Week 2 - Homework - NLP

Improve your analysis from Week-1 by identifying specific words / keywords / N-Grams that lead to failed food inspections in Chicago leveraging tokenization, stemming, lemmatization and N-Gramming in Python.

Rules and requirements:

Your final output and the code should be contained within Jupyter Notebook Previous Next

In [1]:
```
#IMPORT
import nltk as nltk
import nltk.corpus
from nltk.text import Text
from nltk.corpus import stopwords
import pandas as pd
import re
import sys
import matplotlib
import numpy as np
```

In [2]:
```
# import csv
fi = pd.read_csv('../rawdata/Food_Inspections.csv')
```

In [3]:
```
fi.head()
```

Out[3]:

	Inspection ID	DBA Name	AKA Name	License #	Facility Type	Risk	Addres

	Inspection ID	DBA Name	AKA Name	License #	Facility Type	Risk	Addres
0	2130049	JOE & THE JUICE ILLINOIS, LLC	JOE & THE JUICE	2564512.0	Restaurant	Risk 2 (Medium)	10 E DELAWAR PL
1	2130022	BOO BAE TEA INC	BOO BAE TEA INC	2570290.0	NaN	Risk 2 (Medium)	1013 W Webster AVE
2	2130018	FRESHII	FRESHII	2446395.0	Restaurant	Risk 1 (High)	1166 W MADISON ST
3	2129964	LINCOLN PARK PRESCHOOL	LINCOLN PARK PRESCHOOL & KINDERGARTEN	2215624.0	Daycare (2 - 6 Years)	Risk 1 (High)	108 W GERMANI PL
4	2129963	ORIGINAL STEAM	ORIGINAL STEAM	2574892.0	NaN	Risk 1 (High)	2428 S WALLACE AVE

In [4]:
```
# segment into own df
fail = fi[fi["Results"] == "Fail"].reset_index()
fail = fail.Violations
```

In [5]:
```
fail.head()
```

Out[5]:
```
0    9. WATER SOURCE: SAFE, HOT & COLD UNDER CITY P...
1    3. POTENTIALLY HAZARDOUS FOOD MEETS TEMPERATUR...
2    12. HAND WASHING FACILITIES: WITH SOAP AND SAN...
3    18. NO EVIDENCE OF RODENT OR INSECT OUTER OPEN...
4    41. PREMISES MAINTAINED FREE OF LITTER, UNNECE...
Name: Violations, dtype: object
```

In parsing the comments, I noticed there were line breaks introduced with pandas

We regex them out below

In [6]:
```
# Sample
fail[0]
```

Out[6]:
```
'9. WATER SOURCE: SAFE, HOT & COLD UNDER CITY PRESSURE - Comments: OBSE
RVE NO HOT RUNNING WATER ON THE PREMISES; THAT INCLUDES EXPOSED HAND SI
NKS IN FRONT & REAR PREP AREAS, 3-COMPARTMENT SINK IN REAR PREP, AND HA
ND SINKS IN BOTH TOILET ROOMS. INSTRUCTED TO CONTACT PLUMMER TO HAVE HO
T WATER RESTORED. CRITICAL VIOLATION 7-38-030. | 29. PREVIOUS MINOR VIO
LATION(S) CORRECTED 7-42-090 - Comments: PREVIOUS MINOR VIOLATION NOT C
ORRECTED FROM INSPECTION REPORT 1989320, DATED 2/17/2017. VIOLATION INC
LUDES; #38; NO RUNNING HOT AND COLD WATER TO TOP LOADING SOFT SERVE MAC
HINE, INSTRUCTED TO PROVIDE,  \n \nVIOLATION STILL EXISTS. SERIOUS VIOL
ATION 7-42-090 | 32. FOOD AND NON-FOOD CONTACT SURFACES PROPERLY DESIGN
ED, CONSTRUCTED AND MAINTAINED - Comments: MUST DISCONTINUE USING MILK
CRATES AS STORAGE RACKS THROUGHOUT FRONT AND REAR PREP AREAS, AND IN TH
E WALK IN COOLER. INSTALL CORRECT STORAGE RACKS. | 34. FLOORS: CONSTRUC
TED PER CODE, CLEANED, GOOD REPAIR, COVING INSTALLED, DUST-LESS CLEANIN
G METHODS USED - Comments: MUST REPAIR COVING ON WALL ACROSS FROM THE E
XPOSED HAND SINK  \nIN THE REAR PREP AREA.'
```

