In [1]:

```
import pandas as pd
import numpy as np
import string
import re
from collections import Counter
from nltk.corpus import stopwords

pt1 = pd.read_csv('Shakespeare_works2.csv')

pt1.head()
```

Out[1]:

	Title	Publish Date	ParagraphNum	PlainText
0	All's Well That Ends Well	1602	1.0	Enter BERTRAM, the COUNTESS of Rousillon, HELE...
1	All's Well That Ends Well	1602	3.0	In delivering my son from me, I bury a second ...
2	All's Well That Ends Well	1602	4.0	And I in going, madam, weep o'er my father's d...
3	All's Well That Ends Well	1602	7.0	You shall find of the king a husband, madam; y...
4	All's Well That Ends Well	1602	12.0	What hope is there of his majesty's amendment?\n

In [2]:

```
pt1 = pt1[pt1.notnull()]
```

In [3]:

```
len(pt1.Title.unique())
```

Out[3]:

```
53
```

In [4]:

```
pt1.isnull().sum().sort_values(ascending = False)
```

Out[4]:

```
ParagraphNum    9
PlainText       9
Publish Date    6
Title           1
dtype: int64
```

In [5]:

```
pt1 = pt1.dropna()
```

In [6]:

```
pt1['Publish Date'] = pt1['Publish Date'].astype(int)
```

In [7]:

```
pt1.Title.unique()
```

Out[7]:

```
array(["All's Well That Ends Well", 'Antony and Cleopat
ra',
       'As You Like It', 'Comedy of Errors', 'Coriolanu
s', 'Cymbeline',
       'Hamlet', 'Henry IV, Part I', 'Henry IV, Part I
I', 'Henry V',
       'Henry VI, Part I', 'Henry VI, Part II', 'Henry
VI, Part III',
       'Henry VIII', 'Julius Caesar', 'King John', 'Kin
g Lear',
       "Lover's Complaint", "Love's Labour's Lost", 'Ma
cbeth',
       'Measure for Measure', 'Merchant of Venice',
       'Merry Wives of Windsor', "Midsummer Night's Dre
am",
       'Much Ado about Nothing', 'Othello', 'Passionate
Pilgrim',
       'Pericles', 'Phoenix and the Turtle', 'Rape of L
ucrece',
       'Richard II', 'Richard III', 'Romeo and Juliet',
'Sonnets',
       'Taming of the Shrew', '\nTaming of the Shrew"',
'Tempest',
       'Timon of Athens', 'Titus Andronicus', 'Troilus
and Cressida',
       'Twelfth Night', 'Two Gentlemen of Verona', 'Ven
us and Adonis',
       "Winter's Tale"], dtype=object)
```

In [8]:

```
print(pt1['Publish Date'].min())
pt1['Publish Date'].max()
```

```
1589
```

Out[8]:

```
1612
```

In [9]:

```
pt1['Publish Date'].value_counts().sort_index()
```

Out[9]:

```
1589     664
1590    1870
1591     787
1592    1224
1593    1829
1594    3324
1595    1241
1596    1343
1597    1871
1598    1958
1599    2798
1600    1430
1601    1320
1602    1034
1604    2296
1605    1946
1606    1361
1607    2110
1608     756
1609    1180
1610     814
1611     702
1612     788
Name: Publish Date, dtype: int64
```

In [10]:

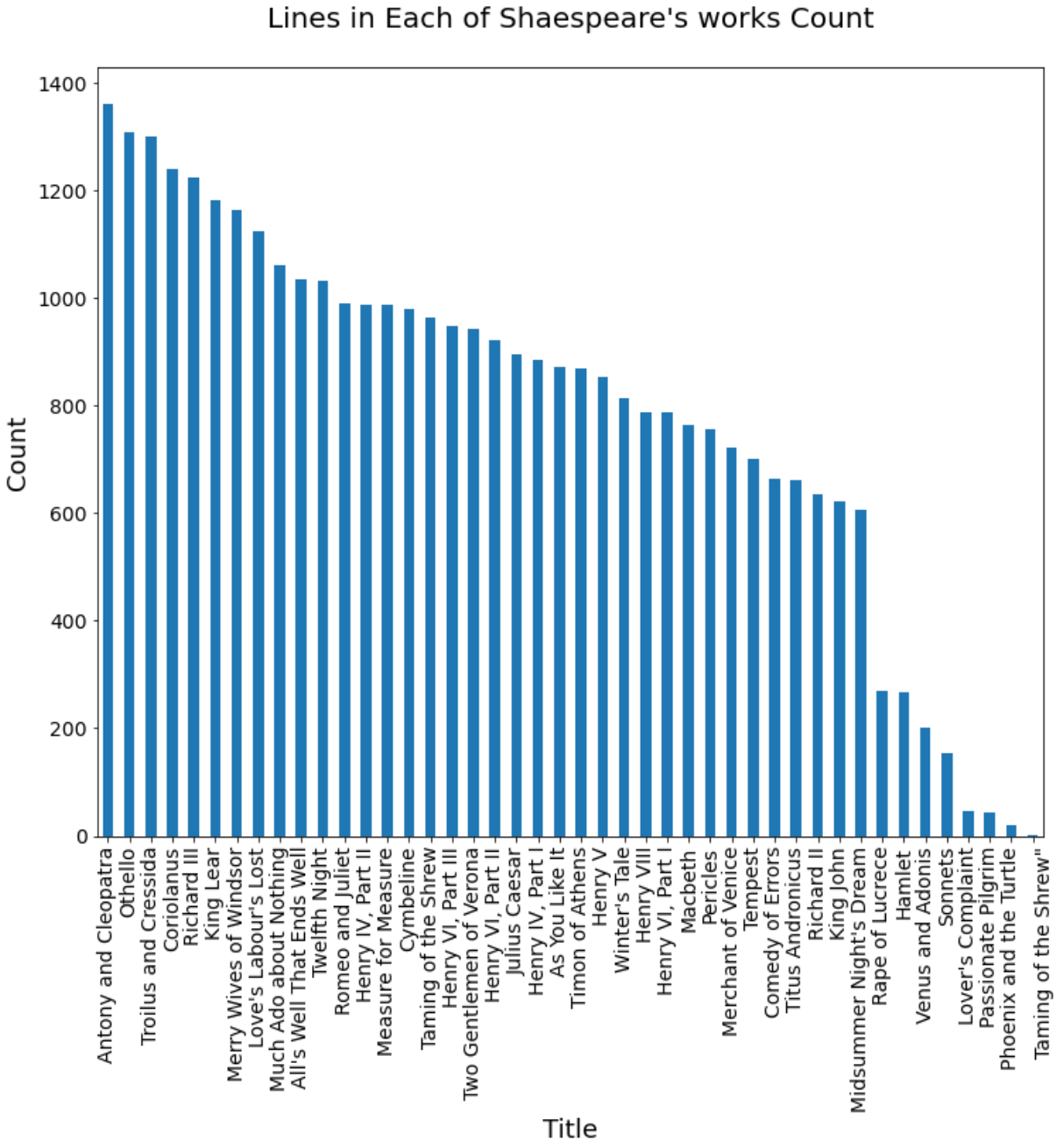
```
def clean_text(pt1):
    clean1 = re.sub(r'['+string.punctuation + '’—”'+']', "", pt1.lower
    return re.sub(r'\W+', ' ', clean1)
```

In [11]:

```
pt1['tokenized'] = pt1['PlainText'].map(lambda x: clean_text(x))
```

