Data Science for Business 2nd Assignment (Web Scrabbing and Text classification)

Katesopon Kunpanperng 60070127

The task is to scrabbing text data about News Article in Prof's website, storing it in txt file and do a classification task to see which article is in which category

Part 1 : Data Collection

First just import all (really all of it) package used in this notebook.

In [51]:
```
# Data Collection
import requests
import time
import pandas as pd
import numpy as np
import re
from bs4 import BeautifulSoup
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import ComplementNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score

import nltk
from nltk.stem.porter import PorterStemmer
nltk.download('wordnet')

import matplotlib.pylab as plt
%matplotlib inline
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\theka\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Collecting all month link using original link and beautifulsoup scrab

In [2]:
```
# use as a link for access to month and article
link = "http://www.it.kmitl.ac.th/~teerapong/news_archive/"

page = requests.get("http://www.it.kmitl.ac.th/~teerapong/news_archive/index.html")
soup = BeautifulSoup(page.content, "html.parser")
```

In [3]:
```
mlist = []
for i in soup.find_all('a', href=True):
    i = str(i).split('"')
    if i[1] == "#":
        break
    else:
        mlist.append(link+i[1])
# print first link and last link to confirm
print("First Link:", mlist[0]+"\n"+"Last Link:", mlist[-1])
```

```
First Link: http://www.it.kmitl.ac.th/~teerapong/news_archive/month-jan-2017.html
Last Link: http://www.it.kmitl.ac.th/~teerapong/news_archive/month-dec-2017.html
```

Collecting all category and store it in pandas series

In [4]:
```
# define new list to store category
catlist = []
for i in mlist:
    monthpage = requests.get(i)
    # access each month page
    monthsoup = BeautifulSoup(monthpage.content, "html.parser")
    for j in monthsoup.find_all(class_="category"):
        catlist.append(re.findall("[a-z]+", str(j))[5])
# save as Series for easier data manipulate
catlist = pd.Series(catlist)
catlist.head()
```

Out[4]:
```
0    technology
1      business
2    technology
3      business
4         sport
dtype: object
```

Lets take a look at category values

In [5]:
```
catlist.value_counts()
```

Out[5]:
```
sport         526
business      491
technology    391
td             53
dtype: int64
```

oops it got **"td"** which is false at first by storing data techniques but if it in website article category its actually mean N/A value or its mean that there is no article available there right now so I will remove it.

In [6]:
```
# first I will replace it with nan(np.nan)
catlist.replace("td", np.nan, inplace=True)

# second I will drop it (cause there is no meaning to keep it)
catlist = catlist.dropna()

# check if it still have null values
```

In [6]:
```
catlist.isnull().sum()
```

Out[6]:
```
0
```

In [7]:
```
catlist.value_counts()
```

Out[7]:
```
sport         526
business      491
technology    391
dtype: int64
```

In [8]:
```
print(526 + 491 + 391)
```

```
1408
```

In [9]:
```
df = pd.DataFrame({'category':catlist})
```

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

Out[10]:

	category
0	technology
1	business
2	technology
3	business
4	sport

In [11]:
```
# store article category to csv file in datastore
df.to_csv("datastore/cate.csv", index=False)
```

After finish collecting and storing article, now it time to access all article link and collect article text

Get every link to access article data

In [12]:
```
datalink = []
for i in mlist:
    monpage = requests.get(i)
    monsoup = BeautifulSoup(monpage.content, "html.parser")
    monsoup = monsoup.find(class_="table table-condensed table-striped")
    # find where it match condition in find all
    for j in monsoup.findAll('a', attrs={'href': re.compile(".html$")}):
        datalink.append(link+str(j.get('href')))
datalink[0:3]
```

Out[12]:
```
['http://www.it.kmitl.ac.th/~teerapong/news_archive/article-jan-0418.html',
 'http://www.it.kmitl.ac.th/~teerapong/news_archive/article-jan-0027.html',
 'http://www.it.kmitl.ac.th/~teerapong/news_archive/article-jan-0631.html']
```

In [13]:
```
# to make sure that there are same amount of data as category
len(datalink)
```

Out[13]:
```
1408
```

access website by datalink and store text data

In [ ]:
```
# open text file to write data
filex = open('datastore/article.txt', 'w', encoding='utf-8')

