In [2]:

```
import pandas as pd
```

In [3]:

```
df = pd.read_csv("training.tsv", sep='\t')
```

In [4]:

```
df.shape
```

Out[4]:

```
(1200000, 3)
```

In [5]:

```
df.head(10)
```

Out[5]:

	title	description	category
0	ZicZac // Black + Red (Euro: 44)	Clothing & related products (B2C) - Shoes and ...	R
1	9X9 RESISTA/484938	Publishing/Printing - Printing Services	S
2	Halle Pant - Short Inseam 013049561D0010001_ 02	Clothing & related products (B2C) - General	R
3	Harry Houser Travel Expenses - Meals	Security - personnel	S
4	Tee Time: 740078609 : Greens Fee - Composite	Admissions - Green Fees for Privately Owned Go...	R
5	Flat Rate (5-7 Business Days) Shipping line: 4...	Shipping Only - common carrier - FOB destination	R
6	Travel to Water Batteries Plant 1 During regul...	Repair (other) - Performed on TPP (labor only)	S
7	F5 Networks Consulting Services Standard Hourl...	Installation - associated with the sale of TPP...	S
8	Network Time and Materials Services - May 2019...	Consulting - Systems	S
9	2c92a0ad707bb947017095aa4a973307	Cloud Services-Platform as a Service (PaaS)	S

In [6]:

```
df['title']
```

Out[6]:

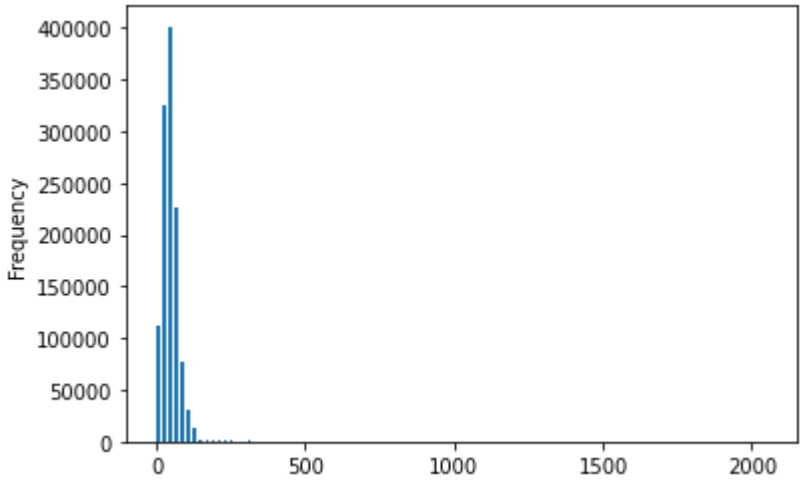
```
0                            ZicZac // Black + Red (Euro: 44)
1                                        9X9 RESISTA/484938
2              Halle Pant - Short Inseam 013049561D0010001_ 02
3                        Harry Houser Travel Expenses - Meals
4                 Tee Time: 740078609 : Greens Fee - Composite
                                      ...
1199995    Gafford Family Medicine: First DataBank Drug D...
1199996                       Video Rental (order # 215505199)
1199997                 Hope For All Rhinestone RIbbon Tee - L
1199998                    AriaCounterpart (order # 304541704)
1199999    Premium support renewal, PA-5220  Serial Numbe...
Name: title, Length: 1200000, dtype: object
```

