```
import numpy as np
import matplotlib.pyplot as plt


import nltk
nltk.download('averaged_perceptron_tagger')
```

```
    [nltk_data] Downloading package averaged_perceptron_tagger to
    [nltk_data]     /root/nltk_data...
    [nltk_data]   Package averaged_perceptron_tagger is already up-to-
    [nltk_data]       date!
    True
```

```
from nltk.tag.perceptron import PerceptronTagger
tagger = PerceptronTagger()
```

Example on how to classify a sentece. Here the split() function splits by ' ' character.

```
# Example on how to
tagger.tag("How".split())
```

```
    [('How', 'WRB')]
```

Create a token class

```
class Token:
  spelling = None
  POSTag = None

  def __init__ (self, spelling, tag):
    self.spelling = spelling.lower()
    if(tag == ""):
      print("Non valid tag")
    self.POSTag = tag



class Sentence:
  sentence = None

  def __init__ (self, ListOfTokens):
    self.sentence = ListOfTokens

  def ToString(self):
    sentenceString = ""
    for token in self.sentence:
      sentenceString += str(token.spelling) + " "
    return sentenceString



class DataSet:
  dataSet = None

  def __init__ (self, ListOfSentences):
    self.dataSet = ListOfSentences

  def split(self, fraction):
    dataSetSize = len(self.dataSet)
    indexOfSplit = int(fraction * dataSetSize)

    print(indexOfSplit)

    trainingDataSet = DataSet(self.dataSet[:indexOfSplit])
    testingDataSet = DataSet(self.dataSet[indexOfSplit:])

    return trainingDataSet, testingDataSet
```

2.)

- Upload the files
- Re-format the sentences
- All words to lower case

```
#from google.colab import files
#uploaded = files.upload()


completeDataSetData = None

with open("BrownCorpus.txt") as file:
  completeDataSetData = file.readlines()
completeDataSetData = np.array(completeDataSetData)

print("File completly read. " + str(completeDataSetData.size) + " senteces loaded.")


print("Convert the sentences to tokens, senteces and a dataSet")

tokenList = []
sentenceList = []

# loop over each element in the completeDataSet
for sentenceData in completeDataSetData:
  # split the sentece in tokens, use space as delimiter
  splitSentence = sentenceData.split()
  sentenceToken = []
  for subToken in splitSentence:
    splitToken = subToken.split("_")
    spelling = splitToken[0]
    if ('|' in spelling):
      continue
    token = Token(splitToken[0], splitToken[1])
    tokenList.append(token)
    sentenceToken.append(token)
  sentence = Sentence(sentenceToken)
  sentenceList.append(sentence)

# initiate the DataSet
completeDataSet = DataSet(sentenceList)

# Convert all tags from Brown to Universal
BrownToUniversalTagData = []

with open("BrownToUniversalTagMap.txt") as file:
  for line in file:
    # replace \t\t with \t
    processedLine = line.replace("\t\t", "\t")
    processedLine = processedLine.split('\t')
    BrownTag = processedLine[0]
    UniversalTag = processedLine[1].strip()
    BrownToUniversalTagData.append([BrownTag, UniversalTag ])

# convert to Numpy Array
BrownToUniversalTagData = np.array(BrownToUniversalTagData)

# Convert the BrownTags of the Tokens into Universal Tags
for token in tokenList:
  tokenBrownTag = token.POSTag
  tokenBrownTagIndex = np.where(BrownToUniversalTagData[:, 0] == tokenBrownTag)

  # get the index of
  try:
    tokenUniversalTag = BrownToUniversalTagData[tokenBrownTagIndex, 1][0][0]
  except:
    print(token.spelling)
    print(token.POSTag)
  # tokenUniversalTag = BrownToUniversalTagData[tokenBrownTagIndex, 1]
  token.POSTag = tokenUniversalTag

print("Conversion of Brown to Universal tag for all tokens done!")
```

```
    File completly read. 57066 senteces loaded.
    Convert the sentences to tokens, senteces and a dataSet
    ]
    .|SB01:1
    .
    .|SC01:1
    .
    .|SD01:1
    .
    .|SE01:1
    .
    .|SF01:1
    .
    .|SG01:1
    .
    .|SH01:1
```

```
.
.|SJ01:1
.
.|SK01:1
.
.|SL01:1
.
.|SM01:1
.
.|SN01:1
]
.|SP01:1
.
.|SR01:1
Conversion of Brown to Universal tag for all tokens done!
```

```
import random
randS = random.randint(0, len(sentenceList))
sentence = sentenceList[randS]
randT = random.randint(0, len(sentence.sentence))

print(sentence.ToString())
print("Spelling:"+ '\t' + sentence.sentence[randT].spelling)
print("Tag:" + '\t'+ sentence.sentence[randT].POSTag)
```

```
he walked rapidly along the buildings scanning their facades : one was a club -- that was out ;
Spelling:       rapidly
Tag:    ADV
```

3.)

- Split the data into a training and test set -> Exacetly like in part a) to make comparison

```
## Split the data set by a fraction
fraction = 0.8
trainingDataSet, testingDataSet = completeDataSet.split(fraction)

print(len(completeDataSet.dataSet))
print(len(trainingDataSet.dataSet))
print(len(testingDataSet.dataSet))
```

```
45652
57066
45652
11414
```

4.)

- Tokenize the test
- -> Already done, with the implementation above

5.)

- Run the perceptron tagger over the test set

--> Classify every token in the test set

```
resultsWithAssignedTags = []
from nltk.tag.mapping import map_tag
nltk.download('universal_tagset')

# loop over each sentence
for sentence in testingDataSet.dataSet:
#sentence =  testingDataSet.dataSet[110]
  #print(sentence.ToString() + '\n\n')

  for token in sentence.sentence:
    spelling = str(token.spelling)

    trueTag = token.POSTag

    assignedTag = tagger.tag(spelling.split())[0][1]
    mappedTag = map_tag('en-ptb', 'universal', assignedTag)

    success = int(mappedTag == trueTag)

    results = [spelling, trueTag, mappedTag, int(success)]
    resultsWithAssignedTags.append(results)

resultsWithAssignedTags = np.array(resultsWithAssignedTags)
```

```
[nltk_data] Downloading package universal_tagset to /root/nltk_data...
[nltk_data]   Package universal_tagset is already up-to-date!
```

7.) -> Compute the accuracy of the perceptron tagger.

```
# take sum of all values in the 4th column of the resultsWithAssignedTags array
successSum = 0
numberOfEntries = resultsWithAssignedTags.shape[0]

for i in range(numberOfEntries):
  successSum += int(resultsWithAssignedTags[i, 3])

print(successSum)
print(numberOfEntries)

accuracy = successSum / numberOfEntries
print(accuracy)
```

```
145080
176041
0.8241261978743588
```