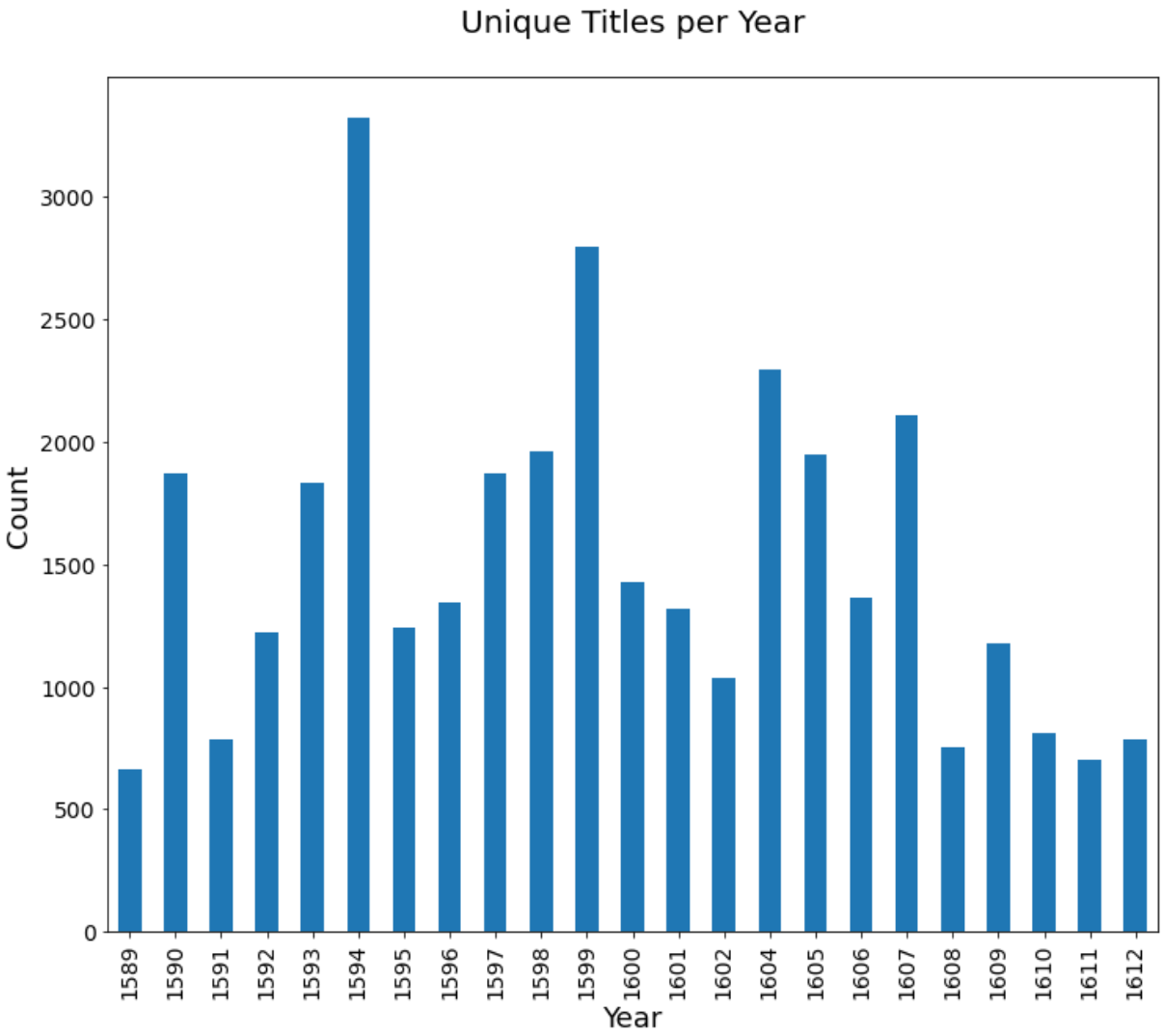
In [12]:

```
ax = pt1['Title'].value_counts(ascending = False).plot(kind='bar', for
ax.set_title("Lines in Each of Shaespeare's works Count\n", fontsize=2
ax.set_xlabel('Title', fontsize=18)
ax.set_ylabel('Count', fontsize=18);
```



In [44]:

```
# Chart unique titles per year and sort by year
ax = pt1['Publish Date'].value_counts().sort_index().plot(kind='bar',
ax.set_title("Unique Titles per Year\n", fontsize=20)
ax.set_xlabel('Year', fontsize=18)
ax.set_ylabel('Count', fontsize=18);
```

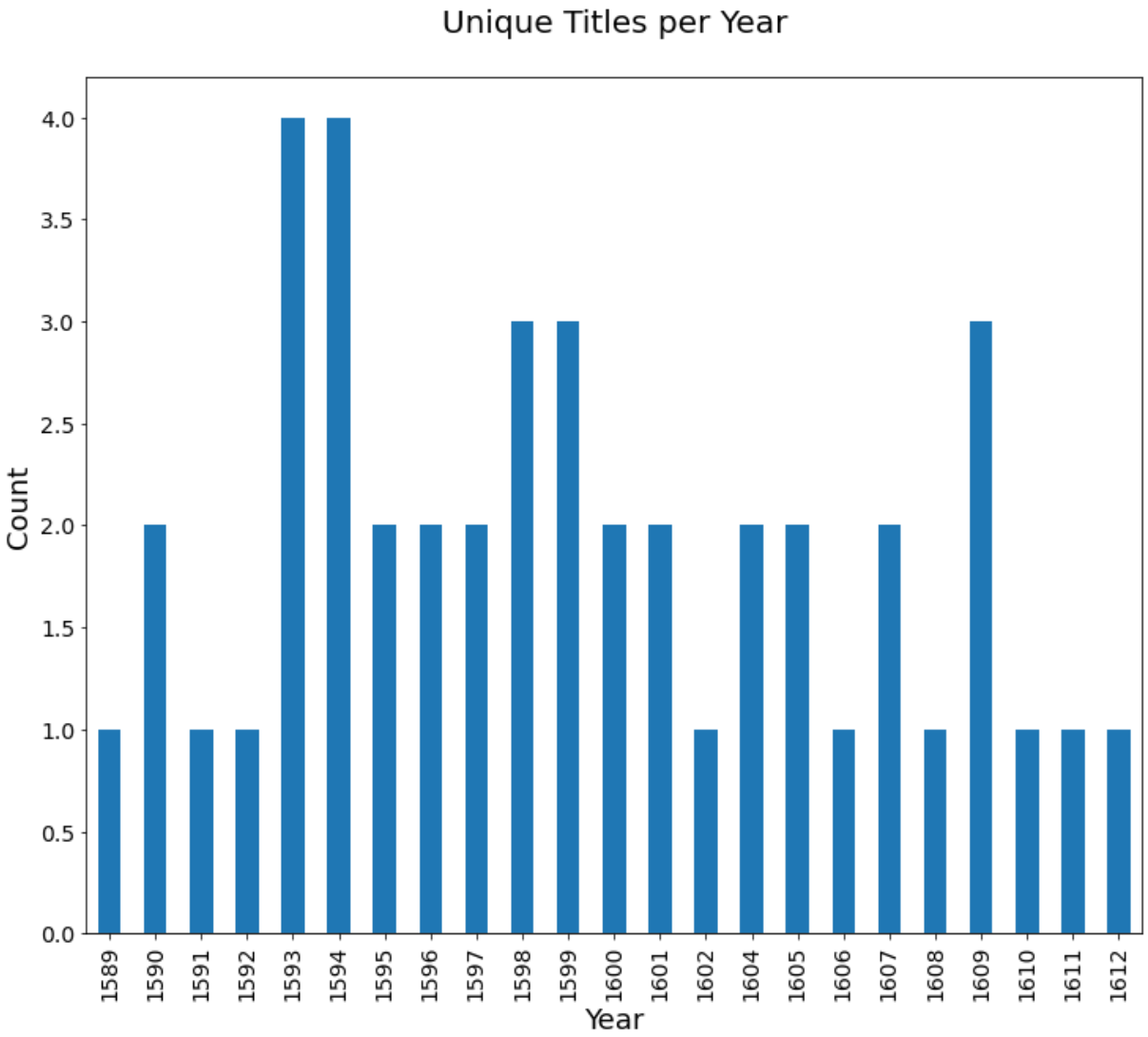


In [14]:

```
#new data frame with only unique titles and publish date
pt2 = pt1[['Title', 'Publish Date']].drop_duplicates()
```

In [45]:

```
# Chart unique titles per year and sort by year
ax = pt2['Publish Date'].value_counts().sort_index().plot(kind='bar',
ax.set_title("Unique Titles per Year\n", fontsize=20)
ax.set_xlabel('Year', fontsize=18)
ax.set_ylabel('Count', fontsize=18);
```



In [16]:

```
pt1['tokenized'].head()
```

Out[16]:

```
0    enter bertram the countess of rousillon helena...
1    in delivering my son from me i bury a second h...
2    and i in going madam weep oer my fathers death...
3    you shall find of the king a husband madam you...
4        what hope is there of his majestys amendment
Name: tokenized, dtype: object
```

In [17]:

```
pt1['num_wds'] = pt1['tokenized'].apply(lambda x: len(x.split()))
pt1['num_wds'].mean()
```

Out[17]:

```
24.822519194134966
```

In [18]:

```
print(pt1['num_wds'].max())
pt1['num_wds'].min()
```

```
588
```

Out[18]:

```
1
```

In [19]:

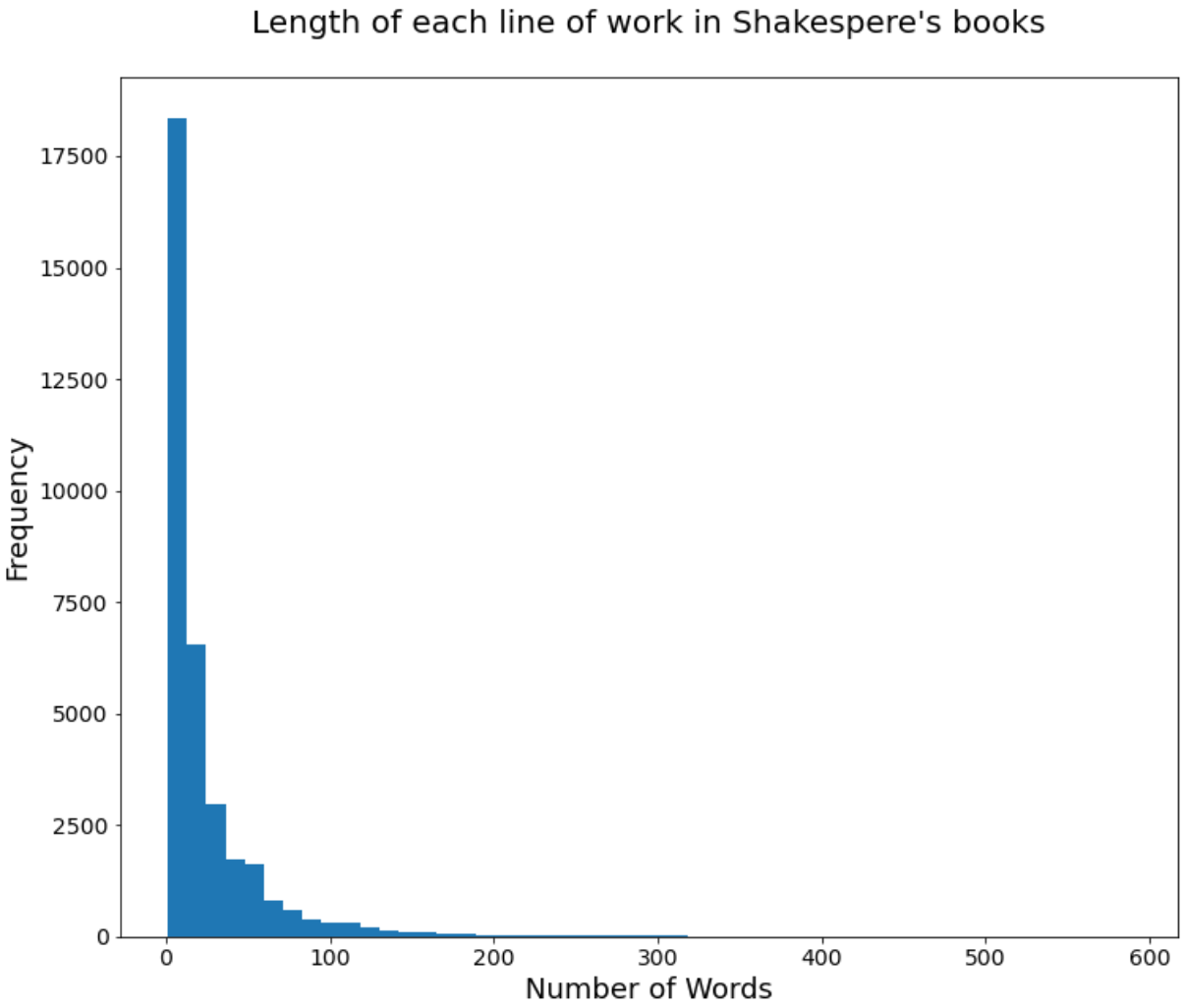
```
len(pt1[pt1['num_wds']==0])
```

Out[19]:

```
0
```

In [20]:

```
ax=pt1['num_wds'].plot(kind='hist', bins=50, fontsize=14, figsize=(12,
ax.set_title("Length of each line of work in Shakespere's books\n", fo
ax.set_ylabel('Frequency', fontsize=18)
ax.set_xlabel('Number of Words', fontsize=18);
```



In [21]:

```
pt1['uniq_wds'] = pt1['tokenized'].str.split().apply(lambda x: len(set
pt1['uniq_wds'].head()
```

Out[21]:

```
0    12
1    11
2    24
3    40
4     8
Name: uniq_wds, dtype: int64
```

In [22]:

```
print(pt1['uniq_wds'].mean())
print(pt1['uniq_wds'].min())
pt1['uniq_wds'].max()
```

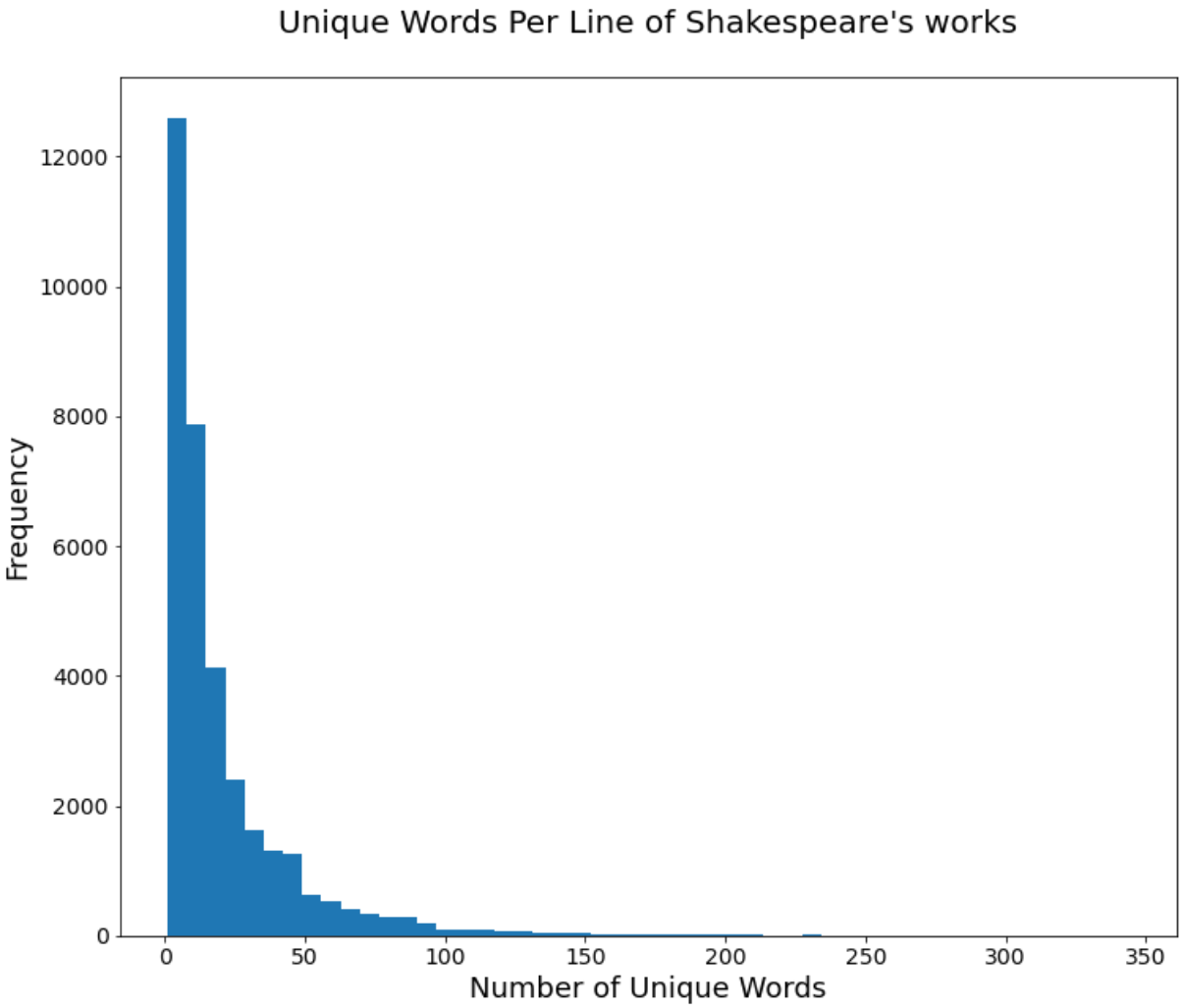
```
20.627980142007736
1
```

Out[22]:

```
344
```

In [23]:

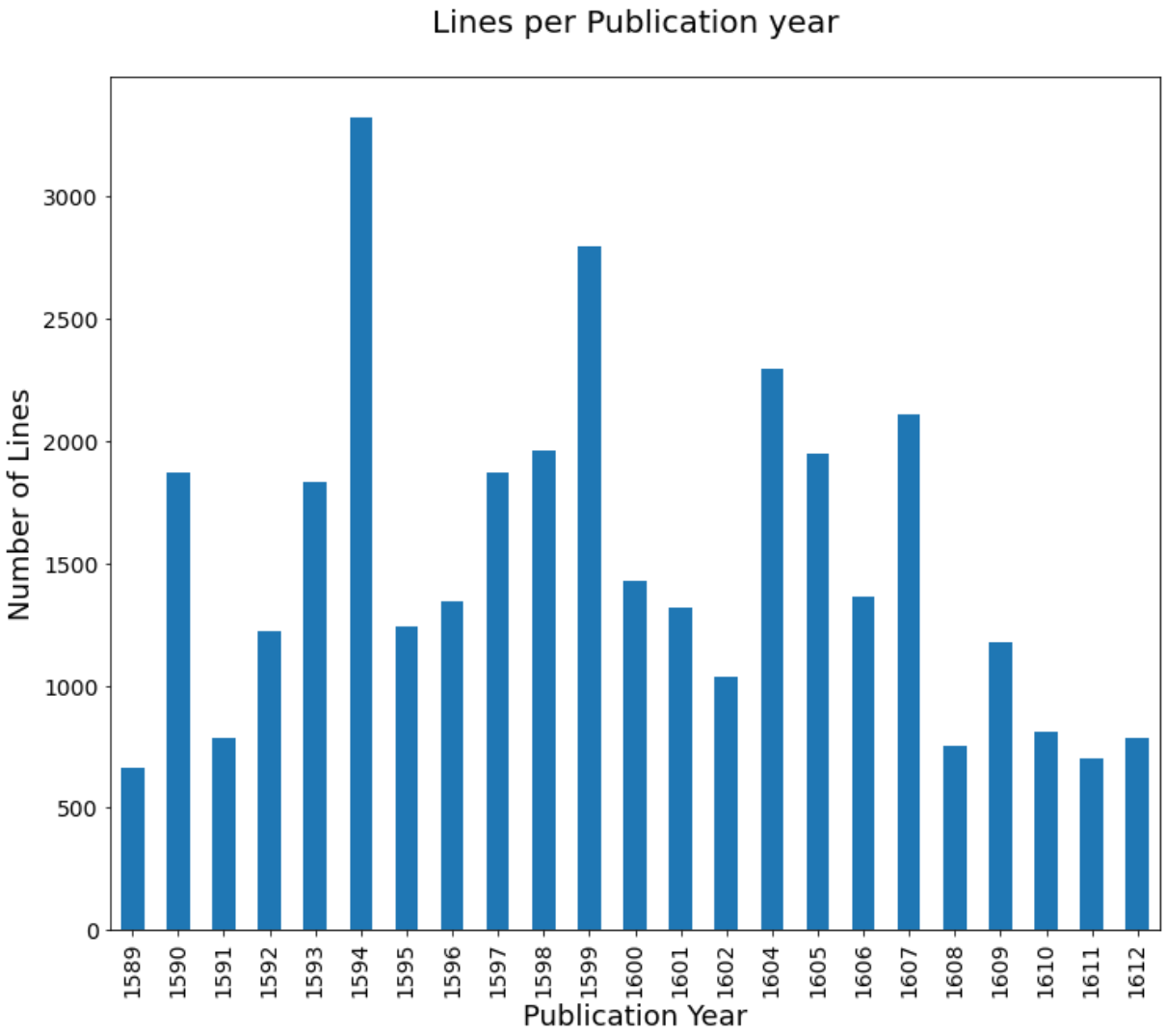
```
ax=pt1['uniq_wds'].plot(kind='hist', bins=50, fontsize=14, figsize=(12
ax.set_title("Unique Words Per Line of Shakespeare's works\n", fontsiz
ax.set_ylabel('Frequency', fontsize=18)
ax.set_xlabel('Number of Unique Words', fontsize=18);
```



In [24]:

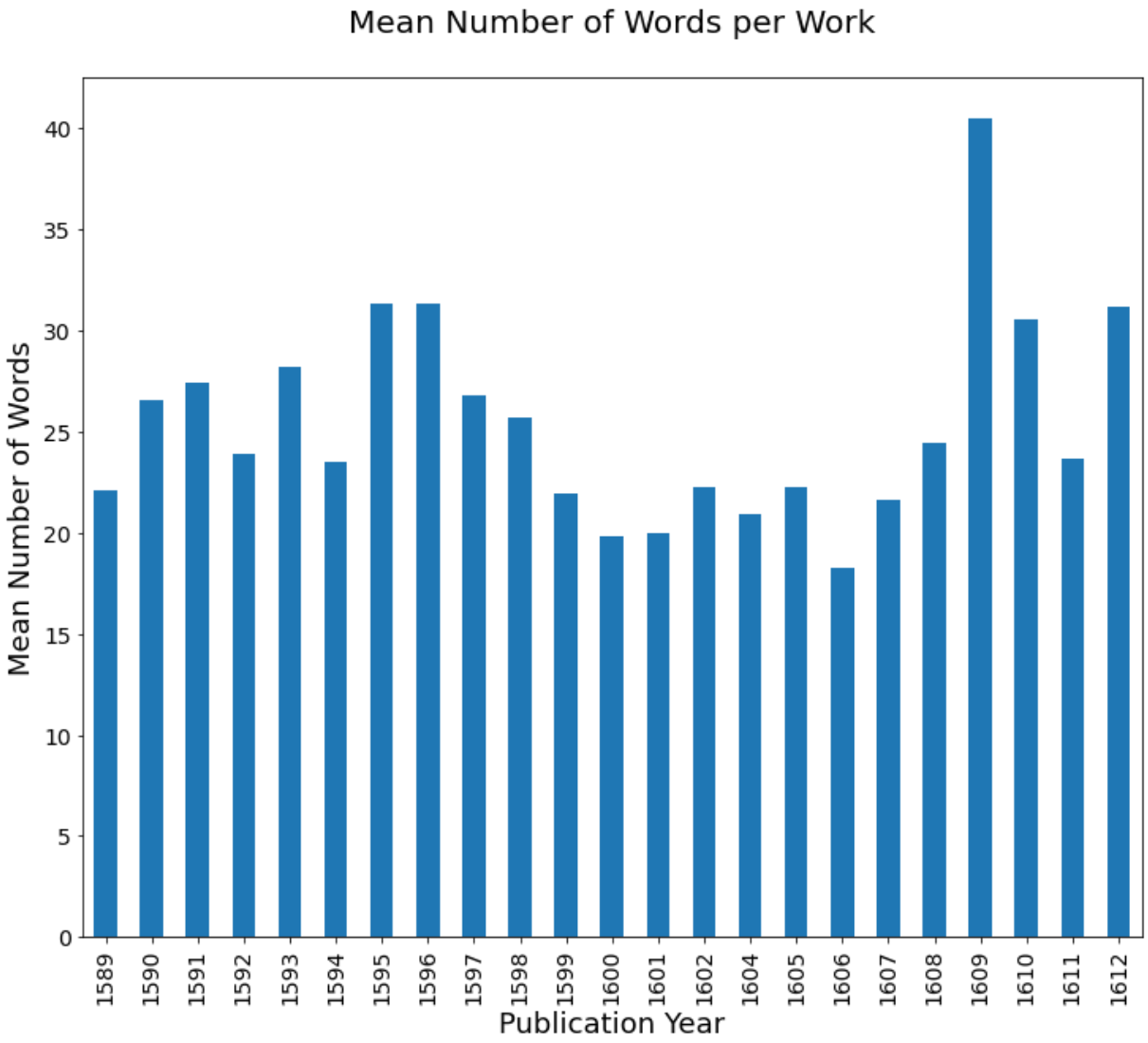
```
art_grps = pt1.groupby('Publish Date')

ax=art_grps['Publish Date'].aggregate(len).plot(kind='bar', fontsize=1
ax.set_title('Lines per Publication year\n', fontsize=20)
ax.set_ylabel('Number of Lines', fontsize=18)
ax.set_xlabel('Publication Year', fontsize=18);
```



In [25]:

```
ax=art_grps['num_wds'].aggregate(np.mean).plot(kind='bar', fontsize=14
ax.set_title('Mean Number of Words per Work\n', fontsize=20)
ax.set_ylabel('Mean Number of Words', fontsize=18)
ax.set_xlabel('Publication Year', fontsize=18);
```



In [26]:

```
wd_counts = Counter()
for i, row in pt1.iterrows():
    wd_counts.update(row['tokenized'].split())
```

In [27]:

```
wd_counts
```

Out[27]:

```
Counter({'enter': 1725,
         'bertram': 28,
         'the': 24898,
         'countess': 13,
         'of': 15589,
         'rousillon': 13,
         'helena': 44,
         'pand': 7072,
         'lafeu': 18,
         'all': 3663,
         'in': 10185,
         'black': 158,
         'delivering': 3,
         'my': 11639,
         'son': 599,
         'from': 2423,
         'me': 7734,
         'i': 17988,
```

