A3_section3_Friday_Yusuf

April 25, 2021

1.0 Data pre-processing

The pre-processing technique used for this analysis include; part of speech tagging, bigram distribution, removing non-ASCII, converting all characters to lower case, removing punctuation marks and stop words, and tokenization. These are some of the common approaches used for NLP analysis.

1.1 Importing all the libraries and data

All the required libraries were imported the python Jyputer notebook. Panda was used to read the scrapped data on indeed_data. To have a proper view of the data, the first ten observations were read, as illustrated in figure 1. Figure 1.0 shows that there are many NaN on the rating con and pros, there are also unnamed variables which are duplicates of the index and these were dropped.

```
# loading the data
data = pd.read_csv('indeed_data.csv')
data.head(10)
```

	Unnamed: 0	rating	rating_title	rating_description	rating_pros	rating_cons
0	0	3.0	Good culture and peers	Very formal client oriented service consulting...	NaN	NaN
1	1	5.0	Productive and competitive workplace with many...	During a typical day at work we attend any cus...	innovation, leading skills, be part of Deloitt...	Not at all, I love working.
2	2	2.0	Very stressful	Company is very focused on their profits rathe...	Great benefits	Extremely demanding
3	3	5.0	Excellent culture	The project I worked in is relaxed. The overal...	NaN	NaN
4	4	5.0	I love Deloitte!	Honestly if you feel like applying to work at ...	Treated equally	N/a
5	5	5.0	Challenging but incredible career opportunity	This is definitely a challenging and at times ...	NaN	NaN
6	6	3.0	Decent Job	Decent job. I would recommend to others. I jus...	NaN	NaN
7	7	4.0	Wonderful - and challenging place to work.	Unlimited opportunities to take your career in...	NaN	NaN

Figure 1.0 The first ten observations.

The info() function was used to print the summary of the dataset which displayed the total number of columns and entries. The dataset has total number of five (5) variables and four thousand, one hundred and seventy-eight observations (Figure 1.1). The rating and rating_description had 4,179 entries while the rating_title had 4,177 entries. The rating_pros had 1,262 entries and rating_con had

entries of 1,194. The data type for rating was float and the other variables were objects. The observations isnull.sum() function was used to sum up the number of missing values for each variable.

```
In [4]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4179 entries, 0 to 4178
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   rating              4179 non-null   float64
 1   rating_title        4177 non-null   object
 2   rating_description  4179 non-null   object
 3   rating_pros         1262 non-null   object
 4   rating_cons         1194 non-null   object
dtypes: float64(1), object(4)
memory usage: 163.4+ KB
```

Figure 1.1 Variable summary table.

```
In [5]: data.isnull().sum()

Out[5]: rating                 0
        rating_title           2
        rating_description     0
        rating_pros         2917
        rating_cons         2985
        dtype: int64
```

Figure 1.2 Number of missing values for each variable

The analysis focussed more on the two of this variables, rating and rating_description because, it contains the most important information.

1.2 Part of speech tagging

Words are often vague in their part of speech. Ambiguity is usually resolved by the context of the word. Parts of speech tagging is a method that mark up the words in format of text for a certain part of speech based on its context and definition. It is responsible for reading text and assigning precise token to each word. (Yogish 2018). The application of part of-speech tagging exists in many areas including speech synthesis and recognition, machine translation and information retrieval. For the online job review of Deloitte, nouns (NN) were the most abundant part of speech used. Followed by cardinal digit (CD) and preposition/ subordinate conjunction. Wh-determiner, foreign word and coordinating conjunction were the least used parts of speech.

