In [19]:
```python
import sys
import os


current_work_directory = os.getcwd()    # Return a string representing the current
print('Current work directory: {}'.format(current_work_directory))
# Make sure it's an absolute path.
abs_work_directory = os.path.abspath(current_work_directory)
print('Current work directory (full path): {}'.format(abs_work_directory))
print()

filename = 'yelp_review10K.csv'
# Check whether file exists.
if not os.path.isfile(filename):
    # Stop with leaving a note to the user.
    print('It seems file "{}" not exists in directory: "{}"'.format(filename, curr
    sys.exit(1)

import csv

with open('yelp_review10K.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    for row in reader:
        print(row)
```

```
Current work directory: C:\Users\shabe
Current work directory (full path): C:\Users\shabe

['business_id', 'date', 'review_id', 'stars', 'text', 'type', 'user_id', 'co
ol', 'useful', 'funny']
['9yKzy9PApeiPPOUJEtnvkg', '1/26/2011', 'fWKvX83p0-ka4JS3dc6E5A', '5', 'My w
ife took me here on my birthday for breakfast and it was excellent.  The wea
ther was perfect which made sitting outside overlooking their grounds an abs
olute pleasure.  Our waitress was excellent and our food arrived quickly on 
the semi-busy Saturday morning.  It looked like the place fills up pretty qu
ickly so the earlier you get here the better.\n\nDo yourself a favor and get
their Bloody Mary.  It was phenomenal and simply the best I\'ve ever had.  I
\'m pretty sure they only use ingredients from their garden and blend them f
resh when you order it.  It was amazing.\n\nWhile EVERYTHING on the menu loo
ks excellent, I had the white truffle scrambled eggs vegetable skillet and i
t was tasty and delicious.  It came with 2 pieces of their griddled bread wi
th was amazing and it absolutely made the meal complete.  It was the best "t
oast" I\'ve ever had.\n\nAnyway, I can\'t wait to go back!', 'review', 'rLtl
8ZkDX5vH5nAx9C3q5Q', '2', '5', '0']
```

In [21]:
```
import pandas as pd
from pandas import DataFrame

ReadCsv = pd.read_csv (r'C:\Users\shabe\yelp_review10K.csv')

df = DataFrame(ReadCsv,columns=['business_id','date','review_id','stars','text','t

print (df)


from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer

no_features = 1000

# NMF is able to use tf-idf
tfidf_vectorizer = TfidfVectorizer(max_df=0.95, min_df=2, max_features=no_features
tfidf = tfidf_vectorizer.fit_transform(df['text'])
tfidf_feature_names = tfidf_vectorizer.get_feature_names()

# LDA can only use raw term counts for LDA because it is a probabilistic graphical
tf_vectorizer = CountVectorizer(max_df=0.95, min_df=2, max_features=no_features, s
tf = tf_vectorizer.fit_transform(df['text'])
tf_feature_names = tf_vectorizer.get_feature_names()
```

```
                  business_id        date               review_id  stars  \
0      9yKzy9PApeiPPOUJEtnvkg   1/26/2011  fWKvX83p0-ka4JS3dc6E5A      5
1      ZRJwVLyzEJq1VAihDhYiow   7/27/2011  IjZ33sJrzXqU-0X6U8NwyA      5
2      6oRAC4uyJCsJl1X0WZpVSA   6/14/2012  IESLBzqUCLdSzSqm0eCSxQ      4
3      _1QQZuf4zZOyFCvXc0o6Vg   5/27/2010  G-WvGaISbqqaMHlNnByodA      5
4      6ozycU1RpktNG2-1BroVtw    1/5/2012  1uJFq2r5QfJG_6ExMRCaGw      5
5                      #NAME?  12/13/2007  m2CKSsepBCoRYWxiRUsxAg      4
6      zp713qNhx8d9KCJJnrw1xA   2/12/2010  riFQ3vxNpP4rWLk_CSri2A      5
7      hW0Ne_HTHEAgGF1rAdmR-g   7/12/2012  JL7GXJ9u4YMx7Rzs05NfiQ      4
8      wNUea3IXZWD63bbOQaOH-g   8/17/2012  XtnfnYmnJYi71yIuGsXIUA      4
9      nMHhuYan8e3cONo3PornJA   8/11/2010  jJAIXA46pU1swYyRCdfXtQ      5
10     AsSCv0q_BWqIe3mX2JqsOQ   6/16/2010  E11jzpKz9Kw5K7fuARWfRw      5
11     e9nN4XxjdHj4qtKCOPq_vg  10/21/2011  3rPt0LxF7rgmEUrznoH22w      5
12     h53YuCiIDfEFSJCQpk8v1g   1/11/2010  cGnKNX3I9rthE0-TH24-qA      5
13     WGNIYMeXPyoWav1APUq7jA  12/23/2011  FvEEw1_OsrYdvwLV5Hrliw      4
14     yc5AH9H71xJidA_J2mChLA   5/20/2010  pfUwBKYYmUXeiwrhDluQcw      4
15     Vb9FPCEL6Ly24PNxLBaAFw   3/20/2011  HvqmdqWcerVWO3Gs6zbrOw      2
16     supigcPNO9IKo6olaTNV-g  10/12/2008  HXP_0Ul-FCmA4f-k9CqvaQ      3
17     O51ORe68mOy9dU490JTKCg    5/3/2010  j4SIzrIy0WrmW4yr4--Khg      5
```

