DoQA Cooking Training Data Cleaning

```
import json
import pandas as pd
from pathlib import Path
import re
import matplotlib.pyplot as plt
import itertools # mote up later
from nltk import bigrams, ngrams, trigrams
import collections # must move up
import nltk
from nltk.corpus import stopwords
```

```
import warnings
warnings.filterwarnings("ignore")
```

```
with open("/content/doqa-cooking-train-v2.1.json", 'r') as file:
    json_data = json.load(file)
```

```
data = json_data['data']
```

```
titles = []
backgrounds = []
all_paragraphs = []
for entry in data:
   title = entry["title"]
   background = entry["background"]
   paragraphs = [p["context"] for p in entry["paragraphs"]]
   # Store each variable in separate lists
   titles.append(title)
   backgrounds.append(background)
   all_paragraphs.append(paragraphs)

title_df = pd.DataFrame({'Title': titles})
background_df = pd.DataFrame({'Background': backgrounds})
paragraphs_df = pd.DataFrame({'Paragraphs': all_paragraphs})
```

```
print(title_df)
```

```
                                                 Title
0                           Tips for grilling duck legs?
1                           Tips for grilling duck legs?
2                           Tips for grilling duck legs?
3                  Meaning of do not thaw for frozen food
4                  Meaning of do not thaw for frozen food
...                                                  ...
1032  What is the difference between caramelized oni...
1033  What is the difference between caramelized oni...
1034        Is my heavy cream not actually heavy cream?
1035  How to save a dish with an onion paste base wh...
1036  How to save a dish with an onion paste base wh...

[1037 rows x 1 columns]
```

Begin Cleaning

```
title_df_list = title_df['Title'].tolist()
```

```
title_df_list[:3]
```

```
['Tips for grilling duck legs?',
 'Tips for grilling duck legs?',
 'Tips for grilling duck legs?']
```

```
def clean_string(text): # can be Title, Background, Paragraphs
    """re.sub(pattern, repl, string).
    Returns the string obtained by replacing the leftmost
    non-overlapping occurrences of pattern in string by the
    replacement thus removing any urls
    """
    return " ".join(re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", str(text)).split())
```

```
TextOnlyTitle = [clean_string(Title) for Title in title_df_list]#can be Title, Background, Paragraphs
```

```
TextOnlyTitle[:1] # Can be Title, Background, Paragraphs
```

```
['Tips for grilling duck legs']
```

```
ListlowercasewordsTitle = [Title.lower().split() for Title in TextOnlyTitle]
```

```
ListlowercasewordsTitle[:1]
```

```
[['tips', 'for', 'grilling', 'duck', 'legs']]
```

```
data = ListlowercasewordsTitle[:3]