In [28]:

```
for sw in stopwords.words('english'):
    del wd_counts[sw]
```

In [29]:

```
wd_counts.most_common(20)
```

Out[29]:

```
[('pand', 7072),
 ('thou', 5129),
 ('thy', 3870),
 ('thee', 3284),
 ('shall', 3220),
 ('pthe', 3182),
 ('pto', 3177),
 ('pthat', 2689),
 ('good', 2629),
 ('pi', 2608),
 ('lord', 2538),
 ('sir', 2442),
 ('well', 2315),
 ('come', 2186),
 ('would', 2123),
 ('love', 2057),
 ('pbut', 1894),
 ('man', 1746),
 ('enter', 1725),
 ('let', 1716)]
```

In [30]:

```
# Recognize and count dimensia words: sad, remember, forgot, old,this
def dementia_count(pt1):
    dementia = ['sad', 'remember', 'forgot', 'old', 'this world is not
                'forgetful', 'forgetfulness', 'forgetting', 'forgettir
                'forgetting appointments', 'forgetting where you are',
                'Alzheimer', 'Alzheimer\'s', 'Alzheimer\'s disease', '
                'Alzheimer\'', 'mental deterioration', 'mental decline
                'mental loss', 'softening of the brain', 'softening of
    dementia_count = 0
    for word in dementia:
        if word in pt1:
            dementia_count += 1
    return dementia_count
```

In [31]:

```
pt1['num_dim_words'] = pt1.tokenized.apply(dementia_count)
```

In [32]:

```
pt1
```

Out[32]:

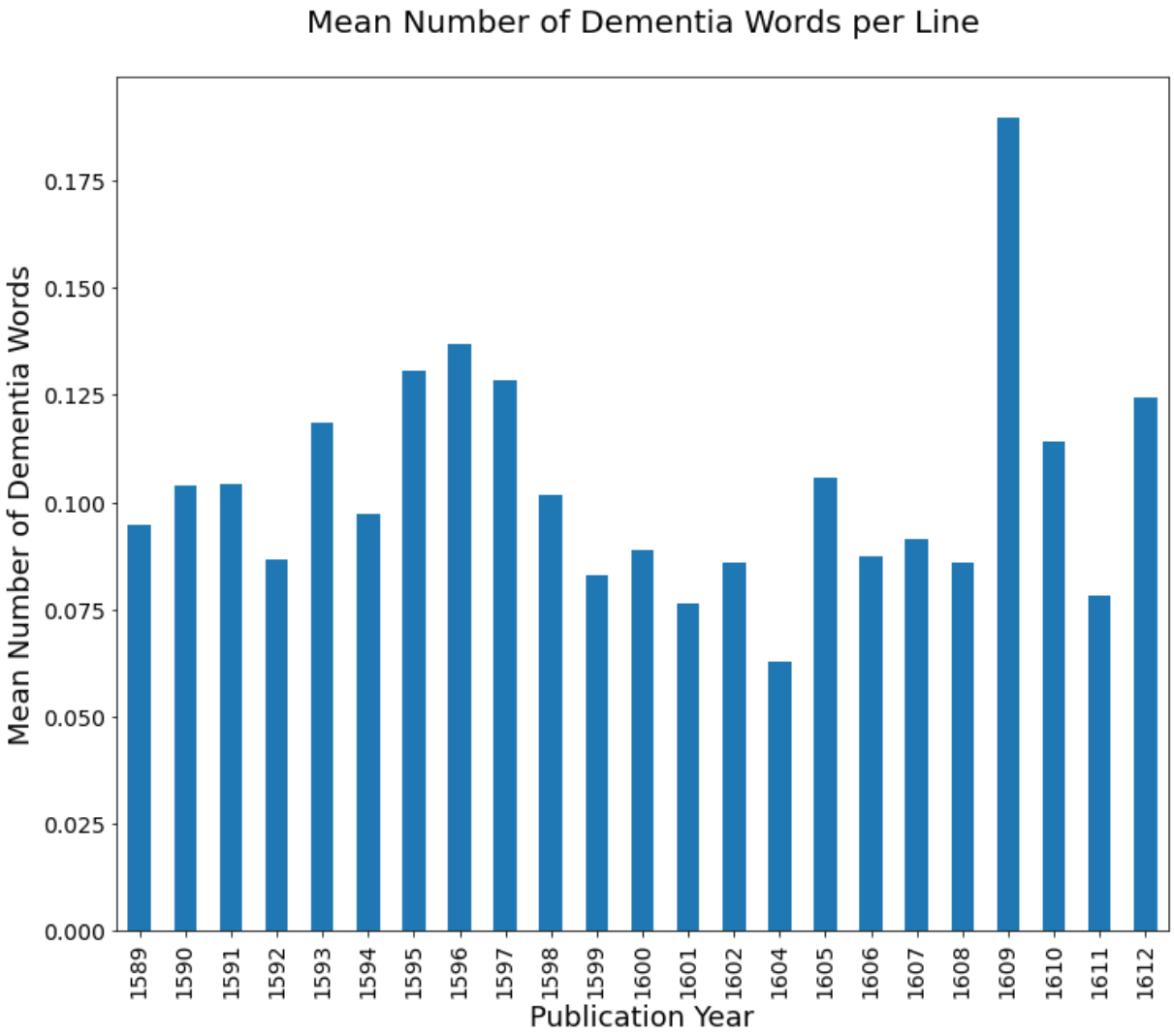
	Title	Publish Date	ParagraphNum	PlainText	tokenized
0	All's Well That Ends Well	1602	1.0	Enter BERTRAM, the COUNTESS of Rousillon, HELE...	enter bertram the countess of rousillon helena...
1	All's Well That Ends Well	1602	3.0	In delivering my son from me, I bury a second ...	in delivering my son from me i bury a second h...
2	All's Well That Ends Well	1602	4.0	And I in going, madam, weep o'er my father's d...	and i in going madam weep oer my fathers death...
3	All's Well That Ends Well	1602	7.0	You shall find of the king a husband, madam; y...	you shall find of the king a husband madam you...
4	All's Well That Ends Well	1602	12.0	What hope is there of his majesty's amendment?\n	what hope is there of his majestys amendment
...	...	...	...	...	...
34650	Winter's Tale	1610	3430.0	That she is living,\n[p]Were it but told you, ...	that she is living pwere it but told you shoul...
34651	Winter's Tale	1610	3437.0	You gods, look down\n[p]And from your sacred v...	you gods look down pand from your sacred vials...

	Title	Publish Date	ParagraphNum	PlainText	tokenized
34652	Winter's Tale	1610	3445.0	There's time enough for that;\n[p]Lest they de...	theres time enough for that plest they desire ...
34653	Winter's Tale	1610	3453.0	O, peace, Paulina!\n[p]Thou shouldst a husband...	o peace paulina pthou shouldst a husband take ...
34654	Winter's Tale	1610	3474.0	[Exeunt]	exeunt

34646 rows × 8 columns

In [46]:

```
# Chart the number of dementia words per line per year
ax=art_grps['num_dim_words'].aggregate(np.mean).plot(kind='bar', fonts
ax.set_title('Mean Number of Dementia Words per Line\n', fontsize=20)
ax.set_ylabel('Mean Number of Dementia Words', fontsize=18)
ax.set_xlabel('Publication Year', fontsize=18);
```



