30))
plt.xticks(fontsize=40)
plt.yticks(fontsize=40)
sns.heatmap(correlation, cmap='coolwarm', annot=True, annot_kws=
{"size": 40}, linewidths=10, vmin=-1.5, mask=mask)
```



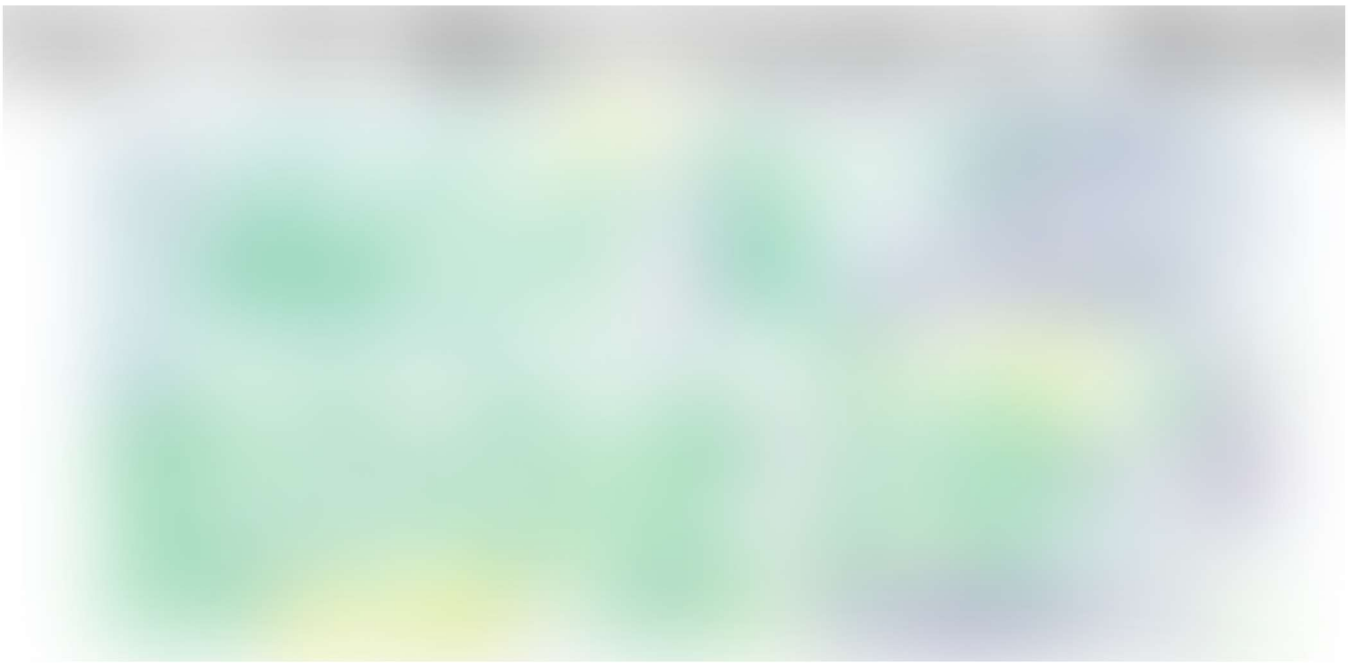
## Term Frequency Analysis

Let's take an in-depth look at the actual reviews themselves. What are the most common words? What are the most common words by rating? Answers to these questions will provide further insights into the opinions of Google's employees.

NLTK has a great library named "FreqDist" which allows us to determine the count of the most common terms in our corpus. First, we need to convert our individual lists of tokenized reviews into a comprehensive list of iterable tokens which stores all the reviews together. Finally, we pass FreqDist() the "allwords" object and apply the "most_common(100)" function to obtain the 100 most common words.