In [7]:
```
# replace line breaks
fail.replace({r'\n\s\n': ''}, regex=True,inplace=True)
```

In [8]:
```
# fixed.
fail[0]
```

Out[8]:
```
'9. WATER SOURCE: SAFE, HOT & COLD UNDER CITY PRESSURE - Comments: OBSE
RVE NO HOT RUNNING WATER ON THE PREMISES; THAT INCLUDES EXPOSED HAND SI
NKS IN FRONT & REAR PREP AREAS, 3-COMPARTMENT SINK IN REAR PREP, AND HA
ND SINKS IN BOTH TOILET ROOMS. INSTRUCTED TO CONTACT PLUMMER TO HAVE HO
T WATER RESTORED. CRITICAL VIOLATION 7-38-030. | 29. PREVIOUS MINOR VIO
LATION(S) CORRECTED 7-42-090 - Comments: PREVIOUS MINOR VIOLATION NOT C
ORRECTED FROM INSPECTION REPORT 1989320, DATED 2/17/2017. VIOLATION INC
LUDES; #38; NO RUNNING HOT AND COLD WATER TO TOP LOADING SOFT SERVE MAC
HINE, INSTRUCTED TO PROVIDE,  VIOLATION STILL EXISTS. SERIOUS VIOLATION
7-42-090 | 32. FOOD AND NON-FOOD CONTACT SURFACES PROPERLY DESIGNED, CO
NSTRUCTED AND MAINTAINED - Comments: MUST DISCONTINUE USING MILK CRATES
AS STORAGE RACKS THROUGHOUT FRONT AND REAR PREP AREAS, AND IN THE WALK
IN COOLER. INSTALL CORRECT STORAGE RACKS. | 34. FLOORS: CONSTRUCTED PER
CODE, CLEANED, GOOD REPAIR, COVING INSTALLED, DUST-LESS CLEANING METHOD
S USED - Comments: MUST REPAIR COVING ON WALL ACROSS FROM THE EXPOSED H
AND SINK  \nIN THE REAR PREP AREA.'
```

We've now imported the violations again and segmented out the failures

However we are looking for the individual comments, so we need to tweak the parsing.

In [9]:
```
# test pattern
reason_codes = pd.DataFrame(columns = ['reason'])
counter = 0

for i in range(0,len(fail)):
    test_text = str(fail[i])
    m = re.findall(r"Comments:\s(.*?)\s\|",test_text)
#     print("0",m[0])
#     print("1",m[1])
#     print("2",m[2])
    if len(m) > 0:
        for j in range(0,len(m)):
            reason_codes.loc[counter] = m[j]
            counter += 1
    else:
```

```
        reason_codes.loc[counter] = None
        counter += 1
```

In [10]:
```
reason_codes.head()
```

Out[10]:

	reason
0	OBSERVE NO HOT RUNNING WATER ON THE PREMISES; ...
1	PREVIOUS MINOR VIOLATION NOT CORRECTED FROM IN...
2	MUST DISCONTINUE USING MILK CRATES AS STORAGE ...
3	OBSERVED POTENTIALLY HAZARDOUS FOODS AT IMPROP...
4	INADEQUATE SUPPLY OF HOT WATER AVAILABLE IN TH...

In [11]:
```
# change to string
reason_str = reason_codes.to_string()
```

In [12]:
```
type(reason_str)
```

Out[12]:
```
str
```

In [13]:
```
# words
words = nltk.tokenize.word_tokenize(reason_str)
```

tokenization

In [14]:
```
#stopwords = stopwords.words('english')
stopwords = set(nltk.corpus.stopwords.words('english'))

# Remove single-character tokens (mostly punctuation)
words = [word for word in words if len(word) > 1]

# Remove punctuation
```

```
words = [word for word in words if word.isalpha()]

# Lowercase all words (default_stopwords are lowercase too)
words = [word.lower() for word in words]

# Remove stopwords
words = [word for word in words if word not in stopwords]
```

Now we can gt the most common key words that might indicate a failed response.

