In [0]:

```
## Reading Computer Science Papers
```

In [1]:

```
## Read the computer science papers from 2016 to 2018 and store it in a file
import urllib
url = 'http://export.arxiv.org/oai2?verb=ListRecords&set=cs&from=2015-11-01&until=2018-11-31&metadataPrefix=
arXiv'
data = urllib.request.urlopen(url).read()

f = open('computer1', 'wb')
f.write(data)
```

Out[1]:

```
2008844
```

In [0]:

```
## Extract the title and abstract from papers - Read from finance1 to finance2
!xml_grep 'title|abstract' computer1  > computer2.txt
```

In [0]:

```
## Remove Junk lines , here we remove first 3 lines and last 3 lines which are not necessary
!cat computer2.txt | tail -n +4 | head -n -3 > computer3.txt
```

In [5]:

```
## 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
```

```
Using TensorFlow backend.
```

In [6]:

```
## 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]   Package stopwords is already up-to-date!
```

In [7]:

```
# 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[7]:

	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 [9]:

```
## 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))
```

```
19162
367182244
<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 [12]:

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

test = open("computer3.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 [13]:

```
## 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>2:
    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/-2867593985518337823> <http://xmlns.com/foaf/0.1/name>"Pseudo-ra
ndom Puncturing: A Technique to Lower the Error Floor of Turbo  Codes".
<https://w3id.org/skg/articles/-2867593985518337823> <http://purl.org/dc/terms/abstract> <http:
//purl.org/dc/terms/abstract/-4392215306747285796>
<https://w3id.org/skg/articles/-4392215306747285796><http://purl.org/dc/terms/abstract/text>"
It has been observed that particular rate-1/2 partially systematic parallelconcatenated convolu
tional codes (PCCCs) can achieve a lower error floor thanthat of their rate-1/3 parent codes  N
evertheless, good puncturing patterns canonly be identified by means of an exhaustive search, w
hilst convergence towardslow bit error probabilities can be problematic when the systematic out
put of arate-1/2 partially systematic PCCC is heavily punctured  In this paper, wepresent and s
tudy a family of rate-1/2 partially systematic PCCCs, which wecall pseudo-randomly punctured co
des  We evaluate their bit error rateperformance and we show that they always yield a lower err
or floor than that oftheir rate-1/3 parent codes  Furthermore, we compare analytic results tosi
mulations and we demonstrate that their performance converges towards theerror floor region, ow
ning to the moderate puncturing of their systematicoutput  Consequently, we propose pseudo-rand
om puncturing as a means ofimproving the bandwidth efficiency of a PCCC and simultaneously lowe
ring itserror floor."
<http://purl.org/dc/terms/abstract/MISC > "  It has been observed that particular rate-1/2 part
ially systematic parallelconcatenated convolutional codes (PCCCs) can achieve a lower error flo
or thanthat of their rate-1/3 parent codes"
<http://purl.org/dc/terms/abstract/MISC > " Nevertheless, good puncturing patterns canonly be i
dentified by means of an exhaustive search, whilst convergence towardslow bit error probabiliti
es can be problematic when the systematic output of arate-1/2 partially systematic PCCC is heav
ily punctured"
<http://purl.org/dc/terms/abstract/AIMX > " In this paper, wepresent and study a family of rate
-1/2 partially systematic PCCCs, which wecall pseudo-randomly punctured codes"
<http://purl.org/dc/terms/abstract/OWNX > " We evaluate their bit error rateperformance and we
show that they always yield a lower error floor than that oftheir rate-1/3 parent codes"
<http://purl.org/dc/terms/abstract/OWNX > " Furthermore, we compare analytic results tosimulati
ons and we demonstrate that their performance converges towards theerror floor region, owning t
o the moderate puncturing of their systematicoutput"
<http://purl.org/dc/terms/abstract/OWNX > " Consequently, we propose pseudo-random puncturing a
s a means ofimproving the bandwidth efficiency of a PCCC and simultaneously lowering itserror f
loor."
<https://w3id.org/skg/articles/3544482385407756047> <http://xmlns.com/foaf/0.1/name>"A Low Comp
lexity Algorithm and Architecture for Systematic Encoding of  Hermitian Codes".
<https://w3id.org/skg/articles/3544482385407756047> <http://purl.org/dc/terms/abstract> <http:/
/purl.org/dc/terms/abstract/5993190164001199460>
<https://w3id.org/skg/articles/5993190164001199460><http://purl.org/dc/terms/abstract/text>"  W
e present an algorithm for systematic encoding of Hermitian codes  For aHermitian code defined
over GF(q^2), the proposed algorithm achieves a run timecomplexity of O(q^2) and is suitable fo
r VLSI implementation  The encoderarchitecture uses as main blocks q varying-rate Reed-Solomon
encoders andachieves a space complexity of O(q^2) in terms of finite field multipliers andmemor
y elements."
<http://purl.org/dc/terms/abstract/OWNX > "  We present an algorithm for systematic encoding of
 Hermitian codes"
<http://purl.org/dc/terms/abstract/OWNX > " For aHermitian code defined over GF(q^2), the propo
sed algorithm achieves a run timecomplexity of O(q^2) and is suitable for VLSI implementation"
<http://purl.org/dc/terms/abstract/MISC > " The encoderarchitecture uses as main blocks q varyi
```