In [7]:

```
df['title_len'] = df['title'].astype(str).apply(len)
```

In [8]:

```
ax = df['title_len'].plot.hist(bins=100, alpha=1.0, width=10)
```

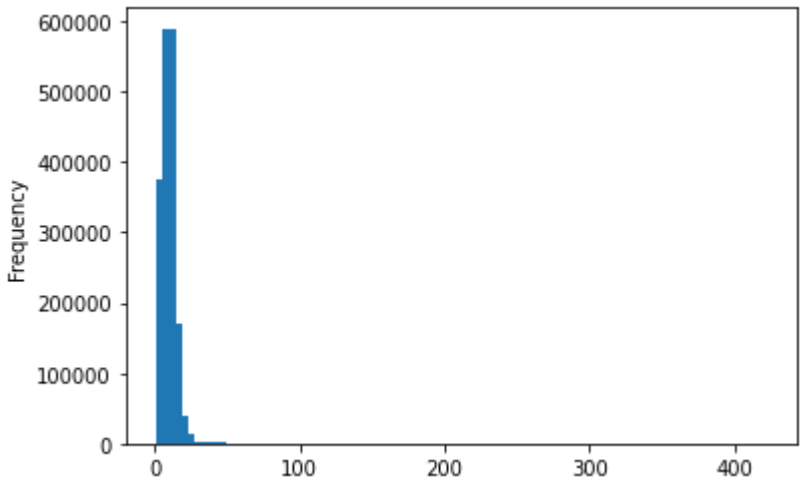


In [9]:

```
df['word_count'] = df['title'].apply(lambda x: len(str(x).split()))
```

In [10]:

```
ax1 = df['word_count'].plot.hist(bins=100, alpha=1.0, width=10)
```



In [11]:

```
df['class'] = df['category'].replace(to_replace ="S",
                 value =1)
```

In [12]:

```
df['class'] = df['class'].replace(to_replace ="R",
                 value =0)
```

In [13]:

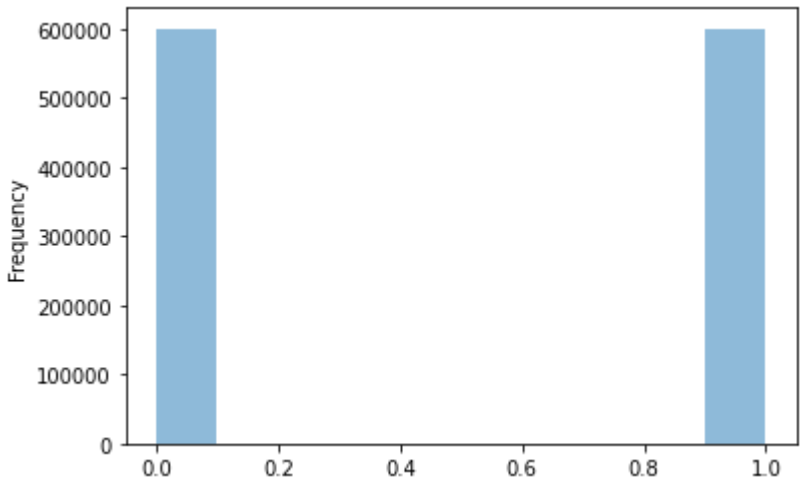
```
df['class']
```

Out[13]:

```
0          0
1          1
2          0
3          1
4          0
          ..
1199995    1
1199996    0
1199997    0
1199998    0
1199999    1
Name: class, Length: 1200000, dtype: int64
```

In [14]:

```
ax1 = df['class'].plot.hist(bins=10, alpha=0.5)
```



In [15]:

```
# almost equal number of samples for each class. Balanced Data.
```

In [16]:

```
df[~df['title'].isna()]['title']
```

Out[16]:

```
0                            ZicZac // Black + Red (Euro: 44)
1                                        9X9 RESISTA/484938
2              Halle Pant - Short Inseam 013049561D0010001_ 02
3                        Harry Houser Travel Expenses - Meals
4                  Tee Time: 740078609 : Greens Fee - Composite
                                    ...
1199995    Gafford Family Medicine: First DataBank Drug D...
1199996                         Video Rental (order # 215505199)
1199997                  Hope For All Rhinestone RIbbon Tee - L
1199998                     AriaCounterpart (order # 304541704)
1199999    Premium support renewal, PA-5220  Serial Numbe...
Name: title, Length: 1199998, dtype: object
```

In [19]:

```
from sklearn.feature_extraction.text import CountVectorizer
def get_top_n_words(corpus, n=None):

    vec = CountVectorizer().fit(corpus.astype('U'))
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.ite
ms()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

corpus = df[~df['title'].isna()]

corpus_title = corpus['title']

corpus_service = corpus[corpus['category']=='S']['title']
corpus_product = corpus[corpus['category']=='R']['title']

print(corpus_service.shape)
print(corpus_product.shape)

common_words_service = get_top_n_words(corpus_service.sample(frac=0.1), 20)
common_words_product = get_top_n_words(corpus_product.sample(frac=0.1), 20)


df_s = pd.DataFrame(common_words_service, columns = ['newtitle' , 'count'])
df_s = df_s.groupby('newtitle').sum()['count'].sort_values(ascending=False)

df_p = pd.DataFrame(common_words_product, columns = ['newtitle' , 'count'])
df_p = df_p.groupby('newtitle').sum()['count'].sort_values(ascending=False)
```

```
(600176,)
(599822,)
```

