UniversityMichigan

April 6, 2018

1 Clasification example for the University of Michigan

The purpose of this example is to demonstrate (on a first attempt) my ability to collect, work with, clean, and classify a set of text documents applied to a text problem in which the idea is to find the grade associated to different assays based on the lexical-syntactical structure of documents.

Example features:

1. All the code was implemented in Python 2.7 https://www.python.org/
2. The dataset used is part of a essay collection provide by the University of Michigan (econ_a1) which contains 300 Word documents, each one with a label from 1 to 5 that represent the score assigned to that document.
3. The Python packages required to run the programs are the following:
   - Jupyter notebook (Python interactive prompt) http://jupyter.org/index.html
   - PyWin32 (pre-processing Word documents) https://pypi.python.org/pypi/pywin32
   - Numpy (classification) http://www.numpy.org/
   - Scikit-learn (classification) http://scikit-learn.org/stable/
   - NLTK (NLP techniques) https://www.nltk.org/
   - CLips pattern (NLP techniques) https://www.clips.uantwerpen.be/pattern
   - Matplotlib (visualization) https://matplotlib.org/
   - WordCloud (visualization) https://pypi.python.org/pypi/wordcloud

1.1 Pre-processing

The first step performed to classify the essays is to clean and transform the Word documents. The following shows how to achieve this task:

- Collect the list of the Word documents contained in the dataset.

```
In [22]: from os import listdir
         from os.path import isfile, join
         FILEPATH  = "C:/Users/EstebanCj/Desktop/michigan/econ_a1/essays/"
         fileNames = [f for f in listdir(FILEPATH) if isfile(join(FILEPATH, f))]
         #First 10 files in the list
         print fileNames[:10]
```

```
['econ_a1_essay1.docx', 'econ_a1_essay10.docx', 'econ_a1_essay100.docx', 'econ_a1_essay101.docx'
```

- Obtain the scores (classification labels) associated to each Word document

```
In [2]: import json
        FILEPATH2 = "C:/Users/EstebanCj/Desktop/michigan/econ_a1/labels_econ_a1.json"
        jsonFile=open(FILEPATH2)
        jsonData = json.load(jsonFile)
        jsonFile.close()
        #First 10 files in the dictionary
        print {k: jsonData[k] for k in jsonData.keys()[:10]}
```

```
{u'econ_a1_essay44.docx': 4, u'econ_a1_essay68.docx': 4, u'econ_a1_essay165.docx': 2, u'econ_a1_
```

- Generate a single preprocessed file that contains the texts of all Word documents.

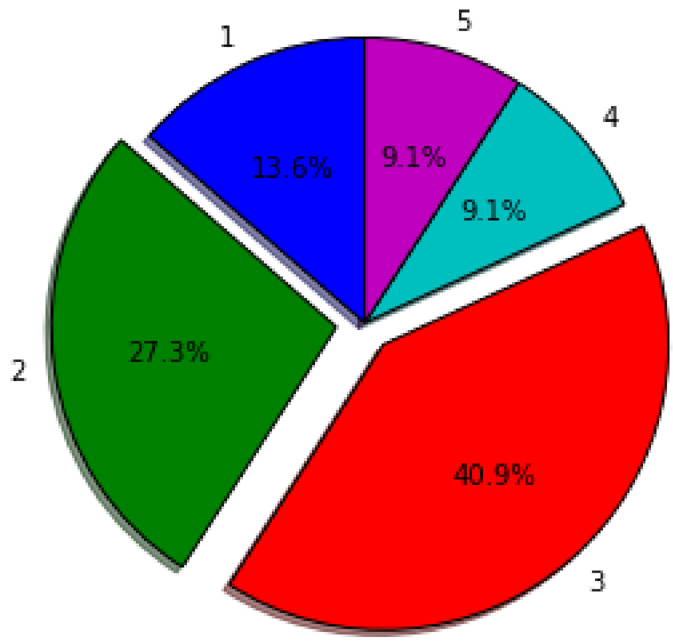
```
In [3]: import codecs
        import re
        import win32com.client
        textList=[]
        for document in fileNames:
            doc  = win32com.client.GetObject(FILEPATH+document)
            text = doc.Range().Text
            text = text.replace('\n',' ')
            text = re.sub('[^A-Za-z0-9 ]+', '',text.lower())
            textList.append((document,text,jsonData[document]))
        FILEPATH3 = "C:/Users/EstebanCj/Desktop/michigan/econ_a1/dataset.txt"
        #First element in the preprocessed text file
        print textList[0]
        with codecs.open(FILEPATH3,"w","UTF-8") as file:
            for element in textList:
                # separator element used: "@-?@"
                file.write(element[0]+"@-?@"+str(element[2])+"@-?@"+element[1]+"\n")
```

```
('econ_a1_essay1.docx', u'the theories used to construct economic models are in some cases also
```

- Plot a pie chart according to the number of scores in the dataset

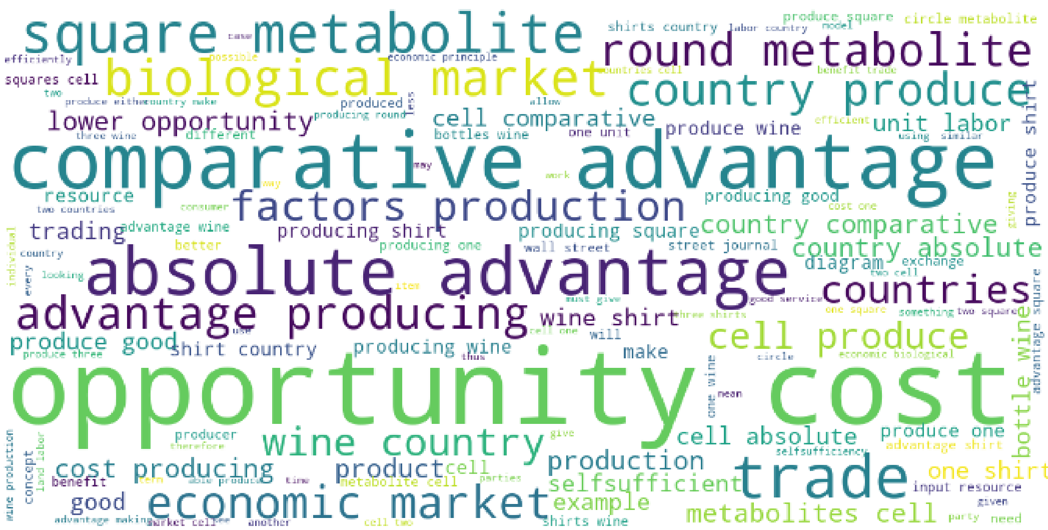
```
In [9]: %matplotlib inline
        import matplotlib.pyplot as plt

        scores=[]
        for score in range(1,6):
          scores.append(sum(1 for x in jsonData.values() if x==score))

        labels = '1', '2', '3', '4', '5'
```

```
sizes = [15, 30, 45, 10,10]
explode = (0, 0.1, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```



- Plot a word cloud that help us to see the main topics related to the dataset

```
In [29]: from wordcloud import WordCloud, STOPWORDS
         stopwords = set(STOPWORDS)
         featuresWordCloud='\t'.join([x[1]  for x in textList])
         wordcloud = WordCloud(
                 background_color='white',
                 stopwords=stopwords,
                 max_words=200,
                 max_font_size=40,
                 scale=3,
                 random_state=1).generate(str(featuresWordCloud))
         fig = plt.figure(1, figsize=(12, 12))
         plt.axis('off')
         plt.imshow(wordcloud)
         plt.show()
```



1.2 Feature set

Having the text preprocesed, the next step is to collect the textual features for the construction of a classification model. In this sense, the fequency of occurrence of Part of Speech bigrams is choosen. The following show how to obtain the proposed features:

- Obtain the Part of speech Tags (PoS tags) associated to each text document.

```
In [10]: from pattern.en import parse
         PoSTagList=[]
         FILEPATH3 = "C:/Users/EstebanCj/Desktop/michigan/econ_a1/dataset.txt"
         with codecs.open(FILEPATH3,"r","utf-8") as file:
             for line in file:
               elementList=line.split("@-?@")
               elementsParser=parse(elementList[2])
               PoSTags=[]
               for PoSTag in elementsParser.split(" "):
                  elements=PoSTag.split("/")
                  PoSTags.append(elements[1])
               PoSTagList.append(PoSTags)
         #Part of Speech tags of  econ_a1_essay1.docx
         print PoSTagList[0]
```

```
[u'DT', u'NNS', u'VBD', u'TO', u'VB', u'JJ', u'NNS', u'VBP', u'IN', u'DT', u'NNS', u'RB', u'JJ',
```

- Transform the Part of speech Tags to 5-grams for each text document

In [11]:
```
from nltk.util import ngrams
NgramList=[]
for posTags in PoSTagList:
    nGrams=ngrams(posTags,5)
    NgramList.append([' '.join(e for e in nGram) for nGram in nGrams])
#Part of Speech bigrams of econ_a1_essay1.docx
print NgramList[0]
```

```
[u'DT NNS VBD TO VB', u'NNS VBD TO VB JJ', u'VBD TO VB JJ NNS', u'TO VB JJ NNS VBP', u'VB JJ NNS
```

- Obtain the 5-grams frequency of occurrence considering all text documents in order to obtain a feature set.

In [12]:
```
import operator
nGramFeatureSet={}
for elements in NgramList:
  for ngram in elements:
     if ngram in nGramFeatureSet:
        nGramFeatureSet[ngram]=nGramFeatureSet[ngram]+1
     else:
        nGramFeatureSet[ngram]=1
nGramFeatureSetSort = sorted(nGramFeatureSet.items(),
                             key=operator.itemgetter(1),
                             reverse=True)
#50 most frequent Part of Speech bigrams in all documents
print nGramFeatureSetSort[:50]
```

```
[(u'VBZ DT JJ NN IN', 1144), (u'DT JJ NN IN NN', 576), (u'DT JJ NN IN VBG', 493), (u'NN VBZ DT J
```

- Generate a feature set file considering the frequency of occurence of Part of Speech 5-grams in all text documents.

In [27]:
```
FILEPATH4 = "C:/Users/EstebanCj/Desktop/michigan/econ_a1/featureSet.txt"
with codecs.open(FILEPATH4,"w","UTF-8") as file:
    for element in nGramFeatureSetSort:
        file.write(element[0]+"@-?@"+str(element[1])+"\n")
```

1.3 Vector representation

The next step to predict the essays scores is to transform each document to a vector representation using the bag of words model. The following steps show how to obtain these vectors:

- Select the top 100 features obtained from the feature set file (featureSet.txt).

In [14]:
```
import codecs
vectorFeatures=[]
FILEPATH5 = "C:/Users/EstebanCj/Desktop/michigan/econ_a1/featureSet.txt"
```

```
counter=0
numberFeatures= 200
with codecs.open(FILEPATH5,"r","UTF-8") as file:
    for line in file:
        counter=counter+1
        if counter <= numberFeatures:
            elementsList=line.split("@-?@")
            vectorFeatures.append(elementsList[0])
        else:
            break
#Textual features selected for the vector representation
print vectorFeatures
```

```
[u'VBZ DT JJ NN IN', u'DT JJ NN IN NN', u'DT JJ NN IN VBG', u'NN VBZ DT JJ NN', u'NN IN DT JJ NN
```

- Transform each text (in dataset.txt) to a vector representation searching for the frequency of occurrence of the selected features.

```
In [15]: from nltk.util import ngrams
         from pattern.en import parse
         import codecs
         FILEPATH6 = "C:/Users/EstebanCj/Desktop/michigan/econ_a1/dataset.txt"
         vectorList=[]
         sentimentTags=[]
         with codecs.open(FILEPATH6,"r","utf-8") as file:
             for line in file:
                 vector=[]
                 elementList=line.split("@-?@")
                 sentimentTags.append(elementList[1])
                 elementsParser=parse(elementList[2])
                 PoSTags=[]
                 for PoSTag in elementsParser.split(" "):
                    elements=PoSTag.split("/")
                    PoSTags.append(elements[1])
                 nGrams=ngrams(PoSTags,5)
                 nGramsList=[' '.join(e for e in nGram) for nGram in nGrams]
                 for feature in vectorFeatures:
                     vector.append(nGramsList.count(feature))
                 vectorList.append(vector)
         # Vector representation of econ_a1_essay1.docx
         print "vector"
         print vectorList[0]
         print "score associated"
         print sentimentTags[0]
```

```
vector
[2, 2, 1, 0, 3, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 3, 0, 1, 0, 0, 0, 0, 3, 0, 0, 1, 0, 0, 0, 1, 1, 2,
score associated
```

4

1.4 Classification process

In the next step, the vectors previously created are used to construct predictive models using different supervised learning algorithms. The following steps show how to obtain these models:

- Separate the dataset into a train a test subsets.

```
In [16]: X_train= vectorList[0:200]
         y_train= sentimentTags[0:200]
         X_test = vectorList[201:]
         y_test =sentimentTags[201:]
```

- Create different classification models using the training vectors.

```
In [17]: #Support Vector machine classifier(SVM)
         from sklearn import svm
         from sklearn.svm import SVC
         # SVC with polynomial (degree 3) kernel
         C = 1.0
         clf= poly_svc = svm.SVC(kernel='poly', degree=3, C=C)
         #training phase (model construction)
         clf.fit(X_train, y_train)

Out[17]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto', kernel='poly',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False)

In [18]: #Naive Bayes classifier
         from sklearn.naive_bayes import MultinomialNB
         #training phase (model construction)
         clf2 = MultinomialNB().fit(X_train, y_train)

In [19]: #Logistic Regression classifier
         from sklearn import linear_model
         clf3 = linear_model.LogisticRegression(C=1e5)
         clf3.fit(X_train, y_train)

Out[19]: LogisticRegression(C=100000.0, class_weight=None, dual=False,
                   fit_intercept=True, intercept_scaling=1, max_iter=100,
                   multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
                   solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

- Using the previously created models evaluate the vectors of the test dataset.

In [20]:
```
#test the model considering the frequency vectors for the SVM classifier
testResults=[]
for vector in X_test:
    testResults.append(clf.predict([vector])[0])
#Predicted scores of the tes dataset
print "SVM classifier results"
print testResults
#test the model considering the frequency vectors for the Naive Bayes classifier
testResults2=[]
for vector in X_test:
    testResults2.append(clf2.predict([vector])[0])
#Predicted scores of the tes dataset
print "Naive Bayes classifier results"
print testResults2
#test the model considering the frequency vectors for the logistic regression classifie
testResults3=[]
for vector in X_test:
    testResults3.append(clf3.predict([vector])[0])
print "logistic regression classifier results"
print testResults3
print "Correct values of the test datase"
print y_test
```

```
SVM classifier results
[u'3', u'3', u'3', u'3', u'3', u'4', u'3', u'3', u'3', u'4', u'3', u'3', u'3', u'3', u'3', u'3',
Naive Bayes classifier results
[u'2', u'2', u'3', u'4', u'2', u'2', u'3', u'3', u'5', u'4', u'4', u'3', u'4', u'2', u'3', u'3',
logistic regression classifier results
[u'3', u'3', u'2', u'4', u'4', u'4', u'3', u'3', u'3', u'4', u'3', u'4', u'4', u'4', u'3', u'4',
Correct values of the test datase
[u'3', u'4', u'3', u'4', u'3', u'5', u'2', u'3', u'3', u'4', u'1', u'4', u'3', u'4', u'4', u'3',
```

In [21]:
```
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
print "SVM model"
print "Model f1 measure: "+str(f1_score(testResults,y_test, average='micro'))
print "Model micro presicion: "+str(precision_score(y_test,testResults, average='micro'
print "Model macro presicion: "+str(precision_score(y_test,testResults, average='macro'

print "Naive Bayes Model"
print "Model f1 measure: "+str(f1_score(testResults2,y_test, average='micro'))
print "Model micro presicion: "+str(precision_score(y_test,testResults2, average='micro
print "Model macro presicion: "+str(precision_score(y_test,testResults2, average='macro

print "logistic regression Model"
print "Model f1 measure: "+str(f1_score(testResults3,y_test, average='micro'))
print "Model micro presicion: "+str(precision_score(y_test,testResults3, average='micro
print "Model macro presicion: "+str(precision_score(y_test,testResults3, average='macro
```

```
SVM model
Model f1 measure: 0.45454545454545453
Model micro presicion: 0.45454545454545453
Model macro presicion: 0.19710407239819006
Naive Bayes Model
Model f1 measure: 0.3333333333333333
Model micro presicion: 0.3333333333333333
Model macro presicion: 0.18375
logistic regression Model
Model f1 measure: 0.3434343434343434
Model micro presicion: 0.3434343434343434
Model macro presicion: 0.2122335519573683
```

```
c:\python27\lib\site-packages\sklearn\metrics\classification.py:1135: UndefinedMetricWarning: Pr
  'precision', 'predicted', average, warn_for)
```