for x in data:
    print(x, end=' ')
```

```
['tips', 'for', 'grilling', 'duck', 'legs'] ['tips', 'for', 'grilling', 'duck', 'legs'] ['tips', 'for', 'grillin
```

```
TextOnlyTitle = list(itertools.chain(*ListlowercasewordsTitle))
```

```
TextOnlyTitle[:2]
```

```
['tips', 'for']
```

```
len(TextOnlyTitle)
```

```
8873
```

```
UniqueWordsTitle = set(TextOnlyTitle)
```

```
len(UniqueWordsTitle)
```

```
1446
```

```
CountTextOnlyTitle = collections.Counter(TextOnlyTitle)
```

```
CountTextOnlyTitle.most_common(10)
```

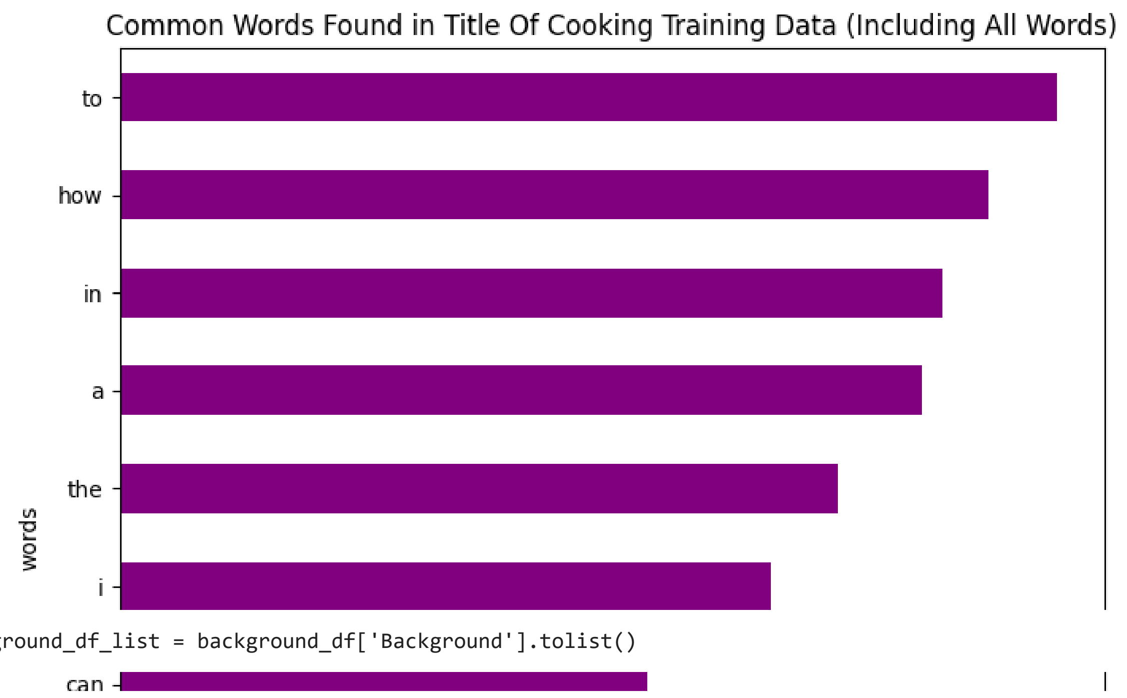
```
[('to', 286),
 ('how', 265),
 ('in', 251),
 ('a', 245),
 ('the', 219),
 ('i', 199),
 ('can', 161),
 ('what', 154),
 ('is', 150),
 ('of', 132)]
```

```
CleanTitle = pd.DataFrame(CountTextOnlyTitle.most_common(10),
                          columns=['words', 'count'])
```

```
CleanTitle
```

	words	count
0	to	286
1	how	265
2	in	251
3	a	245
4	the	219
5	i	199
6	can	161
7	what	154
8	is	150
9	of	132

```
fig, ax = plt.subplots(figsize=(8, 8))

# Plot horizontal bar graph
CleanTitle.sort_values(by='count').plot.barh(x='words',
                      y='count',
                      ax=ax,
                      color="purple")

ax.set_title("Common Words Found in Title Of Cooking Training Data (Including All Words)")

plt.show()
```



```
background_df_list = background_df['Background'].tolist()
```

```
background_df_list[:3]
```

```
    ["I recently attempted to grill duck legs on my propane Webber. I was afraid of flare-ups due to the high fat
    content in the duck meat so I grilled with somewhat low and indirect heat. It took a long time, but I got them
    looking lovely and brown and not burned. The only problem was this: they were tough and didn't taste very good
    at all. Clearly I did something very wrong. Any advice?",
     "I recently attempted to grill duck legs on my propane Webber. I was afraid of flare-ups due to the high fat
    content in the duck meat so I grilled with somewhat low and indirect heat. It took a long time, but I got them
    looking lovely and brown and not burned. The only problem was this: they were tough and didn't taste very good
    at all. Clearly I did something very wrong. Any advice?",
     "I recently attempted to grill duck legs on my propane Webber. I was afraid of flare-ups due to the high fat
    content in the duck meat so I grilled with somewhat low and indirect heat. It took a long time, but I got them
    looking lovely and brown and not burned. The only problem was this: they were tough and didn't taste very good
    at all. Clearly I did something very wrong. Any advice?"]
```

```
def clean_string(text): # can be Title, Background, Paragraphs
    """re.sub(pattern, repl, string).