In [20]:

```
df_s  # top 20 service words
```

Out[20]:

```
newtitle
com        9310
for        8512
2019       8041
2018       6432
2017       6421
00         6015
of         5359
12         5148
10         5049
fee        4877
service    4783
01         4618
11         4567
year       4494
renewal    4419
2016       4160
07         3668
08         3653
to         3610
09         3605
Name: count, dtype: int64
```

In [21]:

```
df_p   # top 20 product words.
```

Out[21]:

```
newtitle
line               25828
item               24246
variant            24126
order              11940
black               4140
fee                 3709
tee                 3553
composite           3481
greens              3477
time                3422
ariacounterpart     2503
shipping            2434
all                 1665
video               1595
rental              1433
access              1430
now                 1395
hbo                 1378
white               1355
large               1311
Name: count, dtype: int64
```

In [22]:

```
# from the above top 20 words we can see that for services mostly the date and t
ime are mentioned.
# Whereas for products we can see that those date and time are not common words.
# So Date and Time related words are crucial for distinguishing between them.
```

In [24]:

```
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer(stop_words = 'english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.ite
ms()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]


corpus = df[~df['title'].isna()]

corpus_title = corpus['title']

corpus_service = corpus[corpus['category']=='S']['title']
corpus_product = corpus[corpus['category']=='R']['title']

print(corpus_service.shape)
print(corpus_product.shape)

common_words_service = get_top_n_words(corpus_service.sample(frac=0.1), 20)
common_words_product = get_top_n_words(corpus_product.sample(frac=0.1), 20)


df_s = pd.DataFrame(common_words_service, columns = ['newtitle' , 'count'])
df_s = df_s.groupby('newtitle').sum()['count'].sort_values(ascending=False)

df_p = pd.DataFrame(common_words_product, columns = ['newtitle' , 'count'])
df_p = df_p.groupby('newtitle').sum()['count'].sort_values(ascending=False)
```

```
(600176,)
(599822,)
```

In [27]:

```
# top 20 words in products after removing stop words.
df_p
```

Out[27]:

```
newtitle
line                25952
item                24412
variant             24299
order               11828
black                4247
fee                  3679
tee                  3512
greens               3441
composite            3439
time                 3371
ariacounterpart      2416
shipping             2400
video                1631
rental               1490
white                1443
access               1439
hbo                  1354
hulu                 1303
large                1292
blue                 1253
Name: count, dtype: int64
```

In [29]:

```
# top 20 words after removing stop words for service. words like 'for' and 'of'
 are eliminated.
df_s
```

Out[29]:

```
newtitle
com         9342
2019        8094
2018        6614
2017        6335
00          5939
12          5111
10          4952
fee         4940
service     4746
01          4682
11          4640
year        4482
renewal     4387
2016        4301
08          3715
07          3673
02          3614
06          3481
17          3441
09          3429
Name: count, dtype: int64
```

In [35]:

```
def get_top_n_bigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.ite
ms()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

corpus = df[~df['title'].isna()]

corpus_title = corpus['title']

corpus_service = corpus[corpus['category']=='S']['title']
corpus_product = corpus[corpus['category']=='R']['title']

print(corpus_service.shape)
print(corpus_product.shape)

common_words_service = get_top_n_bigram(corpus_service.sample(frac=0.1), 20)
common_words_product = get_top_n_bigram(corpus_product.sample(frac=0.1), 20)


df_s = pd.DataFrame(common_words_service, columns = ['newtitle' , 'count'])
df_s_bigram = df_s.groupby('newtitle').sum()['count'].sort_values(ascending=Fals
e)

df_p = pd.DataFrame(common_words_product, columns = ['newtitle' , 'count'])
df_p_bigram = df_p.groupby('newtitle').sum()['count'].sort_values(ascending=Fals
e)
```

```
(600176,)
(599822,)
```