```
words = df['lemmatized']
allwords = []
for wordlist in words:
    allwords += wordlist

print(allwords)
```

```
mostcommon = FreqDist(allwords).most_common(100)

wordcloud = WordCloud(width=1600, height=800,
background_color='white').generate(str(mostcommon))
fig = plt.figure(figsize=(30,10), facecolor='white')
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title('Top 100 Most Common Words', fontsize=100)

plt.tight_layout(pad=0)
plt.show()
```



```
mostcommon_small = FreqDist(allwords).most_common(25)
x, y = zip(*mostcommon_small)

plt.figure(figsize=(50,30))
plt.margins(0.02)
plt.bar(x, y)
plt.xlabel('Words', fontsize=50)
plt.ylabel('Frequency of Words', fontsize=50)
plt.yticks(fontsize=40)
plt.xticks(rotation=60, fontsize=40)
plt.title('Frequency of 25 Most Common Words', fontsize=60)
plt.show()
```

The results of the term frequency analysis certainly supports the overall positive sentiment of the reviews. Terms such as “Great”, “Work”, “People”, “Team”, “Company” point to a positive company environment where employees enjoy working together.

> *Based on the fact terms such as "work", "Google", "job" and "company" have such a high frequency in the corpus it might be a good idea to remove them (ie. add them to our stopwords) prior to analysis.*

That said, a company can always improve therefore, let's examine the most common words for each review rating.

It seems the most common words for reviews where the rating = 1 had something to do with the "Management", "Manager", "People". We have to be careful when interpreting these results as there are only (2%) reviews with a rating of 1 based on our pie chart printed above.

```
group_by = df.groupby('rating')['lemma_str'].apply(lambda x:
Counter(' '.join(x).split()).most_common(25))

group_by_0 = group_by.iloc[0]
words0 = list(zip(*group_by_0))[0]
freq0 = list(zip(*group_by_0))[1]

plt.figure(figsize=(50,30))
plt.bar(words0, freq0)
plt.xlabel('Words', fontsize=50)
plt.ylabel('Frequency of Words', fontsize=50)
```

```
plt.yticks(fontsize=40)
plt.xticks(rotation=60, fontsize=40)
plt.title('Frequency of 25 Most Common Words for Rating=1',
fontsize=60)
plt.show()
```

Reviews with a rating of 2 had a common theme of “manager”, “management”. Once again the rating distribution is very skewed but this does give us some clues on ways to improve the organization.

```
group_by_1 = group_by.iloc[1]
words1 = list(zip(*group_by_1))[0]
freq1 = list(zip(*group_by_1))[1]
plt.figure(figsize=(50,30))
plt.bar(words1, freq1)
plt.xlabel('Words', fontsize=50)
plt.ylabel('Frequency of Words', fontsize=50)
plt.yticks(fontsize=40)
plt.xticks(rotation=60, fontsize=40)
plt.title('Frequency of 25 Most Common Words for Rating=2',
fontsize=60)
plt.show()
```

It is difficult to derive accurate insights from a “neutral” rating as the employee didn’t have anything overly positive or negative to say about the company. That said, it is interesting that “Management” has once again crept into the top 10 words. So far roughly 14% of the employees had a negative or neutral (didn’t have anything good or bad to say) about the management at Google. Words like “work” and “Google” seem to be skewing the distribution for all ratings, it would be a good idea to remove these words from future analysis.

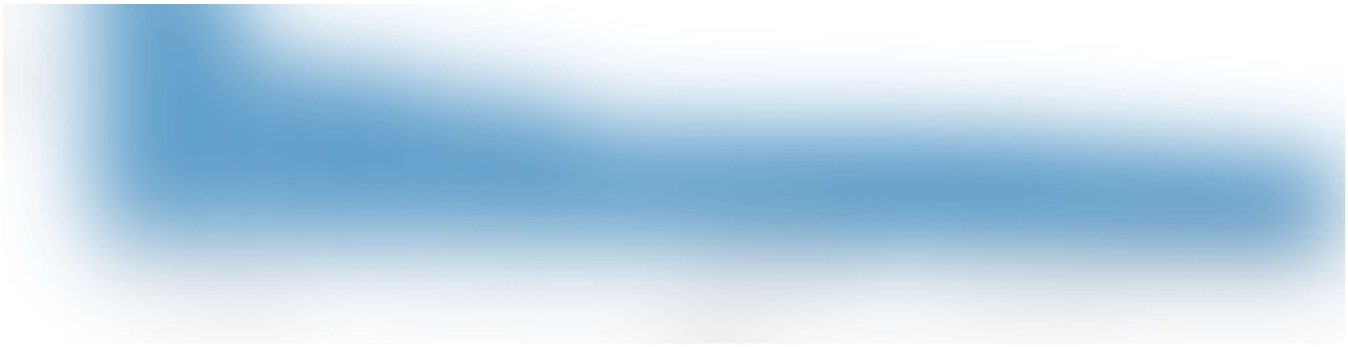
Notice the “…”, we have some more data processing to perform. 😔