    Returns the string obtained by replacing the leftmost
    non-overlapping occurrences of pattern in string by the
    replacement thus removing any urls
    """
    return " ".join(re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", str(text)).split())
```

```
TextOnlyBackground = [clean_string(Background) for Background in background_df_list]#can be Title, Background, Parag
```

```
TextOnlyBackground[:1] # Can be Title, Background, Paragraphs
```

```
    ['I recently attempted to grill duck legs on my propane Webber I was afraid of flareups due to the high fat
    content in the duck meat so I grilled with somewhat low and indirect heat It took a long time but I got them
    looking lovely and brown and not burned The only problem was this they were tough and didnt taste very good at
    all Clearly I did something very wrong Any advice']
```

```
ListlowercasewordsBackground = [Background.lower().split() for Background in TextOnlyBackground]
```

```
data = ListlowercasewordsBackground[:3]
for x in data:
    print(x, end=' ')
```

```
['i', 'recently', 'attempted', 'to', 'grill', 'duck', 'legs', 'on', 'my', 'propane', 'webber', 'i', 'was', 'afra
```

```
TextOnlyBackground = list(itertools.chain(*ListlowercasewordsBackground))
```

```
TextOnlyBackground[:2]
```

```
['i', 'recently']
```

```
len(TextOnlyBackground)
```

```
89776
```

```
UniqueWordsBackground = set(TextOnlyBackground)
```

```
len(UniqueWordsBackground)
```

```
6123
```

```
CountTextOnlyBackground = collections.Counter(TextOnlyBackground)
```

```
CountTextOnlyBackground.most_common(25)
```

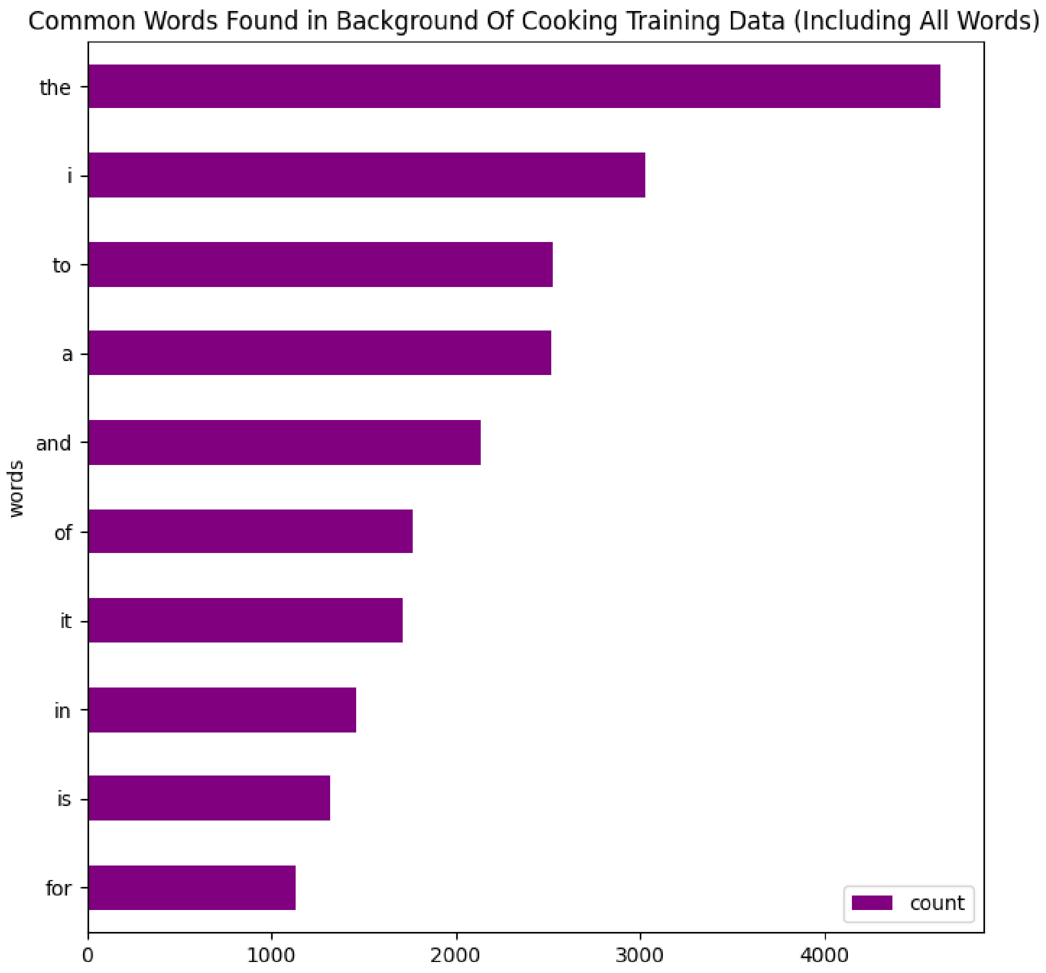
```
[('the', 4627),
 ('i', 3027),
 ('to', 2527),
 ('a', 2518),
 ('and', 2133),
 ('of', 1768),
 ('it', 1710),
 ('in', 1458),
 ('is', 1320),
 ('for', 1136),
 ('that', 961),
 ('with', 771),
 ('but', 714),
 ('this', 679),
 ('have', 646),
 ('or', 620),
 ('be', 618),
 ('my', 547),
 ('on', 517),
 ('are', 457),
 ('not', 443),
 ('was', 427),
 ('as', 424),
 ('if', 405),
 ('what', 397)]
```

```
CleanBackground = pd.DataFrame(CountTextOnlyBackground.most_common(10),
                        columns=['words', 'count'])
```

```
CleanBackground
```

	words	count
0	the	4627
1	i	3027
2	to	2527
3	a	2518
4	and	2133
5	of	1768

```
fig, ax = plt.subplots(figsize=(8, 8))