In [22]:
```
from sklearn.decomposition import NMF, LatentDirichletAllocation

no_topics = 20

# Run NMF
nmf = NMF(n_components=no_topics, random_state=1, alpha=.1, l1_ratio=.5, init='nnd

# Run LDA
lda = LatentDirichletAllocation(n_topics=no_topics, max_iter=5, learning_method='o
```

```
C:\Users\shabe\Anaconda3\lib\site-packages\sklearn\decomposition\online_lda.p
y:294: DeprecationWarning: n_topics has been renamed to n_components in versio
n 0.19 and will be removed in 0.21
  DeprecationWarning)
```

In [26]:
```
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]]))

no_top_words = 10
display_topics(nmf, tfidf_feature_names, no_top_words)
display_topics(lda, tf_feature_names, no_top_words)
```

```
Topic 0:
like just time place don really know people going got
Topic 1:
great place atmosphere prices awesome fun recommend selection fantastic defini
tely
Topic 2:
food mexican restaurant chinese place eat fast better quality atmosphere
Topic 3:
pizza crust wings pizzas slices toppings pie slice sauce cheese
Topic 4:
ordered cheese sauce delicious menu fresh restaurant meal dinner bread
Topic 5:
love place amazing favorite delicious yummy especially family wish awesome
Topic 6:
burger fries burgers sweet potato bun cheese bacon onion crispy
Topic 7:
bar beer night drinks wine music drink selection sports patio
Topic 8:
service excellent customer slow bad food friendly server times fast
Topic 9:
good pretty really place prices price decent times selection tasty
Topic 10:
best ve phoenix town valley years times amazing eaten hands
Topic 11:
staff friendly clean helpful fast super dr wait recommend office
Topic 12:
breakfast eggs pancakes morning toast bacon burrito brunch french wait
Topic 13:
sushi roll rolls tuna fish spicy fresh quality chef salmon
Topic 14:
coffee starbucks shop iced cup tea donuts morning chocolate drink
Topic 15:
happy hour specials appetizers drinks menu day half pretty drink
Topic 16:
salad sandwich lunch sandwiches bread dressing turkey salads soup cheese
Topic 17:
store selection prices items stores shopping buy grocery shop location
Topic 18:
nice hotel clean room area pool stay rooms patio really
Topic 19:
chicken fried rice thai spicy chinese curry sauce soup beef
Topic 0:
store shop selection great prices work buy items need like
Topic 1:
staff friendly recommend highly helpful dr office course nice extremely
Topic 2:
good cheese sauce chicken beef meat like tacos just chips
```

```
Topic 3:
good food place coffee breakfast service like just bad really
Topic 4:
food mexican chinese better taco restaurant best like hot authentic
Topic 5:
just love place like ice cream dog don know want
Topic 6:
like just time really going did people place new right
Topic 7:
location variety frozen toppings cookies fresh italian yogurt locations chandl
er
Topic 8:
room phoenix area scottsdale pool hotel stay nice town old
Topic 9:
happy hour cake birthday day event drink included chocolate wonderful
Topic 10:
ve sushi place best time food like roll wait times
Topic 11:
amazing favorite love great husband best kids perfect make loved
Topic 12:
menu salad dishes dish soup restaurant food shrimp delicious chicken
Topic 13:
sandwich chicken bbq awesome corn wings salad sandwiches favorites pork
Topic 14:
like burger good fries really just place don time know
Topic 15:
pizza ordered table food server good order just restaurant came
Topic 16:
minutes service said told time got went asked didn 10
Topic 17:
free star ribs stars airport com coupon double rating blue
Topic 18:
bar place great beer night good drinks music food patio
Topic 19:
great food good place service lunch thai love really friendly
```

In [30]:

```
#def display_topics(H, W, feature_names, df, no_top_words, no_top_documents):
#    for topic_idx, topic in enumerate(H):
#        print ("Topic %d:" % (topic_idx))
#        print (" ".join([feature_names[i]
#                        for i in topic.argsort()[:-no_top_words - 1:-1]]))
#        top_doc_indices = np.argsort( W[:,topic_idx] )[::-1][0:no_top_documents]
#        for doc_index in top_doc_indices:
#            print (documents[doc_index])
```

