Aspect Classification

**What are Aspects?**

People write reviews for products and services. Both products and services have different aspects on which people give their opinions. For example, in restaurant domain, someone wrote  ‘**The food in this restaurant is realllllly awsooome but the staff behaviour irks.,,...**’. Here, that guy talked about two aspects of restaurant domain, **food** and **staff**. Similarly in products, for example in mobiles people talk about,  battery, storage capacity, OS, camera etc.  So, **Something which is rateable is an aspect.**

Given a set of sentences, classify each word in each sentence into Aspect (ASP) and Non - Aspect(NASP).

Ex:  **INPUT**:  list of reviews in input text file

food

place

restaurant

service

time

price

friend

pizza

staff

dish

menu

roll

Thai

.

,

**OUTPUT**: food ASP

place ASP

restaurant ASP

service ASP

time ASP

price ASP

friend NASP

pizza ASP

staff ASP

dish ASP

menu ASP

Thai NASP

fish NASP

roll NASP

. NASP

, NASP

# With pandas, we can also view **aggregate statistics** easily:

Out[46]:

words

Aspects

ASP count 1020

unique 208

top food

freq 90

NASP count 9491

unique 1944

top .

freq 773

train\_data.length.describe()

Out[52]:

count 10511.000000

mean 3.921606

std 2.451187

min 1.000000

25% 2.000000

50% 3.000000

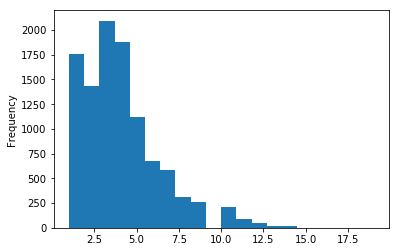
75% 5.000000

max 19.000000

Name: length, dtype: float64

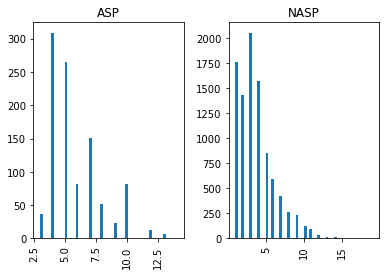
#Length of the words

train\_data.length.plot(bins=20, kind='hist')



# difference in message length between aspect(ASP) and non-aspect(NASP)

train\_data.hist(column='length', by='Aspects', bins=50)



#Data processing

##and normalize words into their base form (lemmas) with:

In [15]:

**def** split\_into\_lemmas(train\_words):

train\_words = train\_words.lower()

train\_words = TextBlob(train\_words).words

*# for each word, take its "base form" = lemma*

**return** [word.lemma **for** word **in** words]

train\_data.train\_words.head().apply(split\_into\_lemmas)

# **Data to vectors**

print(len(count\_vectors.vocabulary\_))

1812

#sparse matrix shape, non-zeros, sparsity

print( 'sparse matrix shape:', train\_cv.shape )

print( 'number of non-zeros:', train\_cv.nnz )

print( 'sparsity: %.2f%%' % (100.0 \* train\_cv.nnz / (train\_cv.shape[0] \* train\_cv.shape[1])))

sparse matrix shape: (10511, 1812)

number of non-zeros: 8786

sparsity: 0.05%

# term frequency

print(tfidf\_transformer1)

(1, 1783) 1.0

(2, 652) 1.0

(3, 124) 1.0

(6, 660) 1.0

(7, 67) 1.0

(8, 515) 1.0

(9, 1145) 1.0

(10, 1034) 1.0

(12, 68) 1.0

(13, 1727) 1.0

(14, 683) 1.0

(15, 1373) 1.0

(16, 1070) 1.0

(17, 1008) 1.0

(21, 1783) 1.0

(22, 1248) 1.0

(24, 592) 1.0

(25, 1544) 1.0

(27, 224) 1.0

(28, 1022) 1.0

(29, 592) 1.0

(30, 1549) 1.0

(31, 588) 1.0

(33, 989) 1.0

(35, 68) 1.0

: :

### **Training a model, detecting aspect**

#MultinomialNB

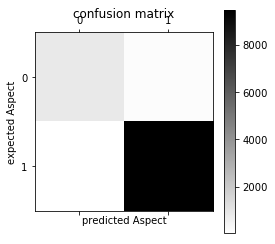
accuracy 0.982304252688

confusion matrix

[[ 869 151]

[ 35 9456]]

(row=expected, col=predicted)



#BernoulloNB

accuracy 0.9578536771

confusion matrix

[[ 595 425]

[ 18 9473]]

(row=expected, col=predicted)

#From this confusion matrix, we can compute precision and recall, or their combination (harmonic mean) F1:

###train data

print(classification\_report(train\_Aspects, all\_predictions2))

precision recall f1-score support

ASP 0.96 0.85 0.90 1020

NASP 0.98 1.00 0.99 9491

avg / total 0.98 0.98 0.98 10511

##test data

print(classification\_report(test\_Aspects, all\_predictions))

precision recall f1-score support

ASP 0.96 0.77 0.85 348

NASP 0.97 1.00 0.98 2986

avg / total 0.97 0.97 0.97 3334

accuracy 0.972405518896

#Generate a simple plot of the test and traning learning curve.



Conclusion:

Prediction of aspected words from sentences is almost over. In this problem I am choosing best prediction classifier by based on accuracy of the naïve bayes(Multinomial-98% and Bernoullo-95%) and svm-96% algorithms. And now I am working on classification algorithms. My aim is to classify each word in each sentence into Aspect (ASP) and Non - Aspect(NASP). Right now I am working on this project.

Ex:  **INPUT**:  The food is good but the ambience in bad.

**OUTPUT**: The**/NASP** food**/ASP** is**/NASP** good**/NASP** but**/NASP** ambience**/AS** is**/NASP** bad**/NASP**

**./NASP**

And I want to cluster them into ‘n’ clusters.

INPUT:    :  [ food, Biriyani, fish, song, staff, waiting, theme, service]

OUTPUT :  [food, Biriyani, fish], [staff], [theme, song], [waiting, service],

I hope this project will complete on within 5-10 days.

If you want the result of this project, I will share after 10 days.

Thanks ☺