Feature Engineering

This section will cover the following types of features for the Yelp reviews:

1. Bag of Words
2. Bag of N-Grams
3. TF-IDF (term frequency over inverse document frequency)

In [17]:

```
import pandas as pd
import numpy as np
import re
import nltk
```

The corpus or the reviews were extracted from the Yelp review dataset using pandas

In [18]:

```
corpus_df = pd.read_csv('yelp_review10K.csv')
corpus = corpus_df['text']
corpus.head()
```

Out[18]:

```
0    My wife took me here on my birthday for breakf...
1    I have no idea why some people give bad review...
2    love the gyro plate. Rice is so good and I als...
3    Rosie, Dakota, and I LOVE Chaparral Dog Park!!...
4    General Manager Scott Petello is a good egg!!!...
Name: text, dtype: object
```

Text pre-processing

In [ ]:

```
As part of Text pre-processing we removed the special characters, whitespaces and
lower case.
```

In [44]:

```
wpt = nltk.WordPunctTokenizer()
stop_words = nltk.corpus.stopwords.words('english')

def normalize_document(doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z\s]', '', doc, re.I)
    # doc = re.sub(r'[^a-zA-Z0-9\s]', '', doc, re.I)
    doc = doc.lower()
    doc = doc.strip()
    # tokenize document
    tokens = wpt.tokenize(doc)
    # filter stopwords out of document
    filtered_tokens = [token for token in tokens if token not in stop_words]
    # re-create document from filtered tokens
    doc = ' '.join(filtered_tokens)
    doc = ''.join(i for i in doc if not i.isdigit())
    return doc

normalize_corpus = np.vectorize(normalize_document)
```

In [45]:
```
norm_corpus = normalize_corpus(corpus)
norm_corpus
```

Out[45]:
```
array(['wife took birthday breakfast excellent weather perfect made sitting out
side overlooking grounds absolute pleasure waitress excellent food arrived quic
kly semi - busy saturday morning . looked like place fills pretty quickly earli
er get better . favor get bloody mary . phenomenal simply best \' ever . \' pre
tty sure use ingredients garden blend fresh order . amazing . everything menu l
ooks excellent , white truffle scrambled eggs vegetable skillet tasty delicious
. came  pieces griddled bread amazing absolutely made meal complete . best " to
ast " \' ever . anyway , \' wait go back !',
       'idea people give bad reviews place goes show please everyone . probably
griping something fault ... many people like . case , friend arrived  :  pm pas
t sunday . pretty crowded , thought sunday evening thought would wait forever g
et seat said \' seated girl comes back seating someone else . seated  :  waiter
came got drink orders . everyone pleasant host seated us waiter server . prices
good well . placed orders decided wanted  :  . shared baked spaghetti calzone s
mall " \' beef " pizza try . calzone huge got smallest one ( personal ) got sma
ll  " pizza . awesome ! friend liked pizza better liked calzone better . calzon
e sweetish sauce \' like sauce ! box part pizza take home door  :  . , everythi
ng great like bad reviewers . goes show try things bad reviewers serious issues
.',
       'love gyro plate rice good also dig candy selection )', ...,
       "recently visited olive ivy business last week visits , convinced fox re
staurants best establishments valley . olive ivy fox restaurant choice consiste
ntly good food , great drinks , , outstanding service . spend lot time various
restaurants across valley always amazed bad service popular valley restaurants
. olive ivy . first phone call reservations , greeting upon walking door smiles
warm reception receive every server cross , restaurant knows make feel special
. many reviews focus food could spend hours talking experiences sum couple favo
rites . hate dates prunes , bacon wrapped dates crazy good ! shrimp risotto mai
n course one best dishes tasted since arriving phoenix  years ago . hands , sho
rt rib entree best valley . catch cost . olive ivy cheap definitely brothers lo
oking execute cheap date . wine , apps , main course , easy break century mark
, really generous , $  obtainable . , ' worth every dollar knows , may even get
breakfast deal !",
       'nephew moved scottsdale recently bunch friends brought show local bar g
irlfriend could come shoot pool watch football play volleyball etc ... well
.... \' minutes  kids running around pool tables , messing games screaming . \'
believe staff allowing happen . hitting pool sticks everything crying mom attem
pted ( vain ) quiet . \' think mom would leave point kids .... um ... .... staf
f seem annoyed well said nothing . .... happened ... said " guys better behave
mommy fired "!! holy shit .... works !! even worse ! shame owner allowing happe
n . employee needs recognize ... work bar ..... daycare .... bar !!!',
       ' locations ..  .  star average .. think arizona really fantastic pizza
options , spinatos top pizza fix list .. semi sweet sauce addictive , great ser
vice , fresh ingredients , spicy italian favorite ... chocolate chips cookies l
aced mind altering drugs , make body parts " romantic " think cookies .. btw  :
pm , tuesday , left ,  minute wait .. list ..'],
      dtype='<U3709')
```

1. Bag of Words Model

```
In [ ]: We created the Bag of Words model to determine the unique words in each document a
```

```
In [46]: from sklearn.feature_extraction.text import CountVectorizer

         cv = CountVectorizer(min_df=0., max_df=1.)
         cv_matrix = cv.fit_transform(norm_corpus)
         cv_matrix = cv_matrix.toarray()
         cv_matrix
```

```
Out[46]: array([[0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                ...,
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

Thus you can see that our documents have been converted into numeric vectors such that each document is represented by one vector (row) in the above feature matrix. The following code will help represent this in a more easy to understand format.