In [36]:

```
df_p_bigram
```

Out[36]:

```
newtitle
line item                24368
greens fee                3432
fee composite             3432
tee time                  3335
ariacounterpart order     2429
black line                1930
all access                1437
shipping line             1418
hbo now                   1394
rental order              1319
video rental              1319
cbs all                   1122
now order                 1017
commercials order          910
access order               872
showtime order             799
plus order                 752
limited commercials        692
xl line                    689
hulu plus                  657
Name: count, dtype: int64
```

In [37]:

```
df_s_bigram
```

Out[37]:

```
newtitle
recurring fee          2893
for year               2385
com for                1934
renewal of             1734
premium uid            1624
lastpass premium       1616
00 00                  1576
domain renewal         1529
com year               1231
guard service          1207
line item              1187
stationary guard       1127
registration of        1019
2017 through            985
quantity vatnumber      961
vatnumber ipaddress     957
coverage dates          926
2018 through            919
gb hours                855
backup storage          852
Name: count, dtype: int64
```

In [38]:

```
def get_top_n_bigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2), stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.ite
ms()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

corpus = df[~df['title'].isna()]

corpus_title = corpus['title']

corpus_service = corpus[corpus['category']=='S']['title']
corpus_product = corpus[corpus['category']=='R']['title']

print(corpus_service.shape)
print(corpus_product.shape)

common_words_service = get_top_n_bigram(corpus_service.sample(frac=0.1), 20)
common_words_product = get_top_n_bigram(corpus_product.sample(frac=0.1), 20)


df_s = pd.DataFrame(common_words_service, columns = ['newtitle' , 'count'])
df_s_bigram = df_s.groupby('newtitle').sum()['count'].sort_values(ascending=Fals
e)

df_p = pd.DataFrame(common_words_product, columns = ['newtitle' , 'count'])
df_p_bigram = df_p.groupby('newtitle').sum()['count'].sort_values(ascending=Fals
e)
```

```
(600176,)
(599822,)
```

In [41]:

```
df_p_bigram # after stop word removal.
```

Out[41]:

```
newtitle
line item                24148
fee composite             3419
greens fee                3419
tee time                  3318
ariacounterpart order     2500
black line                1887
video rental              1419
rental order              1419
shipping line             1393
cbs access                1077
hbo order                 1007
commercials order          933
access order               870
showtime order             801
xl line                    726
plus order                 706
limited commercials        698
white line                 599
hulu plus                  588
large line                 565
Name: count, dtype: int64
```

In [42]:

```
df_s_bigram # after stop word removal.
```

Out[42]:

```
newtitle
recurring fee          2959
com year               2825
premium uid            1652
lastpass premium       1644
00 00                  1594
domain renewal         1528
guard service          1254
line item              1151
stationary guard       1151
coverage dates          949
quantity vatnumber      910
vatnumber ipaddress     908
gb hours                892
backup storage          884
lppremium uid           798
type lppremium          798
fee linux               650
id protection           641
security officer        526
purchase privacy        484
Name: count, dtype: int64
```

In [44]:

```
def get_top_n_trigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(3, 3)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.ite
ms()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

corpus = df[~df['title'].isna()]

corpus_title = corpus['title']

corpus_service = corpus[corpus['category']=='S']['title']
corpus_product = corpus[corpus['category']=='R']['title']

print(corpus_service.shape)
print(corpus_product.shape)

common_words_service = get_top_n_trigram(corpus_service.sample(frac=0.1), 20)
common_words_product = get_top_n_trigram(corpus_product.sample(frac=0.1), 20)


df_s = pd.DataFrame(common_words_service, columns = ['newtitle' , 'count'])
df_s_trigram = df_s.groupby('newtitle').sum()['count'].sort_values(ascending=Fal
se)

df_p = pd.DataFrame(common_words_product, columns = ['newtitle' , 'count'])
df_p_trigram = df_p.groupby('newtitle').sum()['count'].sort_values(ascending=Fal
se)
```

```
(600176,)
(599822,)
```

In [45]:

```
df_s_trigram
```

Out[45]:

```
newtitle
lastpass premium uid            1681
com for year                    1661
stationary guard service        1123
quantity vatnumber ipaddress     867
type lppremium uid               759
recurring fee linux              598
00 00 00                         525
17 stationary guard              487
purchase of privacy              464
protection service for           464
privacy protection service       464
service for domain               464
of privacy protection            464
18 stationary guard              442
fee linux standard               426
linux standard business          423
service credit no                380
account service credit           380
standard business hosting        338
business hosting business        338
Name: count, dtype: int64
```