In [15]:
```
fdist = nltk.FreqDist(words)
fdist.most_common(50)
```

Out[15]:
```
[('must', 20890),
 ('observed', 18543),
 ('food', 17836),
 ('clean', 14585),
 ('shall', 9503),
 ('floors', 7437),
 ('repair', 6548),
 ('floor', 6281),
 ('prep', 6088),
 ('walls', 5955),
 ('found', 5937),
 ('nan', 5697),
 ('sink', 5664),
 ('provide', 5649),
 ('area', 5493),
 ('violation', 5030),
 ('instructed', 4796),
 ('rear', 4643),
 ('corrected', 4566),
 ('detail', 4523),
 ('replace', 4226),
 ('equipment', 4133),
 ('contact', 4051),
```

```
('light', 4000),
('storage', 3807),
('inside', 3630),
('water', 3523),
('control', 3516),
('door', 3479),
('remove', 3367),
('noted', 3327),
('used', 3202),
('stored', 3174),
('interior', 3173),
('hand', 3079),
('front', 2997),
('ice', 2976),
('cooler', 2963),
('ceiling', 2834),
('missing', 2770),
('exposed', 2722),
('wall', 2620),
('per', 2561),
('code', 2480),
('constructed', 2465),
('previous', 2451),
('along', 2443),
('ceilings', 2431),
('grease', 2394),
('surfaces', 2310)]
```

In [ ]:
```
size
```

Quick thoughts:

In a way these make sense. The words used all are the subjects of concern (door, area, hand, wall) or the actions upon those (corrected, contact, found, repair).

In [18]:
```
sys.getsizeof(reason_str)
```

Out[18]: 7289952

We can see how a word like "observed" might interact in a failed setting.

"Observed {some issue}"

Some cleaning / lemmatization!

In [20]:
```
wnl = nltk.WordNetLemmatizer()
print([wnl.lemmatize(t) for t in words[0:50]])
```

```
['reason', 'observe', 'hot', 'running', 'water', 'premise', 'previous',
'minor', 'violation', 'corrected', 'must', 'discontinue', 'using', 'mil
k', 'crate', 'storage', 'observed', 'potentially', 'hazardous', 'food',
'improp', 'inadequate', 'supply', 'hot', 'water', 'available', 'th', 'd
etail', 'clean', 'pizza', 'oven', 'prep', 'table', 'hot', 'detail', 'cl
ean', 'floor', 'throughout', 'premise', 'detail', 'clean', 'ventilatio
n', 'hood', 'filter', 'maintain', 'outside', 'area', 'remove', 'unneces
sa', 'replace']
```

N Gram application

In [44]:
```
#Create your bigrams or trigrams
bgs = nltk.bigrams(words)
tgs = nltk.trigrams(words)

#compute frequency distribution for all the bigrams in the text
fdist_2 = nltk.FreqDist(bgs)
fdist_3 = nltk.FreqDist(tgs)
```

Frequency

We'll first do bigrams and then trigrams most common.

They make a lot of sense!

In [34]:
```
fdist_2.most_common(50)
```

Out[34]:
```
[(('food', 'contact'), 3947),
 (('detail', 'clean'), 3878),
 (('must', 'clean'), 3699),
 (('must', 'provide'), 3502),
 (('violation', 'corrected'), 2647),
 (('floors', 'shall'), 2467),
 (('per', 'code'), 2455),
 (('constructed', 'per'), 2455),
 (('shall', 'constructed'), 2454),
 (('walls', 'ceilings'), 2359),
 (('prep', 'area'), 2333),
 (('good', 'repair'), 2202),
 (('ceilings', 'shall'), 2191),
 (('shall', 'good'), 2191),
 (('necessary', 'control'), 2168),
 (('control', 'measures'), 2164),
 (('measures', 'shall'), 2162),
 (('shall', 'used'), 2162),
 (('clean', 'floors'), 2088),
 (('must', 'repair'), 2083),
 (('contact', 'surfaces'), 2007),
 (('contact', 'equipment'), 1921),
 (('equipment', 'ut'), 1899),
 (('surfaces', 'equi'), 1785),
 (('previous', 'minor'), 1736),
 (('must', 'detail'), 1633),
 (('food', 'establishments'), 1478),
 (('clean', 'maintain'), 1468),
 (('hand', 'sink'), 1417),
 (('nan', 'nan'), 1409),
 (('exposed', 'hand'), 1398),
 (('must', 'remove'), 1377),
 (('clean', 'interior'), 1322),
 (('certified', 'food'), 1266),
 (('mice', 'droppings'), 1263),
```

```
 (('code', 'walls'), 1250),
 (('pest', 'control'), 1237),
 (('repair', 'replace'), 1207),
 (('city', 'chicago'), 1204),
 (('light', 'shields'), 1191),
 (('compartment', 'sink'), 1182),
 (('plumbing', 'fixtures'), 1153),
 (('ventilation', 'plumbing'), 1140),
 (('ceiling', 'tiles'), 1133),
 (('potentially', 'hazardous'), 1124),
 (('minor', 'violations'), 1102),
 (('light', 'shield'), 1066),
 (('establishments', 'display'), 1061),
 (('display', 'prepare'), 1061),
 (('corrected', 'violation'), 1032)]
```