# Plot horizontal bar graph
CleanBackground.sort_values(by='count').plot.barh(x='words',
                      y='count',
                      ax=ax,
                      color="purple")

ax.set_title("Common Words Found in Background Of Cooking Training Data (Including All Words)")

plt.show()
```



```
#importing stop word dictionary
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
#Defining The Stop Words
stop_words = set(stopwords.words('english')) #there are 179 stop words
```

```
# View a few words from the set
list(stop_words)[0:5]
```

```
["didn't", 'shouldn', "that'll", 'if', 'these']
```

```
ListlowercasewordsBackground[0] #list each lower case tweet
```

```
 'due',
 'to',
 'the',
 'high',
 'fat',
 'content',
 'in',
 'the',
 'duck',
 'meat',
 'so',
 'i',
 'grilled',
 'with',
 'somewhat',
 'low',
 'and',
 'indirect',
 'heat',
 'it',
 'took',
 'a',
 'long',
 'time',
 'but',
 'i',
 'got',
 'them',
 'looking',
 'lovely',
 'and',
 'brown',
 'and',
 'not',
 'burned',
 'the',
 'only',
  'something',
 'very',
 'wrong',
 'any',
 'advice']
```

```
BackgroundWithoutStopwords = [[word for word in TextOnlyBackground if not word in stop_words] #works
              for TextOnlyBackground in ListlowercasewordsBackground]
```

```
BackgroundWithoutStopwords[0]
```

```
    ['recently',
     'attempted',
     'grill',
     'duck',
     'legs',
     'propane',
     'webber',
     'afraid',
     'flareups',
     'due',
     'high',
     'fat',
     'content',
     'duck',
     'meat',
     'grilled',
     'somewhat',
     'low',
     'indirect',
     'heat',
     'took',
     'long',
     'time',
     'got',
     'looking',
     'lovely',
     'brown',
     'burned',
     'problem',
     'tough',
     'didnt',
     'taste',
     'good',
     'clearly',
     'something',
     'wrong',
     'advice']
```

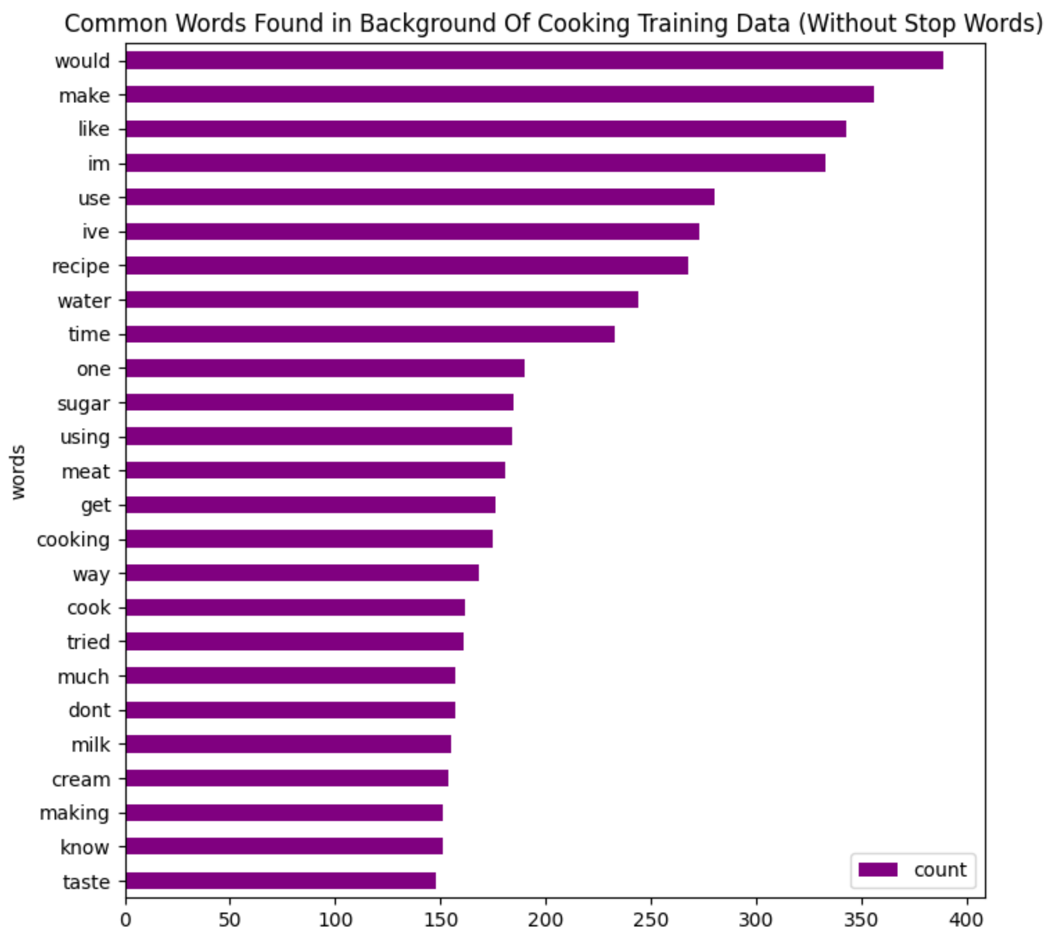
```
BackgroundWithoutStopword = list(itertools.chain(*BackgroundWithoutStopwords))
```

```
CountBackgroundsWithoutStopwords = collections.Counter(BackgroundWithoutStopword)
```

```
CountBackgroundsWithoutStopwords.most_common(10)
```

```
    [('would', 389),
     ('make', 356),
     ('like', 343),
     ('im', 333),
     ('use', 280),
     ('ive', 273),
     ('recipe', 268),
     ('water', 244),
     ('time', 233),
     ('one', 190)]
```

```
BackgroundWithoutStopwords = pd.DataFrame(CountBackgroundsWithoutStopwords.most_common(25)