ng-rate Reed-Solomon encoders andachieves a space complexity of O(q^2) in terms of finite field
 multipliers andmemory elements."
<https://w3id.org/skg/articles/-9205124697652296151> <http://xmlns.com/foaf/0.1/name>"Learning
from compressed observations".
<https://w3id.org/skg/articles/-9205124697652296151> <http://purl.org/dc/terms/abstract> <http:
//purl.org/dc/terms/abstract/-8696850721371301559>
<https://w3id.org/skg/articles/-8696850721371301559><http://purl.org/dc/terms/abstract/text>"
The problem of statistical learning is to construct a predictor of a randomvariable Y as a fu
nction of a related random variable X on the basis of ani i d training sample from the joint
 distribution of (X,Y) Allowablepredictors are drawn from some specified class, and the goal
 is to approachasymptotically the performance (expected loss) of the best predictor in theclass
  We consider the setting in which one has perfect observation of the$X$-part of the sample, wh
ile the Y-part has to be communicated at somefinite bit rate The encoding of the Y-values
is allowed to depend on the$X$-values Under suitable regularity conditions on the admissible p
redictors,the underlying family of probability distributions and the loss function, wegive an i
nformation-theoretic characterization of achievable predictorperformance in terms of conditiona
l distortion-rate functions The ideas areillustrated on the example of nonparametric regressio
n in Gaussian noise."
<http://purl.org/dc/terms/abstract/OWNX > " The problem of statistical learning is to construc
t a predictor of a randomvariable Y as a function of a related random variable X on the bas
is of ani"
<http://purl.org/dc/terms/abstract/MISC > "i"
<http://purl.org/dc/terms/abstract/MISC > "d"
<http://purl.org/dc/terms/abstract/MISC > " training sample from the joint distribution of $(X,
Y)$"
<http://purl.org/dc/terms/abstract/OWNX > " Allowablepredictors are drawn from some specified c
lass, and the goal is to approachasymptotically the performance (expected loss) of the best pre
dictor in theclass"
<http://purl.org/dc/terms/abstract/OWNX > " We consider the setting in which one has perfect ob
servation of the$X$-part of the sample, while the Y-part has to be communicated at somefinite
 bit rate"
<http://purl.org/dc/terms/abstract/OWNX > " The encoding of the Y-values is allowed to depend
 on the$X$-values"
<http://purl.org/dc/terms/abstract/OWNX > " Under suitable regularity conditions on the admissi
ble predictors,the underlying family of probability distributions and the loss function, wegive
 an information-theoretic characterization of achievable predictorperformance in terms of condi
tional distortion-rate functions"
<http://purl.org/dc/terms/abstract/OWNX > " The ideas areillustrated on the example of nonparam
etric regression in Gaussian noise."