In [34]:

```
import requests
import json
import os

# Check if the word is in the local repository of definitions if not g
def check_definition(word):
    try:
        file = open('word_definitions.csv', 'r')
        file.close()
    except:
        file = open('word_definitions.csv', 'w+')
        if os.stat("word_definitions.csv").st_size == 0:
            file.write('word,definition')
        file.close()

    try:
        #if exits in local repository, return the definition
        word_definitions = pd.read_csv('word_definitions.csv', index_c
        if word in word_definitions.word.values:
            return word_definitions[word_definitions["word"]==word]["c
        #else, get the definition from the API
        else:
            definition = get_definition(word)
            #add the word and definition to the local repository
            word_definitions = word_definitions.append({'word':word, '
            # print(word_definitions, word, definition, " internet def
            word_definitions.to_csv('word_definitions.csv')
            return definition
    except:
        word_definitions = pd.read_csv('word_definitions.csv', index_c
        definition = ""
        #add the word and definition to the local repository
        word_definitions = word_definitions.append({'word':word, 'defi
        word_definitions.to_csv('word_definitions.csv')
        return definition


# Get the definition of a word from the API
def get_definition(word):
    word = word.lower()
```

```
    response = ''
#      717d065b-80fb-4a21-9aae-3ddbb7a5a2de
#      c7b1669c-7629-42a4-befd-3f32b966aa74

    base_url = "https://www.dictionaryapi.com/api/v3/references/colleg
    api = "c7b1669c-7629-42a4-befd-3f32b966aa74"


    api_key = "?key=" + api


    full_api = base_url + word + api_key
    try:

        response = requests.get(full_api)
        json_data = json.loads(response.text)
        definition = json_data[0]["shortdef"][0]
        definition = re.sub(' +', ' ', definition)
        definition = re.sub(',', '', definition)
        definition = re.sub("\'", '', definition)
        definition = re.sub('\[', '', definition)
        definition = re.sub('\]', '', definition)
        definition = re.sub('\{', '', definition)
        definition = re.sub('\}', '', definition)
        definition = re.sub('\"', '', definition)
        definition = re.sub('h:', '', definition)
        definition = re.sub('https', '', definition)
        definition = re.sub('http', '', definition)
        definition = re.sub('www', '', definition)
        definition = re.sub('\.', '', definition)
        definition = re.sub('\:', '', definition)
        definition = re.sub('\;', '', definition)
        definition = re.sub('\?', '', definition)
        definition = re.sub('\!', '', definition)
        definition = re.sub('\(', '', definition)
        definition = re.sub('\)', '', definition)
        definition = re.sub('\*', '', definition)
        definition = re.sub('\&', '', definition)
        definition = re.sub('\%', '', definition)
        definition = re.sub('\$', '', definition)
        definition = re.sub('\#', '', definition)
        definition = re.sub('\@', '', definition)
```

```
        definition = re.sub('\^', '', definition)
        definition = re.sub('\+', '', definition)
        definition = re.sub('\=', '', definition)
        definition = re.sub('\-', '', definition)
        definition = re.sub('\_', '', definition)
        definition = re.sub('\|', '', definition)
        definition = re.sub('\~', '', definition)
        definition = re.sub('\`', '', definition)
        definition = re.sub('\>', '', definition)
        definition = re.sub('\<', '', definition)
        definition = re.sub('\/', '', definition)
        return definition
    except:
        return ''
```

In [35]:

```
def word_complex(wordlist):
    stop_words = set(stopwords.words('english'))

    word_depth_value = 0

    known_words = set()
    known_words.add(wordlist)

    unknown_words = set()
    unknown_words2ndLine = set()
    try:
        wordlist = re.sub("[^a-zA-Z]", "", wordlist)
        word_list = wordlist.split(" ")
        for word in word_list:
            word_definition = check_definition(word)
            if (word_definition == -1):
                print('This code can\'t be run without an API key!\nYo
                return;

            word_definition_arr = word_definition.split(" ")
            for word in word_definition_arr:
                if word not in stop_words and len(word) > 1:
                    #print("Adding word: " + str(word))
                    unknown_words.add(word)

            while len(unknown_words) > 0:
                word = unknown_words.pop()
                known_words.add(word)
                word = re.sub("[^a-zA-Z]", "", word)
                word_definition = check_definition(word)
                try:
                    word_definition_arr = word_definition.split(" ")
                except:
                    continue

                for word in word_definition_arr:
                    if word not in known_words and word not in unknown
                        unknown_words2ndLine.add(word)
                        word_depth_value += 1
```

```
            if word_depth_value % 50 is 0:
                pass
                # print("NUM UNKNOWN WORDS: " + str(len(unknown_wo
                # print("NUM KNOWN WORDS: " + str(len(known_words)

            # print("Now I know " + str(word_depth_value) + str("

        for word in unknown_words2ndLine:
                if word not in known_words and word not in unknowr
                    unknown_words.add(word)

        # print("I needed to learn " + str(word_depth_value) + " w
        return word_depth_value
except:
    wordlist = re.sub("[^a-zA-Z]", "", wordlist)
    word_definition = check_definition(wordlist)
    if (word_definition == -1):
        print('This code can\'t be run without an API key!\nYou\']
        return;
    word_definition_arr = word_definition.split(" ")
    for word in word_definition_arr:
        if word not in stop_words and len(word) > 1:
            #print("Adding word: " + str(word))
            unknown_words.add(word)

    while len(unknown_words) > 0:
        word = unknown_words.pop()
        known_words.add(word)
        word = re.sub("[^a-zA-Z]", "", word)
        word_definition = check_definition(word)
        try:
            word_definition_arr = word_definition.split(" ")
        except:
            continue

        for word in word_definition_arr:
            if word not in known_words and word not in unknown_wor
                unknown_words2ndLine.add(word)
                word_depth_value += 1

        if word_depth_value % 50 is 0:
```

```
                pass

        for word in unknown_words2ndLine:
                if word not in known_words and word not in unknown_wor
                    unknown_words.add(word)

        # print("I needed to learn " + str(word_depth_value) + " words
        return word_depth_value

def word_complex2(wordlist):
    stop_words = set(stopwords.words('english'))

    word_depth_value = 0

    known_words = set()
    known_words.add(wordlist)

    unknown_words = set()
    unknown_words2ndLine = set()
    try:
        word_list = wordlist.split(" ")
        initial_complexity = 0
        total_complexity = 0
        for word in word_list:
            word = re.sub("[^a-zA-Z]", "", word)
            try:
                initial_complexity = check_complex(word)
                total_complexity += initial_complexity
            except:
                word_definition = check_definition(word)
                if (word_definition == -1):
                    print('This code can\'t be run without an API key!
                    return;

                word_definition_arr = word_definition.split(" ")
                for word in word_definition_arr:
                    if word not in stop_words and len(word) > 1:
                        #print("Adding word: " + str(word))
                        unknown_words.add(word)

                while len(unknown_words) > 0:
                    word = unknown_words.pop()
```

```
                known_words.add(word)
                word = re.sub("[^a-zA-Z]", "", word)
                word_definition = check_definition(word)
                try:
                    word_definition_arr = word_definition.split("
                except:
                    continue

                for word in word_definition_arr:
                    if word not in known_words and word not in unk
                        unknown_words2ndLine.add(word)
                        word_depth_value += 1

                if word_depth_value % 50 is 0:
                    pass
                    # print("NUM UNKNOWN WORDS: " + str(len(unknow
                    # print("NUM KNOWN WORDS: " + str(len(known_wo

                # print("Now I know " + str(word_depth_value) + st


            for word in unknown_words2ndLine:
                    if word not in known_words and word not in unk
                        unknown_words.add(word)

            # print("I needed to learn " + str(word_depth_value) +
            return word_depth_value
    return total_complexity
except:
    word = re.sub("[^a-zA-Z]", "", wordlist)
    try:
        return check_complex(word)
    except:
        word_definition = check_definition(word)
        if (word_definition == -1):
            print('This code can\'t be run without an API key!\nYo
            return;
        word_definition_arr = word_definition.split(" ")
        for word in word_definition_arr:
            if word not in stop_words and len(word) > 1:
                #print("Adding word: " + str(word))
                unknown_words.add(word)
```

```
            while len(unknown_words) > 0:
                word = unknown_words.pop()
                known_words.add(word)
                word = re.sub("[^a-zA-Z]", "", word)
                word_definition = check_definition(word)
                try:
                    word_definition_arr = word_definition.split(" ")
                except:
                    continue

                for word in word_definition_arr:
                    if word not in known_words and word not in unknowr
                        unknown_words2ndLine.add(word)
                        word_depth_value += 1

                if word_depth_value % 50 is 0:
                    pass
                    # print("NUM UNKNOWN WORDS: " + str(len(unknown_wo
                    # print("NUM KNOWN WORDS: " + str(len(known_words)

                # print("Now I know " + str(word_depth_value) + str("

            for word in unknown_words2ndLine:
                    if word not in known_words and word not in unknowr
                        unknown_words.add(word)

            # print("I needed to learn " + str(word_depth_value) + " w
            return word_depth_value


def check_complex(word):
    try:
        file = open('word_complex.csv', 'r')
        file.close()
    except:
        file = open('word_complex.csv', 'w+')
        if os.stat("word_complex.csv").st_size == 0:
            file.write('word,complexity')
        file.close()
```

```
try:
    #if exits in local repository, return the definition
    word_complexity = pd.read_csv('word_complex.csv', index_col=0)
    if word in word_complexity.word.values:
        return word_complexity[word_complexity["word"]==word]["com
    #else, get the definition from the API
    else:
        complexity = word_complex(word)
        #add the word and definition to the local repository
        word_complexity = word_complexity.append({'word':word, 'co
        # print(word_definitions, word, definition, " internet def
        word_complexity.to_csv('word_complex.csv')
        return complexity
except:
    word_complexity = pd.read_csv('word_complex.csv', index_col=0)
    complexity = 0
    #add the word and definition to the local repository
    word_complexity = word_complexity.append({'word':word, 'comple
    word_complexity.to_csv('word_complex.csv')
    return complexity
```

```
<>:41: SyntaxWarning: "is" with a literal. Did you mean
"=="?
<>:82: SyntaxWarning: "is" with a literal. Did you mean
"=="?
<>:138: SyntaxWarning: "is" with a literal. Did you mea
n "=="?
<>:183: SyntaxWarning: "is" with a literal. Did you mea
n "=="?
<>:41: SyntaxWarning: "is" with a literal. Did you mean
"=="?
<>:82: SyntaxWarning: "is" with a literal. Did you mean
"=="?
<>:138: SyntaxWarning: "is" with a literal. Did you mea
n "=="?
<>:183: SyntaxWarning: "is" with a literal. Did you mea
n "=="?
C:\Users\theoj\AppData\Local\Temp\ipykernel_1580\184334
1898.py:41: SyntaxWarning: "is" with a literal. Did you
mean "=="?
  if word_depth_value % 50 is 0:
C:\Users\theoj\AppData\Local\Temp\ipykernel_1580\184334
1898.py:82: SyntaxWarning: "is" with a literal. Did you
mean "=="?
  if word_depth_value % 50 is 0:
C:\Users\theoj\AppData\Local\Temp\ipykernel_1580\184334
1898.py:138: SyntaxWarning: "is" with a literal. Did yo
u mean "=="?
  if word_depth_value % 50 is 0:
C:\Users\theoj\AppData\Local\Temp\ipykernel_1580\184334
1898.py:183: SyntaxWarning: "is" with a literal. Did yo
u mean "=="?
  if word_depth_value % 50 is 0:
```

