In [0]:

```
## Reading Electrical Papers
```

In [1]:

```
## Read the math papers from 2016 to 2018 and store it in a file
import urllib
url = 'http://export.arxiv.org/oai2?verb=ListRecords&set=eess&from=2016-01-01&until=2018-11-31&metadataPrefi
x=arXiv'
data = urllib.request.urlopen(url).read()

e = open('ele1', 'wb')
e.write(data)
```

Out[1]:

```
2082009
```

In [0]:

```
## Extract the title and abstract from papers - Read from finance1 to finance2
!apt-get install xml-twig-tools
!xml_grep 'title|abstract' ele1  > ele2.txt
```

In [0]:

```
## Remove Junk lines , here we remove first 3 lines and last 3 lines which are not necessary
!cat ele2.txt | tail -n +4 | head -n -3 > ele3.txt
```

In [0]:

```
## Reading packages for Text classification
from sklearn import model_selection, preprocessing, linear_model, naive_bayes, metrics, svm
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn import decomposition, ensemble

import pandas, numpy, string
from keras.preprocessing import text, sequence
from keras import layers, models, optimizers
from nltk import word_tokenize
from nltk.corpus import stopwords
import sklearn
#import sklearn_crfsuite
#from sklearn_crfsuite import scorers
#from sklearn_crfsuite import metrics
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics import accuracy_score
from sklearn import metrics
```

In [7]:

```
## Stopwords import and removal
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
stopwords = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

In [8]:

```
# load the dataset # dataset contains combined labels and text from all training papers
data = open('labeled_sentences (1).txt').read()[:-2]
labels, texts = [], []
for i, line in enumerate(data.split("\n")):
    content = line.split()
    #print(content)
    labels.append(content[0])
    filtered_sentence = [w.lower() for w in content[1:] if not w in stopwords]
    texts.append(filtered_sentence)

# create a dataframe using texts and lables
trainDF = pandas.DataFrame()
trainDF['text'] = texts
trainDF['label'] = labels
print(trainDF['label'].unique())
trainDF.head(2)
```

```
['MISC' 'AIMX' 'OWNX' 'CONT' 'BASE']
```

Out[8]:

	text	label
0	[minimum, description, length, principle, onli...	MISC
1	[underlying, model, class, discrete,, total, e...	MISC

In [0]:

```
## Used the obtained dataset for training
train_x, valid1_x, train_y, valid1_y = model_selection.train_test_split(trainDF['text'], trainDF['label'],te
st_size=0)
```

In [10]:

```
## Convert from list to string
tempp = []

for item in train_x:
    tempp.append(" ".join(item))
#print(len(train_x))

#tempp1 =[]
#for item1 in valid_x:
    #tempp1.append(" ".join(item1))

#print(len(tempp1))

temp =[]
temp_len=0
for item2 in texts:
    temp.append(" ".join(item2))
    temp_len = temp_len+len(texts)
print(len(temp))
print(temp_len)
print(type(temp))
```

```
18627
346965129
<class 'list'>
```

In [0]:

```
# create a count vectorizer object
count_vect = CountVectorizer(analyzer='word', token_pattern=r'\w{1,}')
count_vect.fit(temp)

# transform the training and validation data using count vectorizer object
xtrain_count =  count_vect.transform(tempp)
```

In [0]:

```
## Create a classifier
import csv
trainDF2 = pandas.DataFrame()

def train_model(classifier, feature_vector_train, label, feature_vector_valid, is_neural_net=False):
    # fit the training dataset on the classifier
    #std_clf = make_pipeline(StandardScaler(with_mean=False), TruncatedSVD(100), MultinominalNB())
    #std_clf.fit(feature_vector_train, label)
    classifier.fit(feature_vector_train, label)

    # predict the labels on validation dataset
    #predictions = classifier.predict(feature_vector_valid)
    predictions = classifier.predict(feature_vector_valid)
    return predictions
    #tt = classifier.predict(feature_vector_valid)
    #labels3 = classifier.predict(feature_vector_valid)

    #trainDF2['labels'] = labels3
    #trainDF2['text']= valid_x
    #print(trainDF2)
```

In [15]:

```
## Read title and abstracts and loop through them
import re
global_list = []
title_list =[]

test = open("ele3.txt",'r').read().split("</abstract>")
#print(test[1])
for idx,i in enumerate(test):
  title = re.findall(r"(?<=<title>).*(?=</title>)",i.replace("\n",""))
  #print(title)
  abstract = re.findall(r"(?<=<abstract>).*",i.replace("\n",""))
  #print(abstract[0].replace("\n",""))
  nlist = re.split(r"(?:(?<=[^i]\.)|\.(?=[^e]))",abstract[0].replace('"',"").replace('\n',''))
  #temp_abs = re.sub(r"((?<=[^i]\.)|\.(?=[^e]))","\n",abstract[0])
  #print(abstract)
  #temp_str = temp_abs.split("\n")
  #print(temp_str[0])
  #print(nlist[1])
  global_list.append(nlist)
  title_list.append(title)
  #print(global_list)

  if idx >50:
    #print(global_list)
    break
  #print(abstract[0])
  #nlist = re.split(r"(?:(?<=[^i]\.)|\.(?=[^e]))",str(abstract))

  #print(nlist[1])

  #tempp1 =[]
  '''
  for idx, item1 in enumerate(nlist):

    if idx > 1 :
      break;
      print(item1)
      tempp1.append(" ".join(item1))
    #print(tempp1)

    xvalid_count =  count_vect.transform(tempp1)
    for item in nlist:
      print(item)
      valid_x = item
      #accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_count, train_y, xvalid_count)