                               columns=['words', 'count'])

fig, ax = plt.subplots(figsize=(8, 8))

# Plot horizontal bar graph
BackgroundWithoutStopwords.sort_values(by='count').plot.barh(x='words',
                      y='count',
                      ax=ax,
                      color="purple")

ax.set_title("Common Words Found in Background Of Cooking Training Data (Without Stop Words)")

plt.show()
```



```
# Create list of lists containing bigrams in tweets
Backgroundbigram = [list(bigrams(Background)) for Background in BackgroundWithoutStopwords]


# View bigrams for the first tweet
Backgroundbigram[:]
```

```
[[('w', 'o'), ('o', 'r'), ('r', 'd'), ('d', 's')],
 [('c', 'o'), ('o', 'u'), ('u', 'n'), ('n', 't')]]
```

Paragraph

```
#background_df = pd.DataFrame({'Background': backgrounds})
```

```
paragraphs_df_list = paragraphs_df['Paragraphs'].tolist()
```

```
paragraphs_df_list[:3]
```

```
[['I think grilling is probably a bad plan for duck legs; the fat content is a real danger like you said, and
duck legs are tough enough you probably want to confit them or braise them.If you absolutely have to grill
them, I would suggest confiting them at 200 degrees for three or four hours first (you could use veggie oil in
a pinch) and then resting them in the fridge for a day or so in oil. As for finishing them on the grill, rinse
them off gently, re-season if needed, cook flesh side down on a medium heat portion of the grill for a while
until mostly heated through, then flip them over on a high heat portion of the grill to crisp up the skin,
watching out for flares. CANNOTANSWER'],
 ['I think grilling is probably a bad plan for duck legs; the fat content is a real danger like you said, and
duck legs are tough enough you probably want to confit them or braise them.If you absolutely have to grill
them, I would suggest confiting them at 200 degrees for three or four hours first (you could use veggie oil in
a pinch) and then resting them in the fridge for a day or so in oil. As for finishing them on the grill, rinse
them off gently, re-season if needed, cook flesh side down on a medium heat portion of the grill for a while
until mostly heated through, then flip them over on a high heat portion of the grill to crisp up the skin,
watching out for flares. CANNOTANSWER'],