```
group_by_2 = group_by.iloc[2]
words2 = list(zip(*group_by_2))[0]
freq2 = list(zip(*group_by_2))[1]
plt.figure(figsize=(50,30))
plt.bar(words2, freq2)
plt.xlabel('Words', fontsize=50)
plt.ylabel('Frequency of Words', fontsize=50)
plt.yticks(fontsize=40)
plt.xticks(rotation=60, fontsize=40)
plt.title('Frequency of 25 Most Common Words for Rating=3',
fontsize=60)
plt.show()
```

Rating of 4 and 5 had very similar terms as it seems employees enjoy their work, the people with whom they work, and value the environment/culture at Google. For example, “design”, “learning opportunities”, “people”, “time” and “team” all made an appearance. That said, we don’t see “Management” or “Manager” anywhere in the top 10 words. This is very insightful as it helps to validate the results from ratings 1, 2, and 3. Last but not least, these word frequencies (ie. ratings 4 & 5) have been derived from a very large number of reviews which only adds to the validity of these results; management is certainly an area of improvement.

```
group_by_3 = group_by.iloc[3]
words3 = list(zip(*group_by_3))[0]
freq3 = list(zip(*group_by_3))[1]
plt.figure(figsize=(50,30))
plt.bar(words3, freq3)
plt.xlabel('Words', fontsize=50)
plt.ylabel('Frequency of Words', fontsize=50)
plt.yticks(fontsize=40)
plt.xticks(rotation=60, fontsize=40)
plt.title('Frequency of 25 Most Common Words for Rating=4',
fontsize=60)
plt.show()
```

```
group_by_4 = group_by.iloc[4]
words4 = list(zip(*group_by_4))[0]
freq4 = list(zip(*group_by_4))[1]
plt.figure(figsize=(50,30))
plt.bar(words4, freq4)
plt.xlabel('Words', fontsize=50)
plt.ylabel('Frequency of Words', fontsize=50)
plt.yticks(fontsize=40)
plt.xticks(rotation=60, fontsize=40)
plt.title('Frequency of 25 Most Common Words for Rating=5',
fontsize=60)
plt.show()
```



## Topic Modeling

Finally, let's apply a few topic modeling algorithms to help derive specific topics or themes for our reviews. Before we have determined the topics for each rating we have to perform one additional processing step. Right now our data/words are still readable to us human beings whereas computers only understand numbers. We need to convert our text into numbers or vectors.

### CountVectorizer

The CountVectorizer method of vectorizing tokens transposes all the words/tokens into features and then provides a count of occurrence of each word. The result is called a document term matrix, which you can see below.

First, we create the vectorizer object. Max_df=0.9 will remove words that appear in more than 90% of the reviews. Min_df=25 will remove words that appear in less than 25 reviews. Next, we create the spare matrix as the result of fit_transform(). Finally, we create a list of all the words/features. The result is our document term matrix. Each row represents individual employee reviews and counts of how many times each word/feature occurs in each review.

```
tf_vectorizer = CountVectorizer(max_df=0.9, min_df=25,
max_features=5000)

tf = tf_vectorizer.fit_transform(df['lemma_str'].values.astype('U'))
tf_feature_names = tf_vectorizer.get_feature_names()

doc_term_matrix = pd.DataFrame(tf.toarray(),
columns=list(tf_feature_names))
doc_term_matrix
```

## Latent Dirichlet Allocation (LDA) Topic Modeling

Now that we have prepared our data for topic modeling, we'll be using the Latent Dirichlet Allocation (LDA) approach to determine the topics present in our corpus. In our model, we are going to produce 10 individual topics (ie. n_components). Once the model is created let's create a function to display the identified topics. Each topic will consist of 10 words. The function will have three required parameters; the LDA model, feature names from the document term matrix, and the number of words per topic.

```
lda_model = LatentDirichletAllocation(n_components=10,
learning_method='online', max_iter=500, random_state=0).fit(tf)

no_top_words = 10

def display_topics(model, feature_names, no_top_words):
    for topic_idx, topic in enumerate(model.components_):
        print("Topic %d:" % (topic_idx))
        print(" ".join([feature_names[i]
                        for i in topic.argsort()[:-no_top_words -
1:-1]]))

display_topics(lda_model, tf_feature_names, no_top_words)
```

It certainly takes a little imagination to determine the topics produced by the LDA model. Since we know that “work”, “google” and “job” are very common works we can almost ignore them.

Topic 0: Good Design Processes

Topic 1: Great Work Environment

Topic 2: Flexible Work Hours

Topic 3: Skill Building

Topic 4: Difficult but Enjoyable Work

Topic 5: Great Company/Job

Topic 6: Care about Employees

Topic 7: Great Contractor Pay

Topic 8: Customer Service

Topic 9: ?

## pyLDAvis

pyLDAvis is an interactive LDA visualization python library. Each circle represents a unique topic, the size of the circle represents the importance of the topic and finally, the distance between each circle represents how similar the topics are to each other. Selecting a topic/circle will reveal a horizontal bar chart displaying the 30 most relevant words for the topic along with the frequency of each word appearing in the topic and the overall corpus.

The relevance metric helps to distinguish words which are distinct/exclusive to the topic (λλ closer to 0.0) and words which have a high probability of being included in the selected topic (λλ closer to 1.0).

```
pyLDAvis.enable_notebook()
panel = pyLDAvis.sklearn.prepare(lda_model, tf, tf_vectorizer,
mds='tsne')
panel
```

## TF-IDF

LDA isn't the only approach to topic modeling. Let's try another method named the Non-Negative Matrix Factorization (NMF) approach and see if our topics can be slightly more defined. Instead of using the simple CountVectorizer method to vectorize our words/tokens, we'll use the TF-IDF (Term Frequency — Inverse Document Frequency) method. The TF-IDF method helps to bring down the weight/impact of high-frequency words (ie. "work", "Google" and 'job" in our case).

Much like the CountVectorizer method we first create the vectorizer object. Max_df=0.9 will remove words that appear in more than 90% of the reviews. Min_df=25 will remove words that appear in less than 25 reviews. Next, we create the spare matrix as the result of fit_transform(). Finally, we create a list of all the words/features.

```
tfidf_vectorizer = TfidfVectorizer(max_df=0.90, min_df =25,
max_features=5000, use_idf=True)

tfidf = tfidf_vectorizer.fit_transform(df['lemma_str'])
tfidf_feature_names = tfidf_vectorizer.get_feature_names()
```

```
doc_term_matrix_tfidf = pd.DataFrame(tfidf.toarray(),
columns=list(tfidf_feature_names))
doc_term_matrix_tfidf
```



## Non-Negative Matrix Factorization (NMF)

```
nmf = NMF(n_components=10, random_state=0, alpha=.1,
init='nndsvd').fit(tfidf)

display_topics(nmf, tfidf_feature_names, no_top_words)
```



The topics produced via NMF seem to be much more distinct compared to LDA.

Topic 0: Fun Work Culture

Topic 1: Design Process

Topic 2: Enjoyable Job

Topic 3:

Topic 4: Great Experience

Topic 5: Perks

Topic 6: Learning Opportunities

Topic 7: Great Company/Job

Topic 8: Contractor Employee Experience

Topic 9: Management

Let's add both the LDA and NMF topics into our dataframe for further analysis. Let's also remap the integer topics into our subjectively derived topic labels.

```
nmf_topic_values = nmf.transform(tfidf)
df['nmf_topics'] = nmf_topic_values.argmax(axis=1)
lda_topic_values = lda_model.transform(tf)
df['lda_topics'] = lda_topic_values.argmax(axis=1)

lda_remap = {0: 'Good Design Processes', 1: 'Great Work
Environment', 2: 'Flexible Work Hours', 3: 'Skill Building', 4:
'Difficult but Enjoyable Work', 5: 'Great Company/Job', 6: 'Care
about Employees', 7: 'Great Contractor Pay', 8: 'Customer Service',
9: 'Unknown1'}

df['lda_topics'] = df['lda_topics'].map(lda_remap)

nmf_remap = {0: 'Fun Work Culture', 1: 'Design Process', 2:
'Enjoyable Job', 3: 'Difficult but Enjoyable Work',
             4: 'Great Experience', 5: 'Perks', 6: 'Learning
Opportunities', 7: 'Great Company/Job',
             8: 'Contractor Employee Experience', 9: 'Management'}

df['nmf_topics'] = df['nmf_topics'].map(nmf_remap)
```

Examining the frequency of topics produced by NMF we can see that the first 5 topics show up at a relatively similar frequency. Keep in mind these are the topics across all reviews (positive, neutral, and negative) and if you recall our dataset is negatively skewed as the majority of the reviews are positive.

```
nmf_x = df['nmf_topics'].value_counts()
nmf_y = nmf_x.sort_index()
plt.figure(figsize=(50,30))
sns.barplot(nmf_x, nmf_y.index)
plt.title("NMF Topic Distribution", fontsize=50)
plt.ylabel('Review Topics', fontsize=50)
plt.yticks(fontsize=40)
plt.xlabel('Frequency', fontsize=50)
plt.xticks(fontsize=40)
```

Let's split our data and examine the topics for the negative reviews based on ratings of 1 and 2. It is interesting to see that despite the negative ratings the employees still

overwhelmingly enjoyed their work, culture, and the company overall. It is very difficult to obtain an accurate perspective on the topics for negative reviews due to the skewness of our dataset (ie. the relatively small amount of negative reviews).

```
df_low_ratings = df.loc[(df['rating']==1) | (df['rating']==2)]

nmf_low_x = df_low_ratings['nmf_topics'].value_counts()
nmf_low_y = nmf_low_x.sort_index()
plt.figure(figsize=(50,30))
sns.barplot(nmf_low_x, nmf_low_y.index)
plt.title("NMF Topic Distribution for Low Ratings (1 & 2)",
fontsize=50)
plt.ylabel('Frequency', fontsize=50)
plt.yticks(fontsize=40)
plt.xlabel('Review Topics', fontsize=50)
plt.xticks(fontsize=40)
```

Since we have many more positive reviews the topics derived via NMF will be much more accurate. It seems contractor employees make up many of the reviews. Employees find an efficient design process, work which is difficult but enjoyable, and an overall happy sentiment towards Google.

```
df_high_ratings = df.loc[(df['rating']==4) | (df['rating']==5)]

nmf_high_x = df_high_ratings['nmf_topics'].value_counts()
nmf_high_y = nmf_high_x.sort_index()
plt.figure(figsize=(50,30))
```

```
sns.barplot(nmf_high_x, nmf_high_y.index)
plt.title("NMF Topic Distribution for High Ratings (3 & 4)",
fontsize=50)
plt.ylabel('Frequency', fontsize=50)
plt.yticks(fontsize=40)
plt.xlabel('Review Topics', fontsize=50)
plt.xticks(fontsize=40)
```

## Conclusion

Based on the results obtained it seems Google's employees are overwhelmingly happy working at Google. We see a negatively skewed distribution where 84% of employees had given Google a rating of 4 or 5 (out of a 1–5 Likert scale). A sentiment analysis confirmed these results even when employees who gave Google a rating between 2 and 3 had a positive average sentiment score.

As we dug a bit deeper into the data an interesting discovery was made which would need to be validated with additional data. When we observed the term/word frequencies per rating it seemed that terms around "*managers/management*" seemed to present themselves for ratings 1, 2, and 3. As mentioned earlier, the data was skewed as the majority of ratings were positive but it was interesting to see that employees who had a negative or a neutral rating seemed to mention *"management"* often.

On the other hand, employees who rated Google between 4 and 5 seemed to be using words such as *"great", "work", "job", "design", "company", "good", "culture", "people"*, etc. These results were slightly confirmed with an examination of topics/themes in our

corpus. An NMF analysis of topics determined that employees who rated Google with a 4 or 5 were eager to discuss the difficult but enjoyable work, great culture, design process. It was interesting to also see the absence of any mention of terms like “manager” and “management” which is also telling and helps to validate our previous insight.

Google continues to be a preferred employer of choice for many, as 84% of reviews were positive. That said, we identified a potential area of improvement which stemmed from Google’s managers and/or management techniques.

Can you think of any other EDA methods and/or strategies we could have explored? Posted them in the comments below. Also, if you have any constructive feedback or see a mistake I have made please call me out 😃

## Thanks!



NLP Employer Reviews Beautifulsoup Natural Language Processi Exploratory Data Analysis