  '''
  #print(global_list[0])
  #print(global_list[1])
  #print(global_list[2])
  #for idx, item1 in enumerate(global_list) :
  #  if idx > 1:
  #    break
  #  print(item1)
    #tempp1.append(" ".join(item1))
    #xvalid_count =  count_vect.transform(tempp1)
    #accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_count, train_y, xvalid_count)
```

```
/usr/lib/python3.6/re.py:212: FutureWarning: split() requires a non-empty pattern match.
  return _compile(pattern, flags).split(string, maxsplit)
```

In [16]:

```
## Print triples from data

#print(global_list[1])
for idx, (item, title) in enumerate(zip(global_list, title_list)):



  #print(item)
  valid_x = item
  xvalid_count =  count_vect.transform(valid_x)
  accuracy = train_model(linear_model.LogisticRegression(), xtrain_count, train_y, xvalid_count)
  #print("\n\n")
  if idx>1:
    break

  title_id = hash(str(title))
  abstract_id = hash(str(item))
  line1 = "<https://w3id.org/skg/articles/" + str(title_id) + "> <http://xmlns.com/foaf/0.1/name>" + '"' + "
 ".join(title) + '"' +"."
  line2 = "<https://w3id.org/skg/articles/" + str(title_id) + "> <http://purl.org/dc/terms/abstract> <http:/
/purl.org/dc/terms/abstract/" + str(abstract_id)+ ">"
  line3 = "<https://w3id.org/skg/articles/" + str(abstract_id) +"><http://purl.org/dc/terms/abstract/text>"
+ '"' + " ".join(item) + '"'
  print(line1,line2,line3,sep ="\n")
  for acc,element in zip(accuracy,item):
    print('<http://purl.org/dc/terms/abstract/{} > "{}"'.format(acc, element))
    #line4 = ("<http://purl.org/dc/terms/abstract/" + str(acc) + ">" + '"' + str(element) + '"' )
```

```
<https://w3id.org/skg/articles/-6502291845625822428> <http://xmlns.com/foaf/0.1/name>"Warping P
eirce Quincuncial Panoramas".
<https://w3id.org/skg/articles/-6502291845625822428> <http://purl.org/dc/terms/abstract> <http:
//purl.org/dc/terms/abstract/-6245615208384810736>
<https://w3id.org/skg/articles/-6245615208384810736><http://purl.org/dc/terms/abstract/text>"
The Peirce quincuncial projection is a mapping of the surface of a sphere tothe interior of a s
quare  It is a conformal map except for four points on theequator  These points of non-conforma
lity cause significant artifacts inphotographic applications  In this paper, we propose an algo
rithm anduser-interface to mitigate these artifacts  Moreover, in order to facilitate aninterac
tive user-interface, we present a fast algorithm for calculating thePeirce quincuncial projecti
on of spherical imagery  We then promote the Peircequincuncial projection as a viable alternati
ve to the more popularstereographic projection in some scenarios."
<http://purl.org/dc/terms/abstract/OWNX > "  The Peirce quincuncial projection is a mapping of
the surface of a sphere tothe interior of a square"
<http://purl.org/dc/terms/abstract/MISC > " It is a conformal map except for four points on the
equator"
<http://purl.org/dc/terms/abstract/MISC > " These points of non-conformality cause significant
artifacts inphotographic applications"
<http://purl.org/dc/terms/abstract/OWNX > " In this paper, we propose an algorithm anduser-inte
rface to mitigate these artifacts"
<http://purl.org/dc/terms/abstract/OWNX > " Moreover, in order to facilitate aninteractive user
-interface, we present a fast algorithm for calculating thePeirce quincuncial projection of sph
erical imagery"
<http://purl.org/dc/terms/abstract/OWNX > " We then promote the Peircequincuncial projection as
 a viable alternative to the more popularstereographic projection in some scenarios."
<https://w3id.org/skg/articles/-2823227690419730964> <http://xmlns.com/foaf/0.1/name>"Tracking
Tetrahymena Pyriformis Cells using Decision Trees".
<https://w3id.org/skg/articles/-2823227690419730964> <http://purl.org/dc/terms/abstract> <http:
//purl.org/dc/terms/abstract/7463497532546105089>
<https://w3id.org/skg/articles/7463497532546105089><http://purl.org/dc/terms/abstract/text>"  M
atching cells over time has long been the most difficult step in celltracking  In this paper, w
e approach this problem by recasting it as aclassification problem  We construct a feature set
for each cell, and compute afeature difference vector between a cell in the current frame and a
 cell in aprevious frame  Then we determine whether the two cells represent the same cellover t
ime by training decision trees as our binary classifiers  With the outputof decision trees, we
are able to formulate an assignment problem for our cellassociation task and solve it using a m
odified version of the Hungarianalgorithm."
<http://purl.org/dc/terms/abstract/MISC > "  Matching cells over time has long been the most di
fficult step in celltracking"
<http://purl.org/dc/terms/abstract/AIMX > " In this paper, we approach this problem by recastin
g it as aclassification problem"
<http://purl.org/dc/terms/abstract/OWNX > " We construct a feature set for each cell, and compu
te afeature difference vector between a cell in the current frame and a cell in aprevious frame
"
<http://purl.org/dc/terms/abstract/MISC > " Then we determine whether the two cells represent t
he same cellover time by training decision trees as our binary classifiers"
<http://purl.org/dc/terms/abstract/OWNX > " With the outputof decision trees, we are able to fo
rmulate an assignment problem for our cellassociation task and solve it using a modified versio
n of the Hungarianalgorithm."
```