# run for loop in datalink to access to everylink
for i in datalink:
    pands = requests.get(i)
    pands = BeautifulSoup(pands.content, 'html.parser')
    # create new string value to store data
    eachline = ""

    # why not contain first and last ?
    # because first is when web tell you article title or just blank
    # and last is when web tell you to comment or something kinda that but I dont want it.
    for j in pands.find_all('p')[1:-1:1]:
        j = j.get_text()
        eachline += str(j)

    filex.writelines(str(eachline)+"\n")
    time.sleep(0.01) # using sleep because website not allow you to run for loop rapidly in a short time

filex.close()
```

to check if i got same amount of data

In [16]:
```
filex = open("datastore/article.txt", "r", encoding='utf-8')
f1 = filex.readlines()
filex.close()
print("Total amount of Article stored:",len(f1))
# clear output
f1 = ""
```

```
Total amount of Article stored: 1408
```

As you see we got exactly same amount of links as amount of category because the non-existing link didn't store in website anymore (compare to NA value in category).

Part 2 : Text Classification

Load Data from datastore and store it in Notebook

In [3]:
```
# Load Article document
file_1 = open("datastore/article.txt", "r", encoding='utf-8')
raw_documents = file_1.readlines()
file_1.close()
print("There are %d raw text documents." % len(raw_documents))
```

```
There are 1408 raw text documents.
```

In [4]:
```
raw_documents[2]
```

Out[4]:
```
'BT is offering customers free internet telephone calls if they sign up to broadband in December.The Christmas give-away entitl
es customers to free telephone calls anywhere in the UK via the internet. Users will need to use BT\'s internet telephony softw
```

```
are, known as BT Communicator, and have a microphone and speakers or headset on their PC. BT has launched the promotion to show
off the potential of a broadband connection to customers.People wanting to take advantage of the offer will need to be a BT Tog
ether fixed-line customer and will have to sign up to broadband online. The offer will be limited to the first 50,000 people wh
o sign up and there are limitations - the free calls do not include calls to mobiles, non-geographical numbers such as 0870, pr
emium numbers or international numbers. BT is keen to provide extra services to its broadband customers. "People already using
BT Communicator have found it by far the most convenient way of making a call if they are at their PC," said Andrew Burke, dire
ctor of value-added services at BT Retail. As more homes get high-speed access, providers are increasingly offering add-ons suc
h as cheap net calls. "Broadband and telephony are attractive to customers and BT wants to make sure it is in the first wave of
services," said Ian Fogg, an analyst with Jupiter Research. "BT Communicator had a quiet launch in the summer and now BT is wav
ing the flag a bit more for it," he added.BT has struggled to maintain its market share of broadband subscribers as more compet
itors enter the market. Reports say that BT has lost around 10% of market share over the last year, down from half of broadband
users to less than 40%. BT is hoping its latest offer can persuade more people to jump on the broadband bandwagon. It currently
has 1.3 million broadband subscribers.\n'
```

In [5]:
```
# Load Article Category file
category = pd.read_csv("datastore/cate.csv")
len(category)
```

Out[5]: 1408

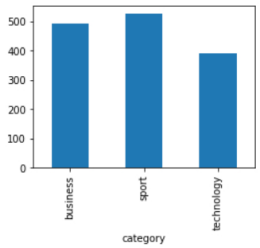
In [6]:
```
category.head()
```

Out[6]:

	category
0	technology
1	business
2	technology
3	business
4	sport

to know if the label is balance or imbalance I need to plot bar graph to see if it balance or not

In [7]:
```
fig = plt.figure(figsize=(4,3))
category.groupby('category').category.count().plot.bar(ylim=0)
plt.show()
```



From this bar plot im pretty sure that it a balance data so now i will do a train and test split

In [8]:
```
# I'm not splitting data because it better to try train test using cross validation
X = raw_documents
Y = category.category
```

before I go straight tokenize my data and want to define function that use to reduce a term into its canonical form (a bit more advance form of stemming) by using **Lemmatization**.

Define Lemmatization Function

use this function for being parameter in text tokenizer model

In [9]:
```
# define lemma tization function
def lemma_tokenizer(text):
    # use the standard scikit-learn tokenizer first
    standard_tokenizer = CountVectorizer().build_tokenizer()
    tokens = standard_tokenizer(text)
    # then use NLTK to perform lemmatisation on each token
    lemmatizer = nltk.stem.WordNetLemmatizer()
    lemma_tokens = []
    for token in tokens:
        lemma_tokens.append( lemmatizer.lemmatize(token) )
    return lemma_tokens
```

Note

I'm not do a train test split becuase I will use **cross-validation** to testing data.

Count Vectorizer with binary vectorization

In [10]:
```
count_vectorizer = CountVectorizer(binary=True, stop_words='english', min_df = 2, tokenizer=lemma_tokenizer)
count_vectorizer.fit(X)