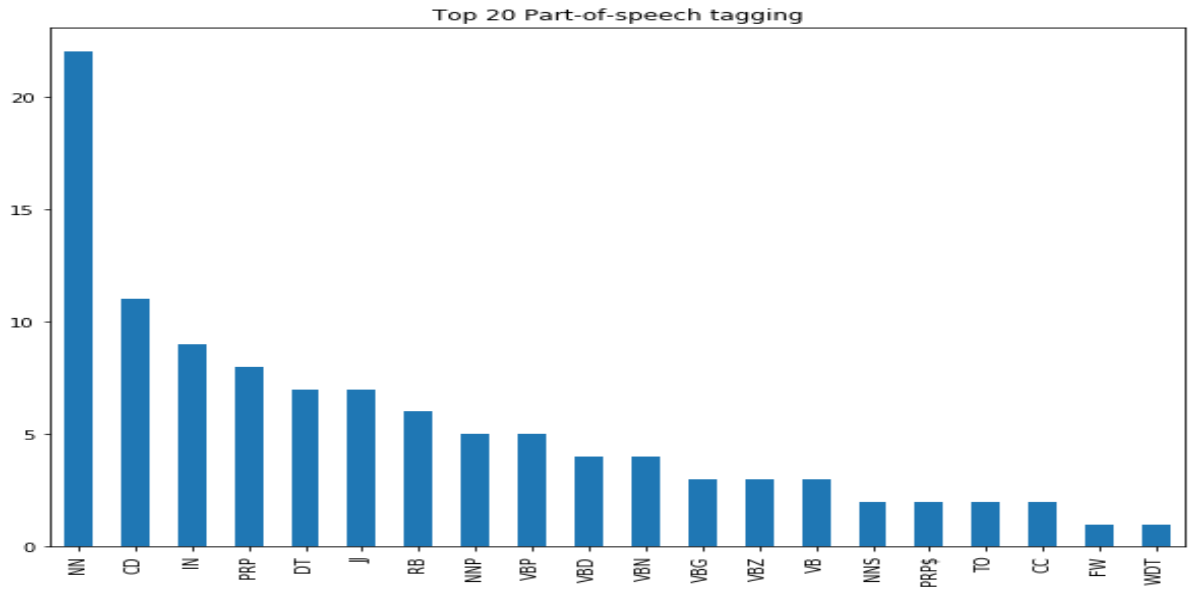
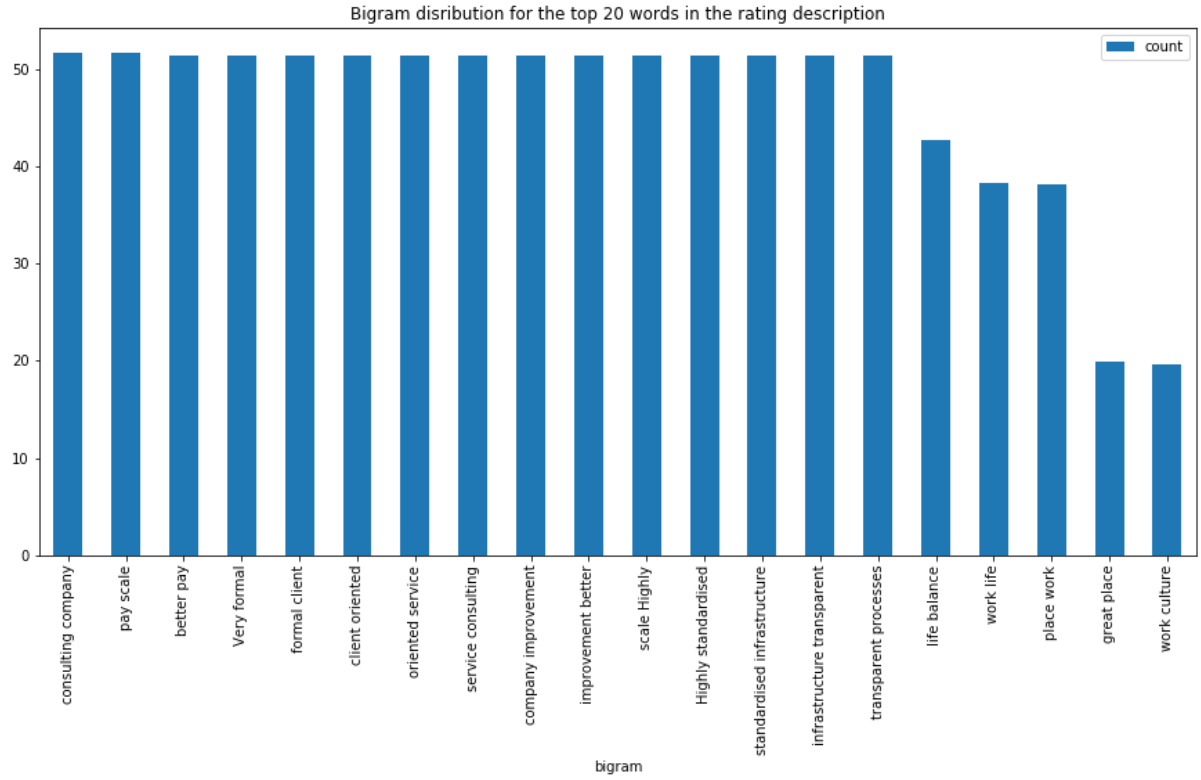


Figure 2.0 Frequency count of the top 20 part of speech tagging (write meaning of the acronyms used in the chart eg NN= nouns).

1.3 The Bigram models

The bigram model approximates the probability of a word given all the prior words by applying only the conditional probability of the prior word. Bigram model can be utilised to predicting the conditional probability of what the next word will be (Martin, S, 1998)

It converts the text rating_description into vector then applies function called TF_IDF vectorizer and transforming the variable and sort the frequency and apply the lambda function to iterate over the toke then save the bigram into dataframe and plot the top 20 bigrams as shown in figure 2.1



1.4 Removing non-ASCII

Non-ASCII is a list of words from a language with alphabet of non-Latin to a standardise form using American Code of Information Interchange (ASCII) formats. A function is created to remove non-ASCII called def_removeNonASCII and to return to create a new variable.

```
In [11]: # Function for removing NonAscii
         def _removeNonAscii(s):
             return "".join(i for i in s if  ord(i)<128)
```

Next is converting the text into lower cases which is one of the common practices in natural language processing. Another function is creating to convert all the text into lower cases called def make_lower-cases.

```
# Function to convert into lower characer
def make_lower_case(text):
    return text.lower()
```