```
In [47]: # get all unique words in the corpus
         vocab = cv.get_feature_names()
         vocab
```

```
Out[47]: ['______',
          '_______',
          '_______________',
          '____berto',
          '_accommodating',
          '_c',
          '_finally_',
          '_gyibeahdfylsszc_g',
          '_lozhqednolhvbg',
          '_reasonable',
          '_she',
          '_third_',
          '_us_',
          '_very',
          '_xhxtuykqnyphmylm',
          'aa',
          'aaa',
          'aaaaaalright',
          'aaaamazing',
          'aaammmazing'
```

In [48]:

```
# show document feature vectors
pd.DataFrame(cv_matrix, columns=vocab)
```

Out[48]:

	______	_______	_______________	____berto	_accommodating	_c	_finally_	_gyibeahdfylss
0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	
15	0	0	0	0	0	0	0	
16	0	0	0	0	0	0	0	
17	0	0	0	0	0	0	0	
18	0	0	0	0	0	0	0	
19	0	0	0	0	0	0	0	
20	0	0	0	0	0	0	0	
21	0	0	0	0	0	0	0	
22	0	0	0	0	0	0	0	
23	0	0	0	0	0	0	0	
24	0	0	0	0	0	0	0	
25	0	0	0	0	0	0	0	
26	0	0	0	0	0	0	0	
27	0	0	0	0	0	0	0	
28	0	0	0	0	0	0	0	
29	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	
9970	0	0	0	0	0	0	0	
9971	0	0	0	0	0	0	0	
9972	0	0	0	0	0	0	0	

	______	_______	_______________	____berto	_accommodating	_c	_finally_	_gyibeahdfylss
9973	0	0	0	0	0	0	0	
9974	0	0	0	0	0	0	0	
9975	0	0	0	0	0	0	0	
9976	0	0	0	0	0	0	0	
9977	0	0	0	0	0	0	0	
9978	0	0	0	0	0	0	0	
9979	0	0	0	0	0	0	0	
9980	0	0	0	0	0	0	0	
9981	0	0	0	0	0	0	0	
9982	0	0	0	0	0	0	0	
9983	0	0	0	0	0	0	0	
9984	0	0	0	0	0	0	0	
9985	0	0	0	0	0	0	0	
9986	0	0	0	0	0	0	0	
9987	0	0	0	0	0	0	0	
9988	0	0	0	0	0	0	0	
9989	0	0	0	0	0	0	0	
9990	0	0	0	0	0	0	0	
9991	0	0	0	0	0	0	0	
9992	0	0	0	0	0	0	0	
9993	0	0	0	0	0	0	0	
9994	0	0	0	0	0	0	0	
9995	0	0	0	0	0	0	0	
9996	0	0	0	0	0	0	0	
9997	0	0	0	0	0	0	0	
9998	0	0	0	0	0	0	0	
9999	0	0	0	0	0	0	0	

10000 rows × 28947 columns

2. Bag of N-Grams Model

In [ ]:

```
We created the Bag of bi-grams and tri-grams to look at the 2-word and 3-word stri
```

In [49]:

```
bv = CountVectorizer(ngram_range=(2,2))
bv_matrix = bv.fit_transform(norm_corpus)

bv_matrix = np.asarray(bv_matrix)
vocab = bv.get_feature_names()
# pd.DataFrame(bv_matrix, columns=vocab)
vocab
```

Out[49]:

```
['______ ordered',
 '_______ oakland',
 '_______________ update',
 '____berto matter',
 '_accommodating evening',
 '_finally_ found',
 '_gyibeahdfylsszc_g adventures',
 '_lozhqednolhvbg http',
 '_reasonable amount',
 '_she listens',
 '_she pretty',
 '_third_ visit',
 '_us_ going',
 '_very friendly',
 '_xhxtuykqnyphmylm mqg',
 'aa accessories',
 'aa battery',
 'aa coming',
 'aa give',
```

In [50]:

```
bv = CountVectorizer(ngram_range=(3,3))
bv_matrix = bv.fit_transform(norm_corpus)

bv_matrix = np.asarray(bv_matrix)
vocab = bv.get_feature_names()
vocab
```

Out[50]:

```
['______ ordered chicken',
 '_______ oakland coliseum',
 '_______________ update first',
 '____berto matter basically',
 '_accommodating evening appointments',
 '_finally_ found place',
 '_gyibeahdfylsszc_g adventures phoenix',
 '_lozhqednolhvbg http www',
 '_reasonable amount time',
 '_she listens every',
 '_she pretty busy',
 '_third_ visit since',
 '_us_ going wonder',
 '_very friendly _accommodating',
 '_xhxtuykqnyphmylm mqg dessert',
 'aa accessories fab',
 'aa battery something',
 'aa coming xl',
 'aa give call',
```

3. TF-IDF Model

In [51]:

```python
from sklearn.feature_extraction.text import TfidfVectorizer

tv = TfidfVectorizer(min_df=0., max_df=1., use_idf=True)
tv_matrix = tv.fit_transform(norm_corpus)
tv_matrix = tv_matrix.toarray()

vocab = tv.get_feature_names()
pd.DataFrame(np.round(tv_matrix, 2), columns=vocab)
```

Out[51]:

	______	_______	_______________	____berto	_accommodating	_c	_finally_	_gyibeahdfyls
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	0.0	0.0	
2	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	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	______	_______	_______________	____berto	_accommodating	_c	_finally_	_gyibeahdfyls
29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	
9970	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9971	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9972	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9973	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9974	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9975	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9976	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9977	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9978	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9979	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9980	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9981	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9982	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9983	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9984	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9985	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9986	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9987	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9988	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9989	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9990	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9991	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9992	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9993	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9994	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

10000 rows × 28947 columns

The TF-IDF based feature vectors for each of our text documents show scaled and normalized values as compared to the raw Bag of Words model values.