X_binary = count_vectorizer.transform(X)
# Im apply dense because dense data is required if im not converting X to dense it will error when applied in model.
X_binary = X_binary.todense()
```

```
C:\Users\theka\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\feature_extraction\text.py:301: UserWarning: Your st
op_words may be inconsistent with your preprocessing. Tokenizing the stop words generated tokens ['ha', 'le', 'u', 'wa'] not in
stop_words.
  'stop_words.' % sorted(inconsistent))
```

TF-IDF Vectorizer

In [41]:
```
tfidf_vectorizer = TfidfVectorizer(stop_words='english', min_df = 2, tokenizer=lemma_tokenizer)
tfidf_vectorizer.fit(X)

X_tfidf = tfidf_vectorizer.transform(X)
X_tfidf = X_tfidf.todense()
```

```
C:\Users\theka\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\feature_extraction\text.py:301: UserWarning: Your st
op_words may be inconsistent with your preprocessing. Tokenizing the stop words generated tokens ['ha', 'le', 'u', 'wa'] not in
stop_words.
```

```
'stop_words.' % sorted(inconsistent))
```

Applying a model

I'm using 4 model in this time and then apply these 4 model seperately on each tokenization model.

In [48]:
```
model_gnb = GaussianNB() # Gaussian Naive-Bayes
model_mltb = MultinomialNB() # Multinomial Naive-Bayes
model_comple = ComplementNB() # Complement Naive-Bayes
knn = KNeighborsClassifier(n_neighbors=4) # K-Nearest Neighbors
models = [model_gnb, model_mltb, model_comple, knn]
```

I set `n_neighbors = 4` because I test between (3-20) in binary model to see which one got highest scores on cross-validate and 4 got the highest scores(even it not that high).

Cross Validate on Binary-Vectorizer

In [50]:
```
for i in models:
    model = i
    scores = cross_val_score(model, X_binary, Y, cv=10)
    print("Score from Binary vectorizer and", i.__class__.__name__, "model is:", scores.mean())
    print("Score Standard Deviation is", str(scores.std())+"\n")
```

```
Score from Binary vectorizer and GaussianNB model is: 0.9680232533424024
Score Standard Deviation is 0.010728621130922348

Score from Binary vectorizer and MultinomialNB model is: 0.977962888175654
Score Standard Deviation is 0.012155761430401759

Score from Binary vectorizer and ComplementNB model is: 0.9793916721576297
Score Standard Deviation is 0.012916856878405278

Score from Binary vectorizer and KNeighborsClassifier model is: 0.7180970802247398
Score Standard Deviation is 0.03532234877423803
```

Cross Validate on TF-IDF

In [49]:
```
for i in models:
    model = i
    scores = cross_val_score(model, X_tfidf, Y, cv=10)
    print("Score from TF-IDF vectorizer and", i.__class__.__name__, "model is:", scores.mean())
    print("Score Standard Deviation is", str(scores.std())+"\n")
```

```
Score from TF-IDF vectorizer and GaussianNB model is: 0.9431494746388361
Score Standard Deviation is 0.027652661047592016

Score from TF-IDF vectorizer and MultinomialNB model is: 0.9765294634443571
Score Standard Deviation is 0.015327113120478731

Score from TF-IDF vectorizer and ComplementNB model is: 0.9808250968889265
Score Standard Deviation is 0.014615670523954197

Score from TF-IDF vectorizer and KNeighborsClassifier model is: 0.9609112164431313
Score Standard Deviation is 0.0111661212985854
```

Conclusion

After trying many models and two of tokenizer I better say that I would like to use **TF-IDF tokenizer** and **Multinomial Naive-Bayes Algorithm** because I want to avoid some word that appear in every document and Multinomial is work well for multinomial distributed data and TF-IDF and why I don't use ComplementNB even it got higher score is becuase ComplementNB work well with imbalance data. So that why I think MultinomialNB is **suitable** to use in this task.