In [35]:
```
fdist_3.most_common(50)
```

Out[35]:
```
[(('constructed', 'per', 'code'), 2454),
 (('shall', 'constructed', 'per'), 2454),
 (('floors', 'shall', 'constructed'), 2453),
 (('walls', 'ceilings', 'shall'), 2191),
 (('ceilings', 'shall', 'good'), 2191),
 (('shall', 'good', 'repair'), 2189),
 (('necessary', 'control', 'measures'), 2164),
 (('measures', 'shall', 'used'), 2162),
 (('control', 'measures', 'shall'), 2162),
 (('food', 'contact', 'surfaces'), 1983),
 (('food', 'contact', 'equipment'), 1920),
 (('contact', 'equipment', 'ut'), 1897),
 (('contact', 'surfaces', 'equi'), 1785),
 (('must', 'detail', 'clean'), 1588),
 (('per', 'code', 'walls'), 1250),
 (('code', 'walls', 'ceilings'), 1248),
 (('must', 'repair', 'replace'), 1139),
 (('ventilation', 'plumbing', 'fixtures'), 1133),
 (('previous', 'minor', 'violations'), 1078),
 (('food', 'establishments', 'display'), 1061),
 (('establishments', 'display', 'prepare'), 1061),
```

```
(('detail', 'clean', 'floors'), 1006),
(('surfaces', 'equi', 'floors'), 992),
(('equi', 'floors', 'shall'), 988),
(('corrected', 'violation', 'corrected'), 973),
(('violation', 'corrected', 'violation'), 966),
(('exposed', 'hand', 'sink'), 894),
(('equipment', 'ut', 'food'), 846),
(('potentially', 'hazardous', 'foods'), 816),
(('shall', 'used', 'food'), 793),
(('detail', 'clean', 'maintain'), 781),
(('ut', 'food', 'contact'), 761),
(('instructed', 'detail', 'clean'), 728),
(('used', 'food', 'contact'), 720),
(('parts', 'food', 'establishment'), 699),
(('food', 'establishment', 'pa'), 697),
(('observed', 'mice', 'droppings'), 686),
(('pest', 'control', 'log'), 604),
(('previous', 'minor', 'violation'), 597),
(('broken', 'glass', 'fall'), 582),
(('shielding', 'protect', 'broken'), 582),
(('protect', 'broken', 'glass'), 582),
(('food', 'service', 'manager'), 547),
(('clean', 'interior', 'exterior'), 538),
(('certified', 'food', 'manager'), 536),
(('control', 'log', 'book'), 531),
(('stained', 'ceiling', 'tiles'), 522),
(('hot', 'running', 'water'), 508),
(('city', 'chicago', 'certified'), 507),
(('food', 'stored', 'original'), 502)]
```

Longer n grams

In [43]:
```
sgs = nltk.ngrams(words,n=6)

#compute frequency distribution for all the bigrams in the text
fdist_6 = nltk.FreqDist(sgs)
```

In [45]:
```
fdist_6.most_common(10)
```

Out[45]:
```
[(('floors', 'shall', 'constructed', 'per', 'code', 'walls'), 1250),
 (('shall', 'constructed', 'per', 'code', 'walls', 'ceilings'), 1248),
 (('constructed', 'per', 'code', 'walls', 'ceilings', 'shall'), 1247),
 (('per', 'code', 'walls', 'ceilings', 'shall', 'good'), 1247),
 (('code', 'walls', 'ceilings', 'shall', 'good', 'repair'), 1246),
 (('contact', 'surfaces', 'equi', 'floors', 'shall', 'constructed'), 98
8),
 (('equi', 'floors', 'shall', 'constructed', 'per', 'code'), 988),
 (('surfaces', 'equi', 'floors', 'shall', 'constructed', 'per'), 988),
 (('food', 'contact', 'surfaces', 'equi', 'floors', 'shall'), 988),
 (('necessary', 'control', 'measures', 'shall', 'used', 'food'), 793)]
```

Final thoughts

We can see that a simple n grams model could serve us quite well if we wanted to predict failure or not given how much context we can derive simply from the frequencies.

In [ ]:
```
```