 ['I think grilling is probably a bad plan for duck legs; the fat content is a real danger like you said, and
duck legs are tough enough you probably want to confit them or braise them.If you absolutely have to grill
them, I would suggest confiting them at 200 degrees for three or four hours first (you could use veggie oil in
a pinch) and then resting them in the fridge for a day or so in oil. As for finishing them on the grill, rinse
them off gently, re-season if needed, cook flesh side down on a medium heat portion of the grill for a while
until mostly heated through, then flip them over on a high heat portion of the grill to crisp up the skin,
watching out for flares. CANNOTANSWER']]
```

```
TextOnlyParagraphs = [clean_string(paragraphs) for paragraphs in paragraphs_df_list]#can be Title, Background, Parag
```

```
TextOnlyParagraphs[:1] # Can be Title, Background, Paragraphs
```

```
['I think grilling is probably a bad plan for duck legs the fat content is a real danger like you said and duck
legs are tough enough you probably want to confit them or braise themIf you absolutely have to grill them I
would suggest confiting them at 200 degrees for three or four hours first you could use veggie oil in a pinch
and then resting them in the fridge for a day or so in oil As for finishing them on the grill rinse them off
gently reseason if needed cook flesh side down on a medium heat portion of the grill for a while until mostly
heated through then flip them over on a high heat portion of the grill to crisp up the skin watching out for
flares CANNOTANSWER']
```

```
ListlowercasewordsParagraphs= [Paragraphs.lower().split() for Paragraphs in TextOnlyParagraphs]
```

```
data = ListlowercasewordsParagraphs[:3]
for x in data:
    print(x, end=' ')
```

```
['i', 'think', 'grilling', 'is', 'probably', 'a', 'bad', 'plan', 'for', 'duck', 'legs', 'the', 'fat', 'content',
```

```
TextOnlyParagraphs = list(itertools.chain(*ListlowercasewordsParagraphs))
```

```
TextOnlyParagraphs[:2]
```

```
['i', 'think']
```

```
len(TextOnlyParagraphs)
```

```
     120730
```

```
UniqueWordsParagraphs = set(TextOnlyParagraphs)
```

```
len(UniqueWordsParagraphs) #15816/101330=15.6% Unique%=UniqueWords/TextOnlyTweet
```

```
     6974
```

```
ParagraphsWithoutStopwords = [[word for word in TextOnlyParagraphs if not word in stop_words] #works