In [33]:
```
#from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
#from sklearn.decomposition import NMF, LatentDirichletAllocation
#import numpy as np

#def display_topics(H, W, feature_names, documents, no_top_words, no_top_documents
#    for topic_idx, topic in enumerate(H):
#        print ("Topic %d:" % (topic_idx))
#        print (" ".join([feature_names[i]
#                        for i in topic.argsort()[:-no_top_words - 1:-1]]))
#        top_doc_indices = np.argsort( W[:,topic_idx] )[::-1][0:no_top_documents]
#        for doc_index in top_doc_indices:
#            print (documents[doc_index])

#import pandas as pd
#from pandas import DataFrame

#ReadCsv = pd.read_csv (r'C:\Users\shabe\yelp_review10K.csv')

#documents = DataFrame(ReadCsv,columns=['business_id','date','review_id','stars','


#no_features = 1000

# NMF is able to use tf-idf
#tfidf_vectorizer = TfidfVectorizer(max_df=0.95, min_df=2, max_features=no_feature
#tfidf = tfidf_vectorizer.fit_transform(documents['text'])
#tfidf_feature_names = tfidf_vectorizer.get_feature_names()

# LDA can only use raw term counts for LDA because it is a probabilistic graphical
#tf_vectorizer = CountVectorizer(max_df=0.95, min_df=2, max_features=no_features,
#tf = tf_vectorizer.fit_transform(documents['text'])
#tf_feature_names = tf_vectorizer.get_feature_names()

#no_topics = 5

# Run NMF
#nmf_model = NMF(n_components=no_topics, random_state=1, alpha=.1, l1_ratio=.5, in
#nmf_W = nmf_model.transform(tfidf)
#nmf_H = nmf_model.components_

# Run LDA
#lda_model = LatentDirichletAllocation(n_topics=no_topics, max_iter=5, learning_me
#lda_W = lda_model.transform(tf)
#lda_H = lda_model.components_

#no_top_words = 5
#no_top_documents = 2
#display_topics(nmf_H, nmf_W, tfidf_feature_names, documents, no_top_words, no_top
#display_topics(lda_H, lda_W, tf_feature_names, documents, no_top_words, no_top_do
```

```
C:\Users\shabe\Anaconda3\lib\site-packages\sklearn\decomposition\online_lda.p
y:294: DeprecationWarning: n_topics has been renamed to n_components in versio
n 0.19 and will be removed in 0.21
  DeprecationWarning)

Topic 0:
```

```
like just time place don

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, ke
y, method, tolerance)
   3077             try:
-> 3078                 return self._engine.get_loc(key)
   3079             except KeyError:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHash
Table.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHash
Table.get_item()

KeyError: 9763

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-33-5335f0ada2c4> in <module>()
     47 no_top_words = 5
     48 no_top_documents = 2
---> 49 display_topics(nmf_H, nmf_W, tfidf_feature_names, documents, no_top_wo
rds, no_top_documents)
     50 display_topics(lda_H, lda_W, tf_feature_names, documents, no_top_words
, no_top_documents)

<ipython-input-33-5335f0ada2c4> in display_topics(H, W, feature_names, documen
ts, no_top_words, no_top_documents)
     11         top_doc_indices = np.argsort( W[:,topic_idx] )[::-1][0:no_top_
documents]
     12         for doc_index in top_doc_indices:
---> 13             print (documents[doc_index])
     14 
     15 import pandas as pd

~\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   2686             return self._getitem_multilevel(key)
   2687         else:
-> 2688             return self._getitem_column(key)
   2689 
   2690     def _getitem_column(self, key):

~\Anaconda3\lib\site-packages\pandas\core\frame.py in _getitem_column(self, ke
y)
   2693         # get column
   2694         if self.columns.is_unique:
-> 2695             return self._get_item_cache(key)
   2696 
   2697         # duplicate columns & possible reduce dimensionality
```

```
~\Anaconda3\lib\site-packages\pandas\core\generic.py in _get_item_cache(self,
 item)
   2487         res = cache.get(item)
   2488         if res is None:
-> 2489             values = self._data.get(item)
   2490             res = self._box_item_values(item, values)
   2491             cache[item] = res

~\Anaconda3\lib\site-packages\pandas\core\internals.py in get(self, item, fast
path)
   4113 
   4114             if not isna(item):
-> 4115                 loc = self.items.get_loc(item)
   4116             else:
   4117                 indexer = np.arange(len(self.items))[isna(self.items)]

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, ke
y, method, tolerance)
   3078                 return self._engine.get_loc(key)
   3079             except KeyError:
-> 3080                 return self._engine.get_loc(self._maybe_cast_indexer(k
ey))
   3081 
   3082         indexer = self.get_indexer([key], method=method, tolerance=tol
erance)

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHash
Table.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHash
Table.get_item()

KeyError: 9763
```