In [36]:

```
word_complex2("""did the barber shave the barber""")
# test word: love in the word_complex function
```

Out[36]:

```
335
```

In [37]:

```
#calculation of word_complex of pt1.PlainText and store values in a ne
if os.path.exists('pt1.csv'):
    pt1 = pd.read_csv('pt1.csv', index_col=0)
else:
    pt1['word_complexity'] = pt1['PlainText'].apply(word_complex2)
pt1
```

Out[37]:

	Title	Publish Date	ParagraphNum	PlainText	tokenized
0	All's Well That Ends Well	1602	1.0	Enter BERTRAM, the COUNTESS of Rousillon, HELE...	enter bertram the countess of rousillon helena...
1	All's Well That Ends Well	1602	3.0	In delivering my son from me, I bury a second ...	in delivering my son from me i bury a second h...
2	All's Well That Ends Well	1602	4.0	And I in going, madam, weep o'er my father's d...	and i in going madam weep oer my fathers death...
3	All's Well That Ends Well	1602	7.0	You shall find of the king a husband, madam; y...	you shall find of the king a husband madam you...
4	All's Well That Ends Well	1602	12.0	What hope is there of his majesty's amendment?\n	what hope is there of his majestys amendment
...	...	...	...	...	...
34650	Winter's Tale	1610	3430.0	That she is living,\n[p]Were it but told you, ...	that she is living pwere it but told you shoul...
34651	Winter's Tale	1610	3437.0	You gods, look down\n[p]And from your sacred v...	you gods look down pand from your sacred vials...

	Title	Publish Date	ParagraphNum	PlainText	tokenized
34652	Winter's Tale	1610	3445.0	There's time enough for that;\n[p]Lest they de...	theres time enough for that plest they desire ...
34653	Winter's Tale	1610	3453.0	O, peace, Paulina!\n[p]Thou shouldst a husband...	o peace paulina pthou shouldst a husband take ...
34654	Winter's Tale	1610	3474.0	[Exeunt]	exeunt

34646 rows × 9 columns

In [38]:

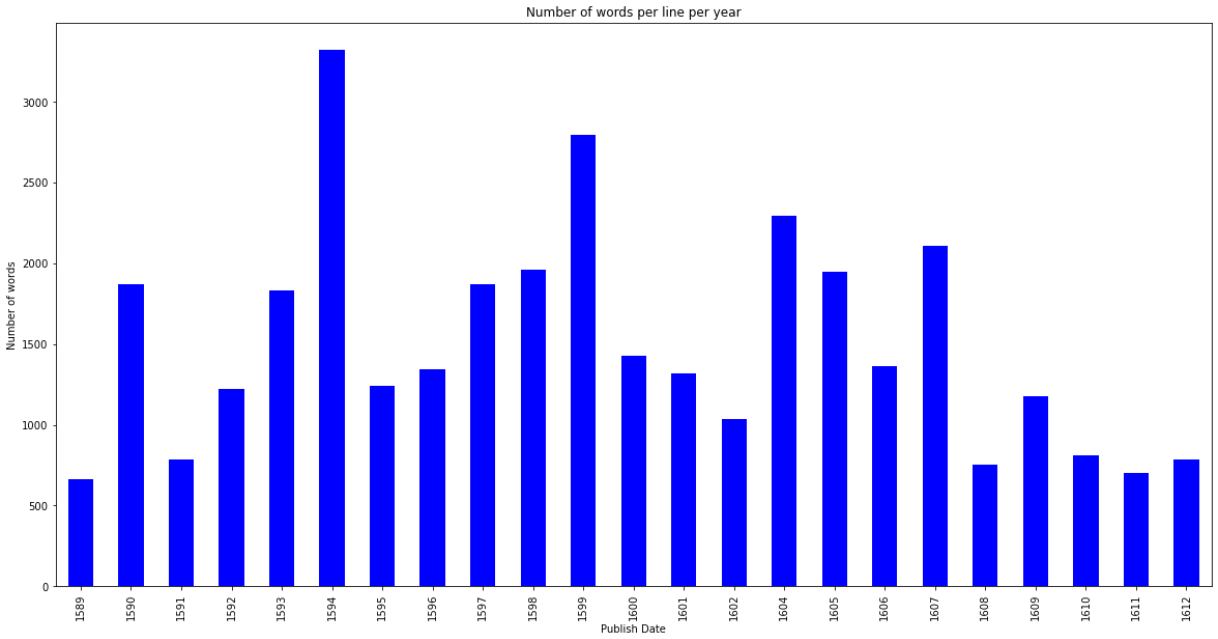
```
pt1.to_csv('pt1.csv')
# stores a copy of the dataframe in a csv file
```

In [48]:

```
# Chart the number of pt1.words_complexity per line per year all  comp
pt1.groupby(['Publish Date', 'word_complexity']).size().unstack().plot
```

Out[48]:

```
<AxesSubplot:title={'center':'Number of words per line
per year'}, xlabel='Publish Date', ylabel='Number of wo
rds'>
```

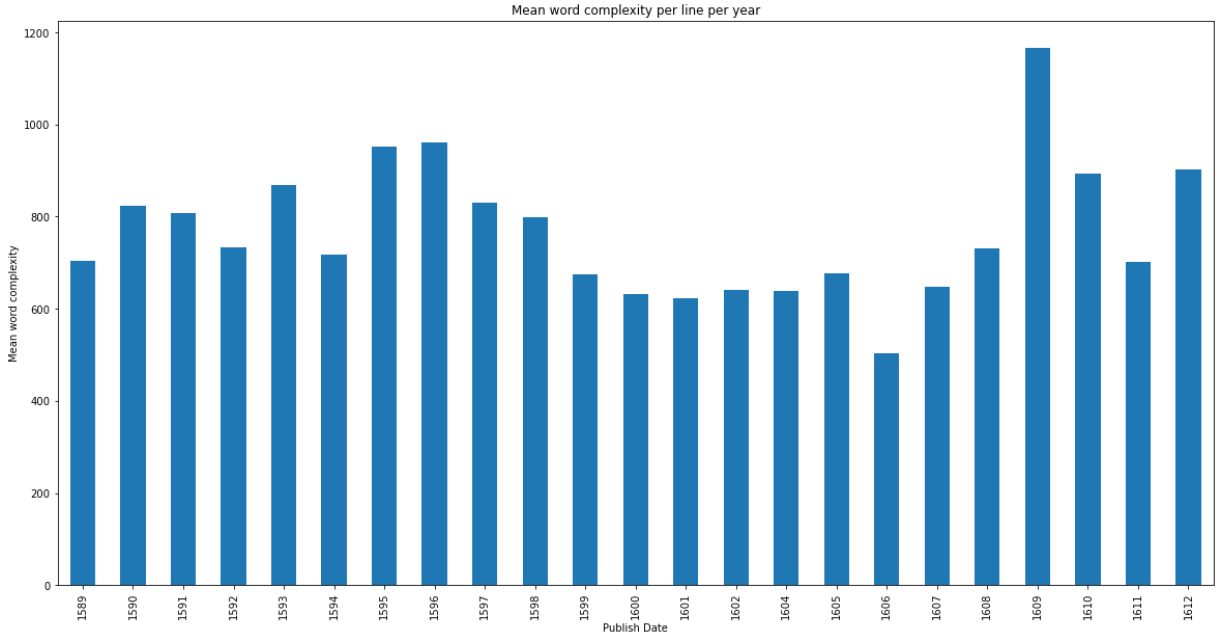


In [49]:

```
import matplotlib.pyplot as plt
# Chart the number of mean word_complexity per line per year with titl
pt1.groupby(['Publish Date'])['word_complexity'].mean().plot(kind='bar
plt.title('Mean word complexity per line per year')
plt.xlabel('Publish Date')
plt.ylabel('Mean word complexity')
```