              for TextOnlyParagraphs in ListlowercasewordsParagraphs]
```

```
ParagraphsWithoutStopwords[0]
```

```
      'real',
      'danger',
      'like',
      'said',
      'duck',
      'legs',
      'tough',
      'enough',
      'probably',
      'want',
      'confit',
      'braise',
      'themif',
      'absolutely',
      'grill',
      'would',
      'suggest',
      'confiting',
      '200',
      'degrees',
      'three',
      'four',
      'hours',
      'first',
      'could',
      'use',
      'veggie',
      'oil',
      'pinch',
      'resting',
      'fridge',
      'day',
      'oil',
      'finishing',
      'grill',
      'rinse',
      'gently',
      'reseason',
      'needed',
      'cook',
      'flesh',
      'side',
      'medium',
      'heat',
      'skin',
      'watching',
      'flares',
      'cannotanswer']
```

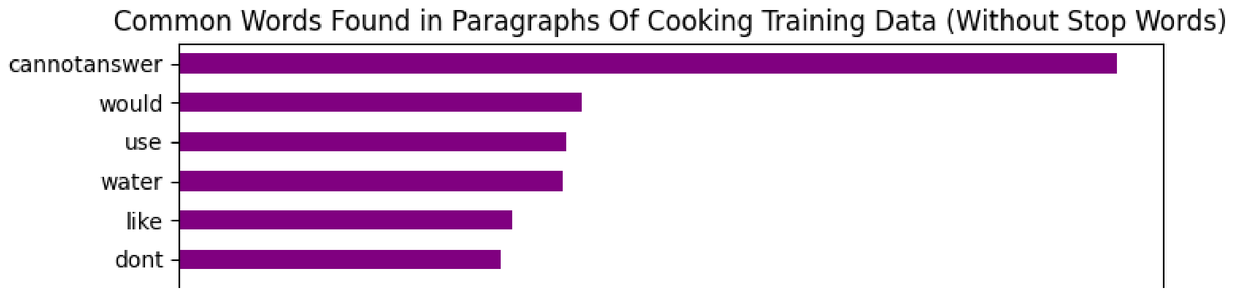
```
ParagraphsWithoutStopword = list(itertools.chain(*ParagraphsWithoutStopwords))
```

```
CountParagraphsWithoutStopwords = collections.Counter(ParagraphsWithoutStopword)
```

```
CountParagraphsWithoutStopwords.most_common(10)
```

```
     [('cannotanswer', 1037),
      ('would', 446),
      ('use', 429),
      ('water', 424),
      ('like', 368),
      ('dont', 356),
      ('make', 330),
      ('also', 324),
      ('get', 320),
      ('much', 289)]
```

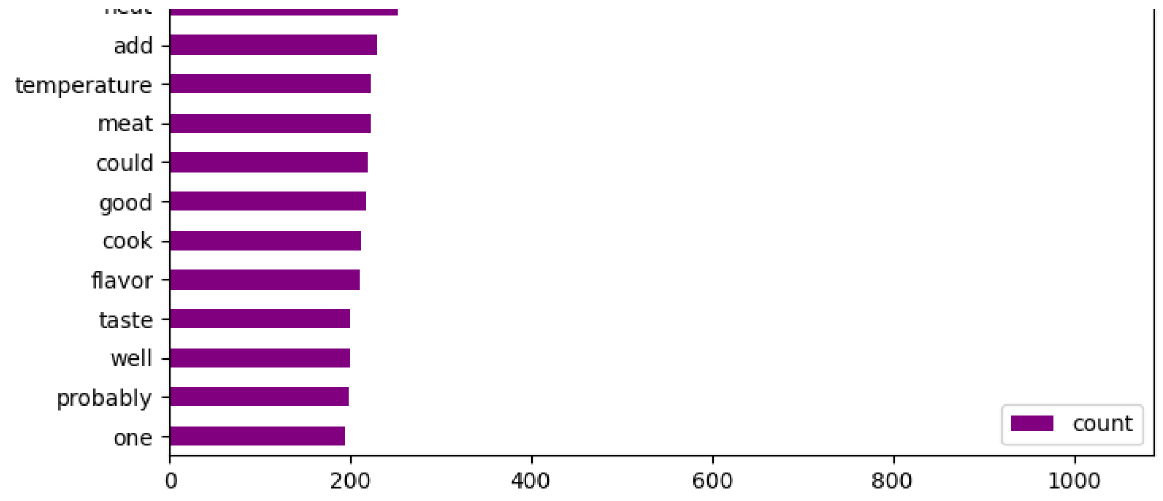
```
ParagraphsWithoutStopwords = pd.DataFrame(CountParagraphsWithoutStopwords.most_common(25),
                             columns=['words', 'count'])

fig, ax = plt.subplots(figsize=(8, 8))

# Plot horizontal bar graph
ParagraphsWithoutStopwords.sort_values(by='count').plot.barh(x='words',
                      y='count',
                      ax=ax,
                      color="purple")

ax.set_title("Common Words Found in Paragraphs Of Cooking Training Data (Without Stop Words)")

plt.show()
```



```
# Create list of lists containing bigrams in tweets
Paragraphsbigram = [list(bigrams(Paragraphs)) for Paragraphs in ParagraphsWithoutStopwords]
```

```
# View bigrams for the first tweet
Paragraphsbigram[:]
```

```
    [[('w', 'o'), ('o', 'r'), ('r', 'd'), ('d', 's')],
     [('c', 'o'), ('o', 'u'), ('u', 'n'), ('n', 't')]]
```


add
temperature
meat
could
good
cook
flavor
taste
well
probably
one
count