A function called def remove_stop_word was created to remove all the stop words and punctuation.The stopwords are mostly meaningless words that does not add much meaningful information to a sentence.

```
8]: # Function to remove stop words
    def remove_stop_words(text):
        text = text.split()
        stops = set(stopwords.words("english"))
        text = [w for w in text if not w in stops]
        text = " ".join(text)
        return text
```

1.5 Tokenisation

This is the process by which large amounts of text are split into individual small part called token. This token is very important to determine the data patterns. It helps to replacing sensitive data part with non-sensitive data part. A function called def remove_punctuation(text) and to return text and uses regexpTokenizer. This function was applied to the rating description.

```
# Function to remove punctuation
def remove_punctuation(text):
    tokenizer = RegexpTokenizer(r'\w+')
    text = tokenizer.tokenize(text)
    text = " ".join(text)
    return text
```

Applying all the four function created and as cleaned rating_description which is done by calling the function and saving the clean variable in the dataframe using apply function as shown in the table 2.4

```
# Applying all the functions in description and storing as a cleaned_desc
data['rating_desc_cleaned'] = data['rating_description'].apply(_removeNonAscii)
data['rating_desc_cleaned'] = data.rating_desc_cleaned.apply(func = make_lower_case)
data['rating_desc_cleaned'] = data.rating_desc_cleaned.apply(func = remove_stop_words)
data['rating_desc_cleaned'] = data.rating_desc_cleaned.apply(func=remove_punctuation)
```

2.0 NLP task

2.1 Sentiment Analysis (SA)

Sentiment analysis (SA) is the computational study of people's thoughts, ideas, attitudes and feelings toward an entity (Liu 2012). Sentiment Analysis is sometimes considered a classification method. There are three major levels of classification in SA: document level, sentence level, and aspect level (Liu 2012). Document level SA considers the entire document as a basic information unit and classifies an opinion document as expressing a positive or negative sentiment (Wilson, Wiebe & Hoffman 2005). Sentence-level SA classifies the sentiment expressed in each sentence by identifying if the sentence is subjective or objective. If the sentence is subjective, Sentence level SA determines whether the sentence expresses positive or negative opinions (Liu 2012). However, there is no fundamental difference between document and sentence level classifications because sentences are just short documents.

Most employees have become choosier when it comes to employers and companies are finding and developing a new strategy to attract the best qualified candidates. The public image of businesses gives the narrative of its value, mission, and culture. Online review sites such as indeed, seek, and Glassdoor help employer's branding. For instance, any employee, current or former can give a review based on their experience while working with a company and future employees looking to apply for a position can go online and check the company's reviews. This will help the employee decide on if it is worth working with the company. Negative reviews help businesses improve on their brand, because it provides more insight on the areas that require improvement.

Deloitte is one of the big fours of accounting with about 245,000 employees in 150 countries. Many graduates aspire to work with Deloitte, therefore, a sentiment analysis of Deloitte's employees will help employees understand the work culture of the company. It will also provide the management with feedback from their employees to identify areas for improvement.

TextBlob library on python was used to analyse the sentiment. The function requires string, the cleaned variable called rating_desc_cleaned is a string which is then passed into Sentiment() function to calculate it sentiment. The rating variable validates how well the sentiment analysis was able to determine the reviewer's attitude.

```
In [30]: data.head()
```

Out[30]:

	rating	rating_title	rating_description	rating_pros	rating_cons	word_count	rating_desc_cleaned	pos_tags	sentiment	tokenized
0	3.0	Good culture and peers	Very formal client oriented service consulting...	NaN	NaN	19	formal client oriented service consulting comp...	[(f, NN), (o, MD), (r, VB), (m, VB), (a, DT), ...	0.330000	[formal, client, oriented, service, consulting...
1	5.0	Productive and competitive workplace with many...	During a typical day at work we attend any cus...	innovation, leading skills, be part of Deloitt...	Not at all, I love working.	106	typical day work attend customer requirements ...	[(t, NN), (y, NN), (p, NN), (i, NN), (c, VBP),...	0.321528	[typical, day, work, attend, customer, require...
2	2.0	Very stressful	Company is very focused on their profits rathe...	Great benefits	Extremely demanding	32	company focused profits rather accommodating n...	[(c, NNS), (o, VBP), (m, JJ), (p, NN), (a, DT)...	0.200000	[company, focused, profits, rather, accommodat...
3	5.0	Excellent culture	The project I worked in is relaxed. The overal...	NaN	NaN	26	project worked relaxed overall company culture...	[(p, NN), (r, NN), (o, IN), (j, NN), (e, NN), ...	0.267273	[project, worked, relaxed, overall, company, c...
4	5.0	I love Deloitte!	Honestly if you feel like applying to work at ...	Treated equally	N/a	28	honestly feel like applying work deloitte so l...	[(h, NN), (o, MD), (n, VB), (e, NN), (s, JJ), ...	0.500000	[honestly, feel, like, applying, work, deloitt...

Figure 2.1 Sentiment analysis Deloitte's employees' reviews

The scores indicate how positive or negative the reviews are. Scores below 0.05 are negative, scores above 0.1 are tagged as positive and scores between 0.05 and 0.1 are tagged neutral. The overall sentiment rating is tremendously positive (Figure 2.2), implying that employees' perception of Deloitte is positive.

