In [0]:

```
## Reading Biology Papers
```

In [1]:

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

bio = open('bio1', 'wb')
bio.write(data)
```

Out[1]:

```
2253178
```

In [0]:

```
## Extract the title and abstract from papers - Read from finance1 to finance2

!xml_grep 'title|abstract' bio1  > bio2.txt
```

In [0]:

```
## Remove Junk lines , here we remove first 3 lines and last 3 lines which are not necessary
!cat bio2.txt | tail -n +4 | head -n -3 > bio3.txt
```

In [4]:

```
## 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 [5]:

```
## 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 [6]:

```
# 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[6]:

	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 [8]:

```
## 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 [12]:

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

test = open("bio3.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>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/-4079202618155772900> <http://xmlns.com/foaf/0.1/name>"Primordia
l Evolution in the Finitary Process Soup".
<https://w3id.org/skg/articles/-4079202618155772900> <http://purl.org/dc/terms/abstract> <http:
//purl.org/dc/terms/abstract/5052414978280071745>
<https://w3id.org/skg/articles/5052414978280071745><http://purl.org/dc/terms/abstract/text>"  A
 general and basic model of primordial evolution--a soup of reactingfinitary and discrete proce
sses--is employed to identify and analyzefundamental mechanisms that generate and maintain comp
lex structures inprebiotic systems  The processes--$\epsilon$-machines as defined incomputation
al mechanics--and their interaction networks both provide welldefined notions of structure  Thi
s enables us to quantitatively demonstratehierarchical self-organization in the soup in terms o
f complexity  We foundthat replicating processes evolve the strategy of successively building h
igherlevels of organization by autocatalysis  Moreover, this is facilitated by localcomponents 
that have low structural complexity, but high generality  In effect,the finitary process soup s
pontaneously evolves a selection pressure thatfavors such components  In light of the finitary 
process soup's generality,these results suggest a fundamental law of hierarchical systems: glob
alcomplexity requires local simplicity."
<http://purl.org/dc/terms/abstract/MISC > "  A general and basic model of primordial evolution-
-a soup of reactingfinitary and discrete processes--is employed to identify and analyzefundamen
tal mechanisms that generate and maintain complex structures inprebiotic systems"
<http://purl.org/dc/terms/abstract/MISC > " The processes--$\epsilon$-machines as defined incom
putational mechanics--and their interaction networks both provide welldefined notions of struct
ure"
<http://purl.org/dc/terms/abstract/OWNX > " This enables us to quantitatively demonstratehierar
chical self-organization in the soup in terms of complexity"
<http://purl.org/dc/terms/abstract/OWNX > " We foundthat replicating processes evolve the strat
egy of successively building higherlevels of organization by autocatalysis"
<http://purl.org/dc/terms/abstract/MISC > " Moreover, this is facilitated by localcomponents th
at have low structural complexity, but high generality"
<http://purl.org/dc/terms/abstract/MISC > " In effect,the finitary process soup spontaneously e
volves a selection pressure thatfavors such components"
<http://purl.org/dc/terms/abstract/OWNX > " In light of the finitary process soup's generality,
these results suggest a fundamental law of hierarchical systems: globalcomplexity requires loca
l simplicity."
<https://w3id.org/skg/articles/1672551164774122412> <http://xmlns.com/foaf/0.1/name>"Anisotropi
c probabilistic cellular automaton for a predator-prey system".
<https://w3id.org/skg/articles/1672551164774122412> <http://purl.org/dc/terms/abstract> <http:/
/purl.org/dc/terms/abstract/-3221819674834026290>
<https://w3id.org/skg/articles/-3221819674834026290><http://purl.org/dc/terms/abstract/text>"  
We consider a probabilistic cellular automaton to analyze the stochasticdynamics of a predator-
prey system  The local rules are Markovian and are basedin the Lotka-Volterra model  The indivi
duals of each species reside on thesites of a lattice and interact with an unsymmetrical neighb
orhood  We look forthe effect of the space anisotropy in the characterization of the oscillatio
nsof the species population densities  Our study of the probabilistic cellularautomaton is base
d on simple and pair mean-field approximations and explicitlytakes into account spatial anisotr
opy."
<http://purl.org/dc/terms/abstract/OWNX > "  We consider a probabilistic cellular automaton to 
analyze the stochasticdynamics of a predator-prey system"
<http://purl.org/dc/terms/abstract/MISC > " The local rules are Markovian and are basedin the L
otka-Volterra model"
<http://purl.org/dc/terms/abstract/MISC > " The individuals of each species reside on thesites 
of a lattice and interact with an unsymmetrical neighborhood"
<http://purl.org/dc/terms/abstract/OWNX > " We look forthe effect of the space anisotropy in th
e characterization of the oscillationsof the species population densities"
<http://purl.org/dc/terms/abstract/OWNX > " Our study of the probabilistic cellularautomaton is
 based on simple and pair mean-field approximations and explicitlytakes into account spatial an
isotropy."
```