In [46]:

```
df_p_trigram
```

Out[46]:

```
newtitle
greens fee composite          3443
black line item               1969
video rental order            1434
cbs all access                1001
hbo now order                  994
all access order               815
hulu plus order                670
xl line item                   662
limited commercials order      653
white line item                604
large line item                581
blue line item                 566
medium line item               548
pack line item                 502
gift line item                 463
shipping shipping line         462
monthly subscription order     416
days shipping line             400
hulu limited commercials       396
navy line item                 394
Name: count, dtype: int64
```

In [47]:

```
def get_top_n_trigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(3, 3), stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.ite
ms()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

corpus = df[~df['title'].isna()]

corpus_title = corpus['title']

corpus_service = corpus[corpus['category']=='S']['title']
corpus_product = corpus[corpus['category']=='R']['title']

print(corpus_service.shape)
print(corpus_product.shape)

common_words_service = get_top_n_trigram(corpus_service.sample(frac=0.1), 20)
common_words_product = get_top_n_trigram(corpus_product.sample(frac=0.1), 20)


df_s = pd.DataFrame(common_words_service, columns = ['newtitle' , 'count'])
df_s_trigram = df_s.groupby('newtitle').sum()['count'].sort_values(ascending=Fal
se)

df_p = pd.DataFrame(common_words_product, columns = ['newtitle' , 'count'])
df_p_trigram = df_p.groupby('newtitle').sum()['count'].sort_values(ascending=Fal
se)
```

```
(600176,)
(599822,)
```

In [48]:

```
df_s_trigram # after removing stop words.
```

Out[48]:

```
newtitle
lastpass premium uid            1581
stationary guard service        1131
quantity vatnumber ipaddress     933
type lppremium uid               832
recurring fee linux              633
00 00 00                         520
purchase privacy protection      476
protection service domain        476
privacy protection service       476
17 stationary guard              475
18 stationary guard              453
fee linux standard               450
linux standard business          449
account service credit           393
standard business hosting        375
hosting business standard        374
business hosting business        374
users monthly payments           320
digital iris service             319
monthly payments renewal         308
Name: count, dtype: int64
```

In [50]:

```
df_p_trigram  # after removing stop words.
```

Out[50]:

```
newtitle
greens fee composite          3479
black line item               1954
video rental order            1363
cbs access order               911
xl line item                   729
limited commercials order      669
hulu plus order                648
white line item                599
large line item                577
blue line item                 541
medium line item               534
pack line item                 480
gift line item                 474
shipping shipping line         474
hulu limited commercials       429
monthly subscription order     423
days shipping line             412
brynn rich black               372
business days shipping         368
navy line item                 354
Name: count, dtype: int64
```

In [60]:

```
df['title'].shape
```

Out[60]:

```
(1200000,)
```

In [82]:

```
from textblob import TextBlob
import nltk
nltk.download('averaged_perceptron_tagger')

all = [str(i) for i in list(df['title'].sample(frac=0.05).values)]
all = ' '.join(all)
blob = TextBlob(all)
#print(blob.tags)
pos_df = pd.DataFrame(blob.tags, columns = ['word' , 'pos'])
print(pos_df['pos'])
pos_df = pos_df.pos.value_counts()
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/rnsandeep/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!

0          JJ
1         NNP
2          NN
3          CD
4         NNP
         ...
416338    NNP
416339    NNP
416340    NNP
416341     JJ
416342     NN
Name: pos, Length: 416343, dtype: object
```

In [83]:

```
pos_df # part of speech.
```

Out[83]:

```
NNP     194077
CD       84555
NN       68599
JJ       22410
IN       14357
NNS       8275
VBD       4004
CC        3302
VBG       2878
DT        2257
TO        1801
NNPS      1386
VB        1280
VBN       1279
RB        1064
VBZ       1010
POS        899
VBP        875
FW         644
PRP        405
PRP$       223
RP         188
MD         127
SYM         94
JJR         87
JJS         80
WRB         70
WDT         52
WP          36
EX           7
UH           7
PDT          6
RBR          5
LS           3
RBS          1
Name: pos, dtype: int64
```

In [ ]:

```

```