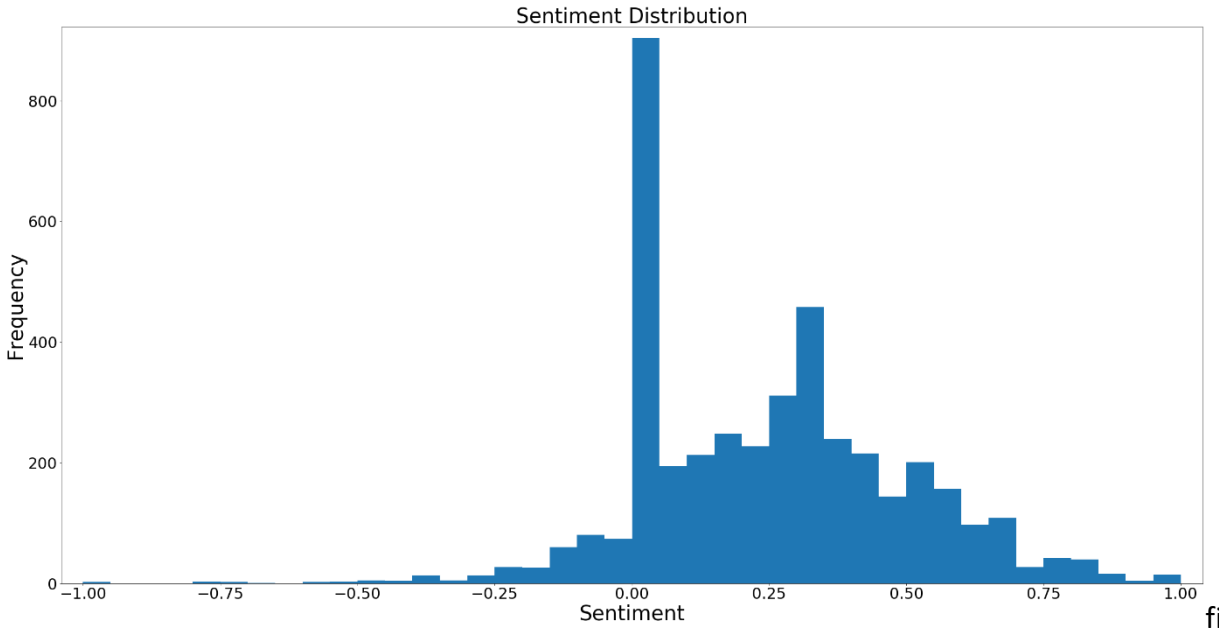


figure 2.2 Sentiment distribution of Deloitte's employees' reviews

Average Sentiment per rating distribution

On the average, 5 star ratings had the highest sentiment, followed by 4, 3, 2 respectively (Figure 2.3). This shows that Deloitte company received a lot of five-star rating.

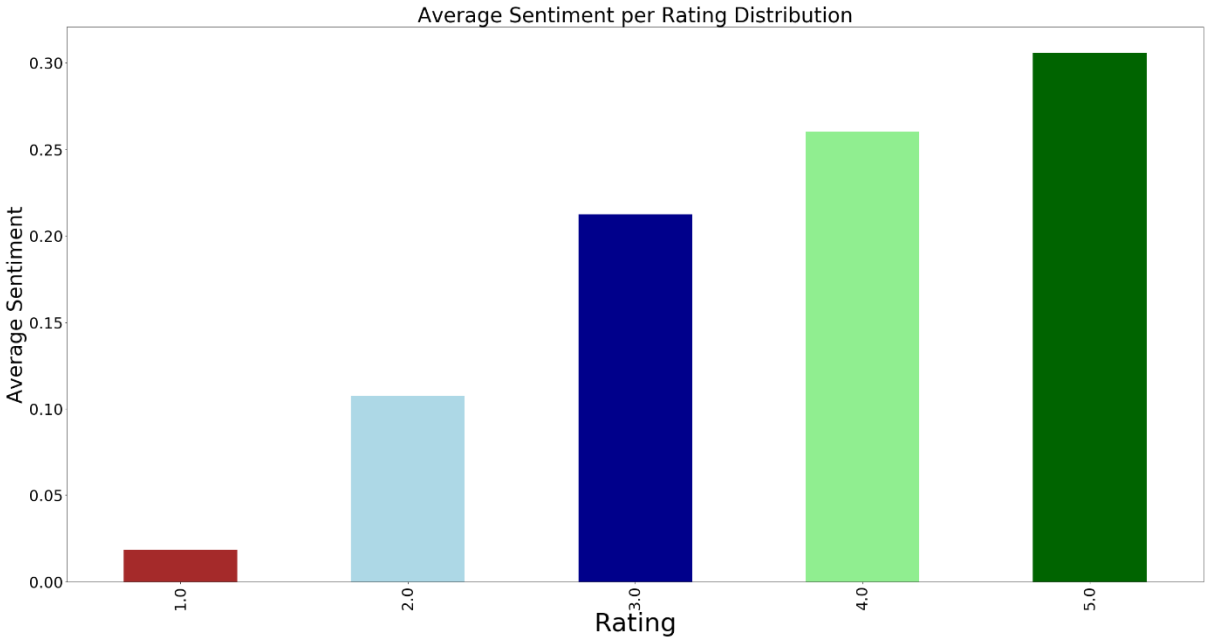


Figure 2.3 Average sentiment of all the rating for Deloitte.

The twenty most frequently words by employees for Deloitte's reviews were; work, deloitte, good and great (2.4).

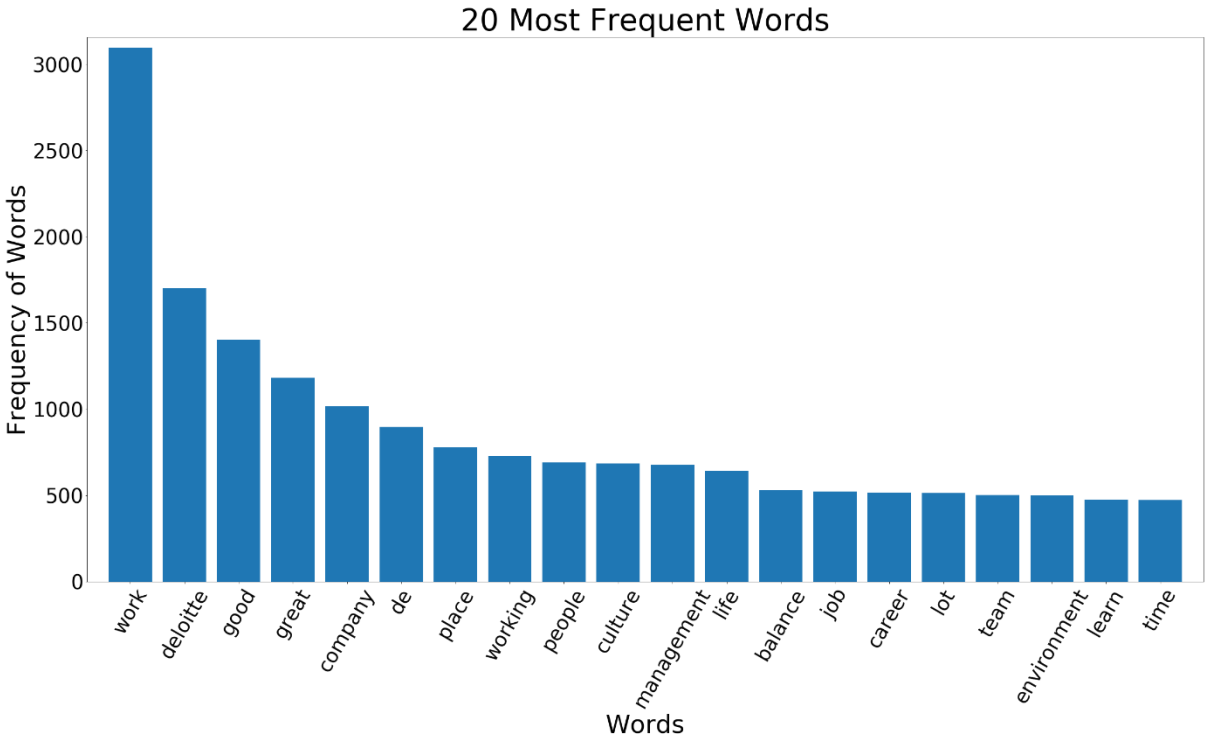


Figure 2.4 The most 20 frequent words for Deloitte employee rating

The figure 2.4.1 The wordcloud chart were plotted to show the top 50 words cloud. It shows that words like 'work', Deloitte, good, great, company, and place are the most frequent used by most of the reviewers.

Top 50 Most Common Words

Figure 2,4.1

Pie chart for the percentage rating in figure 2.5 shows that 1,444 employees rated Deloitte 5 star, making 34.6% of the overall reviews. 34.1% rated Deloitte 4 star, 21.4% rated them 3 star, 5.5% rated them 2 star and 4.5% of the reviewers rated them 1 star (Figure 2.5). This implies that Deloitte company has a very high rates of positive reviews and few negative ones.

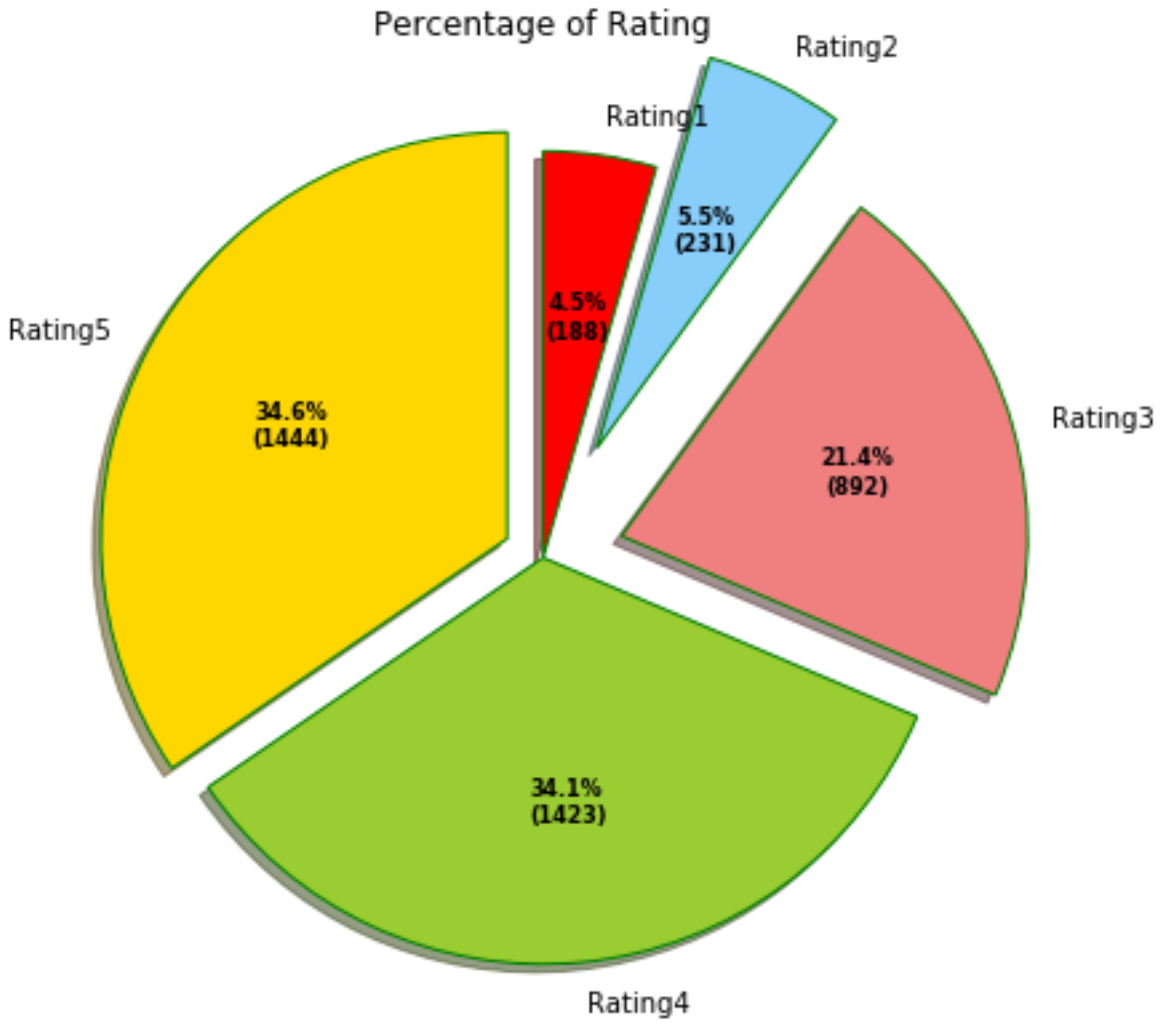


figure 2.5

percentage rating for Deloitte's employee reviews

2.2. Topic Modelling algorithm

Topic models provide a useful technique for dimensionality reduction and exploring data analysis in large text corpora (Arora 2013).

In order to determine each ratings topics, the topic model was implore. Two methods were used; latent Dirichlet Allocation (LDA) topic model method and non-negative matrix factorisation (NMF). Before applying the model, text data was converted into vectors using library called Contvectorizer. It vectorises the tokens and transposes all the words into attributes. It then produces a count of occurrence of each word called document time matrix. The vectorizer item Max_df=90 means that words that are greater than 90% of the reviews should be removed and Min_df to remove words less than 25% from the reviews which then create a spare matrix using fit_fransform() function .it is show in figure 4.0

```
In [75]: # CountVectorizer Method
tf_vectorizer = CountVectorizer(max_df=0.8, min_df=20, max_features=4000)
tf = tf_vectorizer.fit_transform(data['rating_desc_cleaned'].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.head()
```

Out[75]:

	10	12	30	ability	able	accounting	achieve	across	activities	advance	...	workload	workplace	works	world	worth	would	year	years	you	young
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows x 722 columns

Figure 4.0

The LDA is then applied to the document matrix to produce to top 10 topics shown below in figure 4.1

```
In [78]: from sklearn.decomposition import LatentDirichletAllocation
from sklearn.datasets import make_multilabel_classification
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)
```

```
Topic 0:
de la en que el es con para se una
Topic 1:
work project deloitte good firm company employees management one job
Topic 2:
de empresa que uma para com muito se trabalho est
Topic 3:
deloitte great place work career experience learn opportunity professional working
Topic 4:
team client clients year audit every hours industry end system
Topic 5:
consulting service better highly client company pay processes oriented improvement
Topic 6:
challenging di et per un available le non poor professionals
Topic 7:
work job part day many time get things typical management
Topic 8:
work good great life balance culture place company environment management
Topic 9:
deloitte would people time company working work years worked recommend
```

Figure 4.1

The data comprises of their employees all over the world. Therefore, some of the words are in other languages.

A different topic model was explored, using non-negative matrix factorisation (NMF) to get a better accuracy and to see if it can slight more refined. In this case instead of using the countervectorizer method, a method called Term Frequency – Inverse Document Frequency(TF-IDF) was used to twerk down the size and high frequency. Similar approach of CounterVectorizer of Max_90% and Min_25% was applied.

```
In [80]: tfidf_vectorizer = TfidfVectorizer(max_df=0.90, min_df =25, max_features=5000, use_idf=True)
         tfidf = tfidf_vectorizer.fit_transform(data['rating_desc_cleaned'])
         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
```

Out[80]:

	10	12	ability	able	accounting	across	activities	advance	advancement	advisory	...	working	workplace	works	world	worth	would	year	years
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.123135	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.372247	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4174	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
4175	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
4176	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
4177	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
4178	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0

4179 rows × 596 columns

figure 4.2

2.3 Non-Negative Matrix Factorisation (NMF)

```
# Non-Negative Matrix Factorization (NMF)
from sklearn.decomposition import NMF
nmf = NMF(n_components=10, random_state=0, alpha=.1, init='nndsvd').fit(tfidf)
display_topics(nmf, tfidf_feature_names, no_top_words)
```

```
Topic 0:
job team part day management time working hours people work
Topic 1:
standardised formal transparent infrastructure scale improvement oriented processes service highly
Topic 2:
de et travail des trs trabajo les une la trabalho
Topic 3:
good management culture environment learning benefits overall company experience working
Topic 4:
en el que la es se con un una trabajo
Topic 5:
great people place benefits culture opportunities company work learning environment
Topic 6:
work life balance hours culture home difficult flexible however hard
Topic 7:
deloitte working one experience company employees best firm years worked
Topic 8:
empresa para uma com excelente que muito aprender em trabalho
Topic 9:
learn place career lot grow skills start get different new
```

Comparing the result of the two topics model the Non-negative factorisation (NMF) seems to much way better than the latent Dirichlet Allocation (LDA).

Conclusion

The result of the analysis shows that Deloitte employees are very happy working at the company. It shows a very small negative review from a scale of 1 to 5 rating we considered rating 1 and rating 2 as a bad review which come up about less than 10%, rating 3 is more of neutral and rating 4 and 5 has

been considered as a good review. Sentiment analysis confirmed to these results even employees who gave rating 2 and 3 have good score of sentiment average.

Reference

Arora, S., Ge, R., Halpern, Y., Mimno, D., Moitra, A., Sontag, D., ... & Zhu, M. (2013, May). A practical algorithm for topic modeling with provable guarantees. In *International Conference on Machine Learning* (pp. 280-288). PMLR.

B. Liu (2012) Sentiment analysis and opinion mining Synth Lect Human Lang Technol (2012)

Grefenstette, G. (1999). Tokenization. In *Syntactic Wordclass Tagging* (pp. 117-133). Springer,

Dordrecht.

Martin, S., Liermann, J., & Ney, H. (1998). Algorithms for bigram and trigram word clustering. *Speech*

communication, *24*(1), 19-37.

Wang, C., & Blei, D. M. (2011) Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 448-456).

Wilson T, Wiebe J, Hoffman P. Recognizing contextual polarity in phrase-level sentiment analysis. In: Proceedings of HLT/EMNLP; 2005.

Yau, C. K., Porter, A., Newman, N., & Suominen, A. (2014). Clustering scientific documents with topic modeling. *Scientometrics*, *100*(3), 767-786.

Yogish, D., Manjunath, T. N., & Hegadi, R. S. (2018, December). Review on Natural language processing

trends and techniques using NLTK. In *International Conference on Recent Trends in Image Processing*

and Pattern Recognition (pp. 589-606). Springer, Singapore.

Appendix

```
# Libraries
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import re
import nltk
import string
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords, wordnet
from nltk.stem import WordNetLemmatizer
from textblob import TextBlob
import string
import contractions
from nltk.tokenize import RegexpTokenizer
import fasttext
# loading the data

data = pd.read_csv('indeed_data.csv')

data.head(10)
# removing the unnamed column

data.drop('Unnamed: 0', axis=1, inplace=True)

# summary of the data

data.info()
# calculating the number of NA

data.isnull().sum()
# The  20 top of  part-of-speech tags in the rating descriptions



from textblob import TextBlob

text = TextBlob(str(data['rating_description']))

df_pos = pd.DataFrame(text.tags, columns = ['word' , 'pos'])

df_pos = df_pos.pos.value_counts()[:20]

df_pos.plot(kind = 'bar', figsize=(10, 8), title = "Top 20 Part-of-speech tagging")
# Bigram Distribution



# Converting text rating descriptions into vectors using TF-IDF
```

```
from sklearn.feature_extraction.text import CountVectorizer

from sklearn.feature_extraction.text import TfidfVectorizer

tf_vector = TfidfVectorizer(ngram_range=(2, 2), stop_words='english', lowercase = False)

matrix_tf = tf_vector.fit_transform(data['rating_description'])

words_count = matrix_tf.sum(axis=0)

#Finding the word frequency

frequency = [(word, words_count[0, idx]) for word, idx in tf_vector.vocabulary_.items()]

frequency =sorted(frequency, key = lambda x: x[1], reverse=True)

#converting into dataframe

bigram = pd.DataFrame(frequency)

bigram.rename(columns = {0:'bigram', 1: 'count'}, inplace = True)

#Taking first 20 records

bigram = bigram.head(20)

#Plotting the bigram distribution

bigram.plot(x ='bigram', y='count', kind = 'bar', title = ''Bigram disribution for the top 20 words in the
rating description'', figsize = (15,7), )
# Function for removing NonAscii

def _removeNonAscii(s):

   return "".join(i for i in s if  ord(i)<128)

# Function to convert into lower characer

def make_lower_case(text):

   return text.lower()
# Function to remove stop words

def remove_stop_words(text):

   text = text.split()

   stops = set(stopwords.words("english"))

   text = [w for w in text if not w in stops]
```

```
    text = " ".join(text)
    return text

# Function to remove punctuation
def remove_punctuation(text):
    tokenizer = RegexpTokenizer(r'\w+')
    text = tokenizer.tokenize(text)
    text = " ".join(text)
    return text

# Applying all the functions in description and storing as a cleaned_desc
data['rating_desc_cleaned'] = data['rating_description'].apply(_removeNonAscii)
data['rating_desc_cleaned'] = data.rating_desc_cleaned.apply(func = make_lower_case)
data['rating_desc_cleaned'] = data.rating_desc_cleaned.apply(func = remove_stop_words)
data['rating_desc_cleaned'] = data.rating_desc_cleaned.apply(func=remove_punctuation)
data['pos_tags'] = data['rating_desc_cleaned'].apply(nltk.tag.pos_tag)
data['tokenized'] = data['rating_desc_cleaned'].apply(word_tokenize)
# plotting the sentiment distribution
plt.figure(figsize=(40,20))
plt.margins(0.02)
plt.xlabel('Sentiment', fontsize = 40)
plt.xticks(fontsize=30)
plt.yticks(fontsize=30)
plt.ylabel('Frequency', fontsize = 40)
plt.hist(data['sentiment'], bins = 40)
plt.title('Sentiment Distribution', fontsize=40)
plt.show()
colors = ['brown', 'lightblue', 'darkblue', 'lightgreen', 'darkgreen']
```

```
avg = data.groupby('rating')['sentiment'].mean().plot(kind='bar', color = colors, figsize=(40,20))
plt.xlabel('Rating', fontsize=50)
plt.ylabel('Average Sentiment', fontsize=40)
plt.xticks(fontsize=30)
plt.yticks(fontsize=30)
plt.title('Average Sentiment per Rating Distribution', fontsize=40)
plt.show()
# Creating explode data
explode = (0.1, 0.0, 0.2, 0.3, 0.0)
labels = 'Rating5', 'Rating4', 'Rating3', 'Rating2','Rating1'
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue', 'red']


# Wedge properties
wp = { 'linewidth' : 1, 'edgecolor' : "green" }


# Creating autocpt arguments
def func(pct, allvalues):
    absolute = int(pct / 100.*np.sum(allvalues))
    return "{:.1f}%\n({:d})".format(pct, absolute)
# Creating plot
fig, ax = plt.subplots(figsize =(10, 7))
wedges, texts, autotexts = ax.pie(data.rating.value_counts(),
                                  autopct = lambda pct: func(pct, data.rating.value_counts()),
                                  explode = explode,
                                  labels = labels,
```

```
                    shadow = True,

                    colors = colors,

                    startangle = 90,

                    wedgeprops = wp,

                    textprops = dict(color ="black"))



plt.setp(autotexts, size = 8, weight ="bold")

ax.set_title("Percentage of Rating")


# show plot

plt.show()
words = data['tokenized']

allwords = []

for wordlist in words:

   allwords +=wordlist

print(allwords)

from nltk.tokenize import word_tokenize

from nltk.probability import FreqDist

from wordcloud import WordCloud


fdist = FreqDist()

mostcommon = FreqDist(allwords).most_common(100)

wordcloud = WordCloud(width=1600, height=800,
background_color='blue').generate(str(mostcommon))

fig = plt.figure(figsize=(30,10), facecolor='white')
```

```
plt.imshow(wordcloud, interpolation="bilinear")

plt.axis('off')

plt.title('Top 50 Most Common Words', fontsize=100)

plt.tight_layout(pad=0)

plt.show()

# plot 20 most frequent words

common_word = FreqDist(allwords).most_common(20)

x, y = zip(*common_word)

plt.figure(figsize=(40,20))

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('20 Most Frequent Words', fontsize=60)

plt.show()
# CountVectorizer Method

tf_vectorizer = CountVectorizer(max_df=0.8, min_df=20, max_features=4000)

tf = tf_vectorizer.fit_transform(data['rating_desc_cleaned'].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.head()
from sklearn.decomposition import LatentDirichletAllocation

from sklearn.datasets import make_multilabel_classification

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)
# Non-Negative Matrix Factorization (NMF)

from sklearn.decomposition import NMF

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

display_topics(nmf, tfidf_feature_names, no_top_words)
```