Out[49]:

```
Text(0, 0.5, 'Mean word complexity')
```



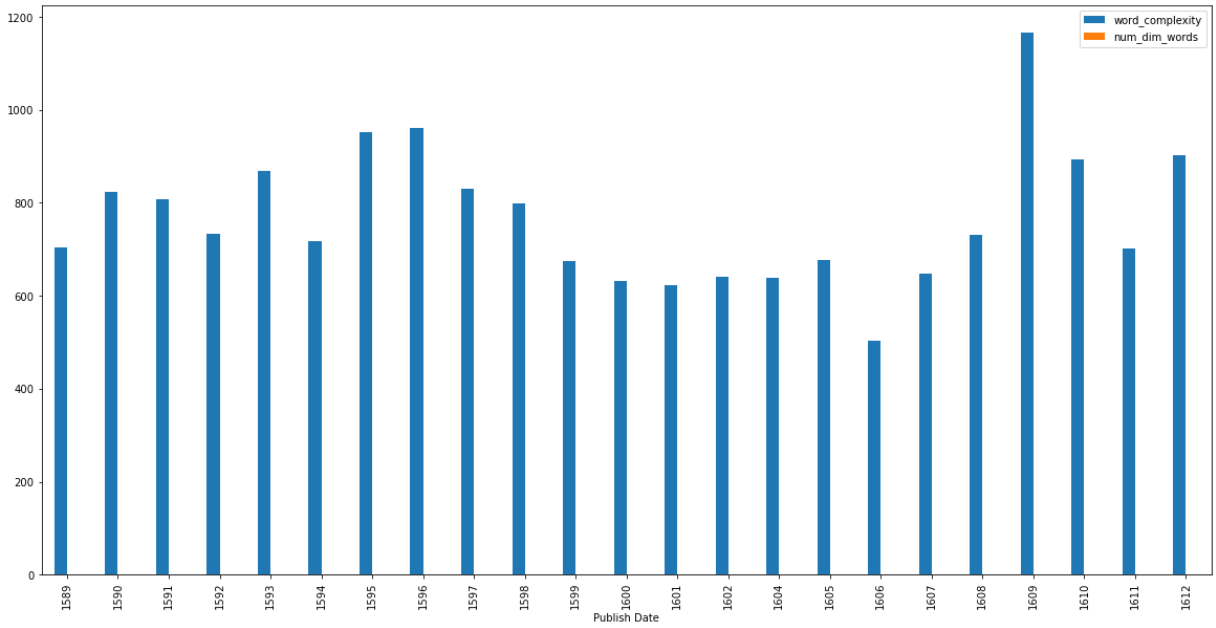
In [41]:

```
# Chart the number of mean word_complexity and num_dim_words per line
pt1.groupby(['Publish Date'])['word_complexity', 'num_dim_words'].mear
plt.title('Mean word complexity and num_dim_words per line per year')
plt.xlabel('Publish Date')
plt.ylabel('Mean word complexity and num_dim_words')
```

```
C:\Users\theoj\AppData\Local\Temp\ipykernel_1580\149934
1372.py:2: FutureWarning: Indexing with multiple keys
(implicitly converted to a tuple of keys) will be depre
cated, use a list instead.
  pt1.groupby(['Publish Date'])['word_complexity', 'num
_dim_words'].mean().plot(kind='bar', figsize=(20,10))
```

Out[41]:

```
<AxesSubplot:xlabel='Publish Date'>
```



In [42]:

```
# create new column for the mean word_complexity and num_dim_words per
pt1['mean_word_complexity'] = pt1.groupby(['Publish Date'])['word_comp
# create new column for the mean num_dim_words per line per year
pt1['mean_num_dim_words'] = pt1.groupby(['Publish Date'])['num_dim_wor
#create new column for the product of mean_word_complexity and mean_nu
pt1['mean_dim_word_complexity'] = pt1['mean_word_complexity']/pt1['mea
pt1
```

Out[42]:

	Title	Publish Date	ParagraphNum	PlainText	tokenized
0	All's Well That Ends Well	1602	1.0	Enter BERTRAM, the COUNTESS of Rousillon, HELE...	enter bertram the countess of rousillon helena...
1	All's Well That Ends Well	1602	3.0	In delivering my son from me, I bury a second ...	in delivering my son from me i bury a second h...
2	All's Well That Ends Well	1602	4.0	And I in going, madam, weep o'er my father's d...	and i in going madam weep oer my fathers death...
3	All's Well That Ends Well	1602	7.0	You shall find of the king a husband, madam; y...	you shall find of the king a husband madam you...
4	All's Well That Ends Well	1602	12.0	What hope is there of his majesty's amendment?\n	what hope is there of his majestys amendment
...	...	...	...	...	...
34650	Winter's Tale	1610	3430.0	That she is living,\n[p]Were it but told you, ...	that she is living pwere it but told you shoul...
34651	Winter's Tale	1610	3437.0	You gods, look down\n[p]And from your sacred v...	you gods look down pand from your sacred vials...

	Title	Publish Date	ParagraphNum	PlainText	tokenized
34652	Winter's Tale	1610	3445.0	There's time enough for that;\n[p]Lest they de...	theres time enough for that plest they desire ...
34653	Winter's Tale	1610	3453.0	O, peace, Paulina!\n[p]Thou shouldst a husband...	o peace paulina pthou shouldst a husband take ...
34654	Winter's Tale	1610	3474.0	[Exeunt]	exeunt

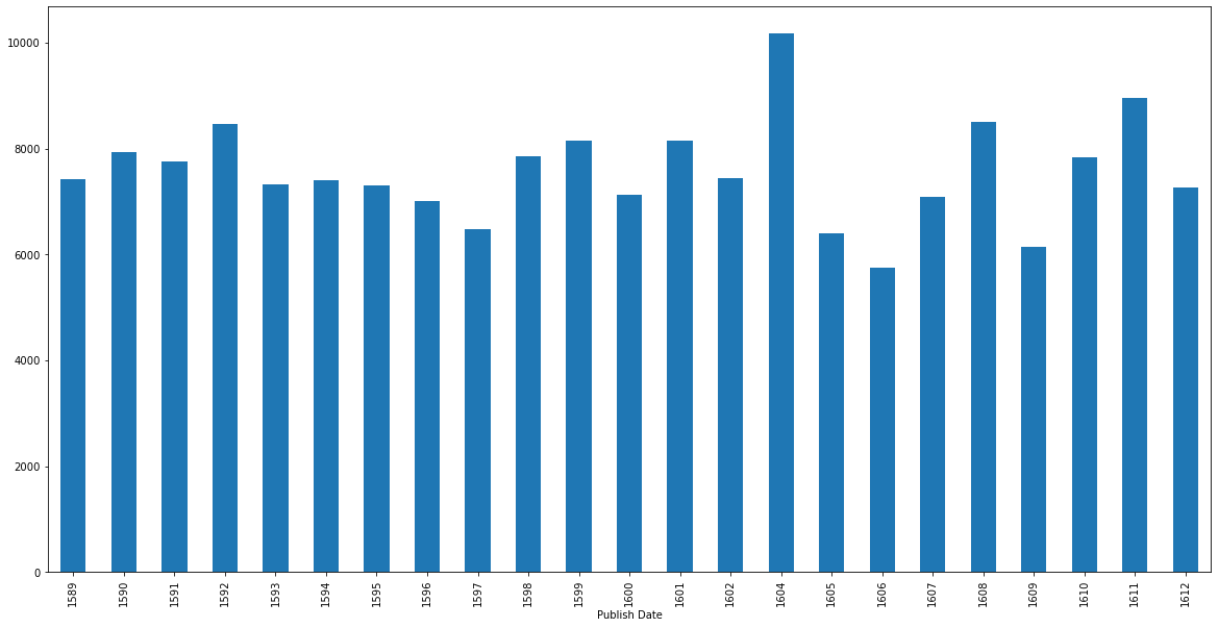
34646 rows × 12 columns

In [43]:

```
# Chart the mean_dim_word_complexity per line per year
pt1.groupby(['Publish Date'])['mean_dim_word_complexity'].mean().plot(
```

Out[43]:

```
<AxesSubplot:xlabel='Publish Date'>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: