CE807: Text Classification Comparison

In this lab we are going to use scikit-learn for text classification, focusing in particular on the most classic example of text classification: spam detection. This builds on the previous lab and tries to compare different classification algorithms. We have already seen how to run code for the Naïve Bayes classifier.

All the materials for this lab, including this script, can be found in moodle.

Copy all the material in that folder, including the data sub-folder, into a directory in your file space.

Note that you always have to work on Google Colab, so make sure that you have copied data into Google drive and know how to mount the GDrive and copy the required data into the Colab working directory. You must learn to work with Google Colab. You will be submitting Assignment 2 using Google Colab only.

1. Cross Validation

Note that, in the Lab-04 we already looked at the how to read the dataset and make a Naïve Bayes classifier. Use that before doing cross-validation.

Let's assume that we have already read the dataset and set the pipeline as follows

```
from sklearn.pipeline import Pipeline
pipeline = Pipeline([
          ('vectorizer',  CountVectorizer()),
('classifier',  MultinomialNB()) ])
pipeline.fit(data['text'].values,
data['class'].values)
pipeline.predict(examples) # ['spam', 'ham']
```

Now, we will see how to use **cross-validation** to assess the performance of the model.

k-fold cross validation is a way for testing that involves splitting your dataset into k **folds**, then in turn use each of the k folds as a test set and using the other k-1 to train.

Scikit-learn makes it very easy to do cross-validation: the class KFold takes as argument the number of folds/splits along with other arguments, and generates k pairs of index vectors, one for each of the k iterations.

```
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix, f1_score

k_fold = KFold(n_splits=2, random_state=None,
shuffle=False)
scores = []
confusion = numpy.array([[0, 0], [0, 0]])

for i, (train_indices, test_indices) in
enumerate(kf.split(data)):
      train_text= data.iloc[train_indices]['text'].values
      train_y = data.iloc[train_indices]['class'].values

      test_text= data.iloc[test_indices]['text'].values
      test_y = data.iloc[test_indices]['class'].values

     pipeline.fit(train_text, train_y)
     predictions= pipeline.predict(test_text)
     confusion+=confusion_matrix(test_y, predictions)
     score=f1_score(test_y, predictions, pos_label='spam')
     scores.append(score)

print('Total emails classified:', len(data))
print('Score:', sum(scores)/len(scores))
print('Confusion matrix:')  print(confusion)
```

Exercise **Try different f1_score (like micro, macro) and see how the value changes. You should always verify that your prediction has different classes. The confusion matrix provides that verify how these changes in each loop.**

Exercise **Find out how to save and load models from GDrive. Remember, training will be done only once, and you will load the same trained model to test on different data sets. This will require saving and loading models**

1.1 Fixed Train, Validation and Test Set

In most cases, you will have or need to create a fixed train, validation, and test set for the classification. Explore whether you could use sklearn's train_test_split for this for the spam dataset.

2. Comparing Different Models and Task

In this part of the lab, your task will be to try and improve the performance of the spam detector developed in Section 1. The idea is that you should try to modify the original script, and test whether the new program is better than the previous one. (This is the task that you will have to do in Assignment!) Here are three suggestions, but if you think of something else, go ahead.

2.1 Using different classifiers

Within a platform like scikit-learn, the easiest way to try and improve a system is by trying different machine learning algorithms and/or different parameters for them. Possible classifier

- `BernoulliNB`
- `SVM`
- `Logistic Regression`

Form an intuition which method performs better and why.

2.2 Using different tasks

You should explore how to use different text classification datasets, for example,

- IMDB Sentiment Analysis
- Emotion Dataset

How performance varies across different tasks and models. Which combination of model and task works best? Which evaluation method is suitable? Do you see changes in F1-Score? Why are these values different?

References

- scikit-learn's documentation at http://scikit-learn.org/stable/index.html
- Zac Stewart's tutorial on spam detection with scikit learn at: https://zacstewart.com/2015/04/28/document-classification-with-scikit-learn.html