ML for NLP 1 - Assignment 1

Language Identification with sklearn

October 9, 2019

1 Part 1 - Linear Classifiers

This part describes the training using linear classifiers.

1.1 Data Set

The Data Set contains following 3 files

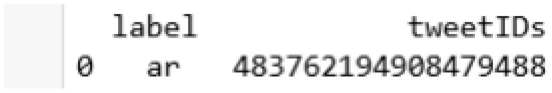
- tweets.jso - This file contains the tweets along with the Tweet-IDs,
- labels-train+dev.tsv – This file contains training and validation set labels along with TweetIDs.
- labels-test.tsv – This file contains test set labels along with Tweet-IDs.

1.2 Data Set Properties

- tweets.jso contains 66921 samples, where in each sample specifies the tweet text along with Tweet-ID. We call it Data Set A

```
             tweetIDs                                                   text
0  483885347374243841  ...اللهم أفرح قلبي وقلب من أحب وأغسل أحزاننا وهمو
```

- labels-train+dev.tsv contains 96414 samples, where in each sample specifies language label along with tweet id. We call it Data Set B



```
  label            tweetIDs
0    ar  483762194908479488
```

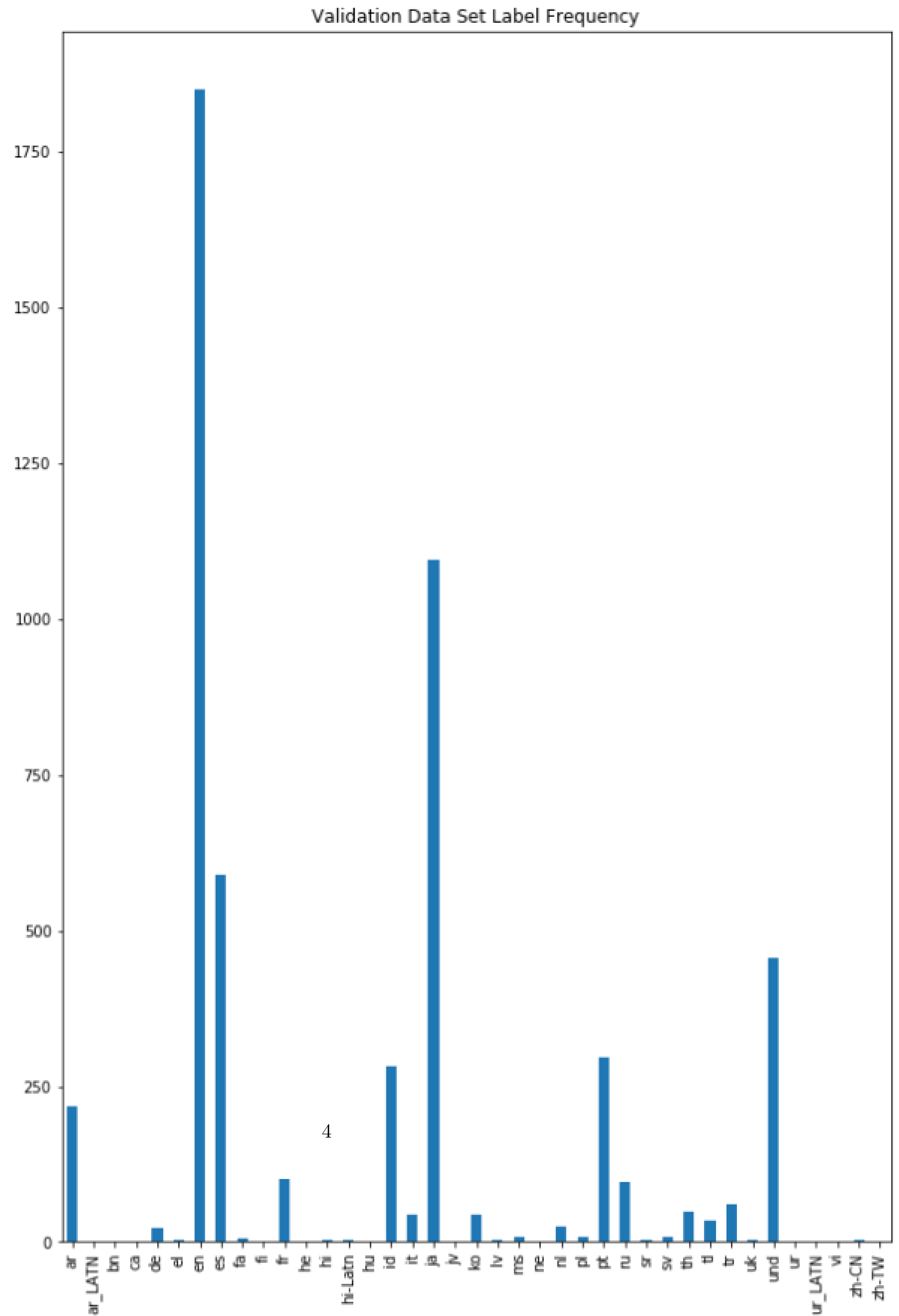
- labels-test.tsv contains 24114 samples, where in each sample specifies language label along with tweet id. We call it Data Set C
- Get Training data set by $A \cap B$. Get test data set by $A \cap C$. Discard training and test set samples, which are not present in data set A. Training set ($A \cap B$) contains 84 duplicate values and Test set ($A \cap C$) contains 6 duplicate values(Tweets having the same text).
  Remove these duplicates.
- Divide the Training set into Training +Development Set with a 90:10 split.
- Remove 39 samples from Training $Set(A \cap B, 53385)$ for which nu of label instances are less than 5. Reason for this step is that cross validation with value 5 requires at least 5 samples for each label corresponding to each combination of training and validation set.

```
train=train.groupby("label").filter(lambda x: len(x) >= 10)
```

- Remove samples from Test and Validation set corresponding to labels that not present in Training Set. Classifier cannot predict a label class until it has seen this class during training.

```
validation = (validation[validation.label.isin(train.label)])
```

- Finally plot the Training and Development Set
  Training Set Count: 48007
  Test Set Count: 13425
  Validation Set Count: 5331


Validation Data Set Label Frequency
0
250
500
750
1000
1250
1500
1750
ar
ar_LATN
bn
ca
de
el
en
es
fa
fi
fr
he
hi
hi-Latn
hu
id
it
ja
jv
ko
lv
ms
ne
nl
pl
pt
ru
sr
sv
th
tl
tr
uk
und
ur
ur_LATN
vi
zh-CN
zh-TW

1.3 Classifier comparison

As a baseline the SGDClassifier produces an accuracy of 96.1% on the training set, 83.9% on the validation set and 84.3% on the test set using the default settings provided by sklearn. Where as the MNBClassifier produces an accuracy of 80.1% on the training set, 63.6% and 63.7% respectively.

In order to optimize the results the following hyperparameters were used in the GridSearchCV:

Parameter	Option	Description
Loss function	hinge squared hinge modified huber	
# iterations	50, 100, 200	
alpha	From 0.1 to 0.001	Learning rate (larger rates omitted due to lecture)
penalty	l2, l1, none	

Table 1: SGDClassifier Hyperparameters

The best performing hyperparameters for the SGDClassifier produces an accuracy of 82.7% using the following combination of hyperparameters:

- Loss: hinge
- Alpha: 0.001
- #iterations: 50
- Penalty: none
- ngram range: (1,1)

Parameter	Option	Description
ngram range	(1,1), (1,2), (1,3)	
alpha	From 0.1 to 0.001	Learning rate (larger rates omitted due to lecture)
fit prior	true, false	

Table 2: MNBClassifier Hyperparameters

The best performing hyperparameters for the MNBClassifier produces an accuracy of 76.4% using the following combination of hyperparameters:

- Alpha: 0.001
- fit prior: True
- ngram range: (1,2)

Accuracy	Multinomial Naive Bayes (Default)	Multinomial Naive Bayes (GridSearch)	SGDClassifier (Default)	SGDClassifier (GridSearch)
Training Set	80.1	99.4	96.1	88.1
Validation Set	63.2	76.4	83.9	82.7
Test Set	63.5	75.7	84.3	82.2

Table 3: Classifier Comparison

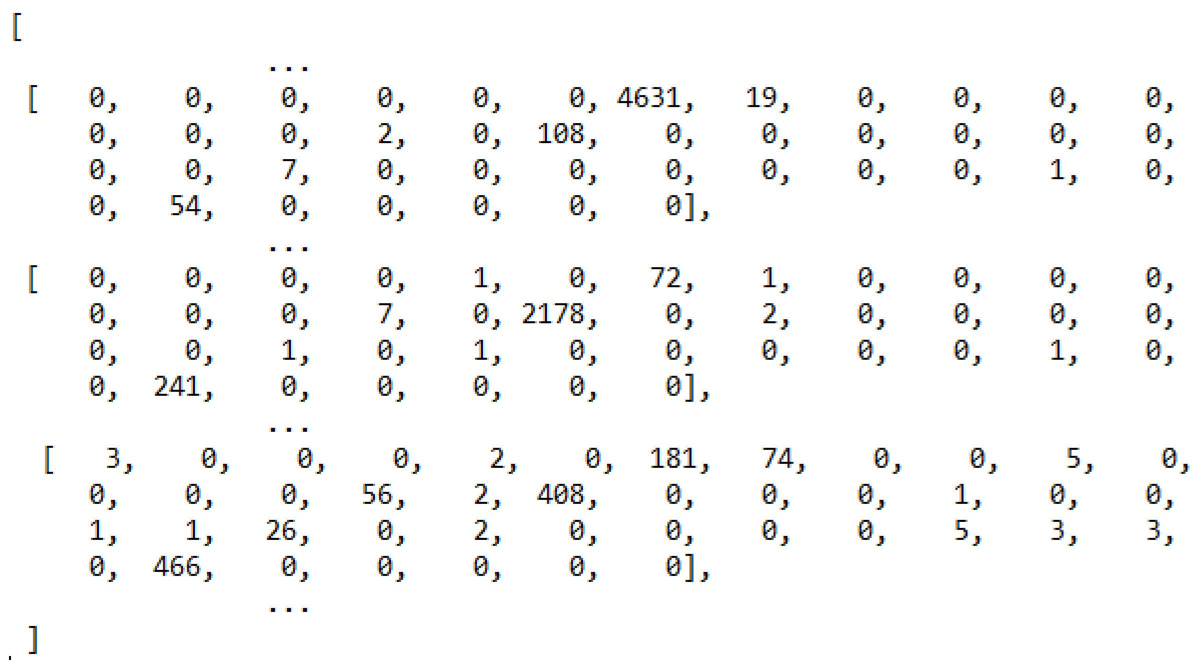
It is quite clear that model is doing overfitting in case of MNB in case of Grid Search. MNB Classifier with GridSearch gives an accuracy of 99.41% on training data, where as on unseen data, prediction accuracy drops to around 75%.

We can see that Gridsearch gives an improvement of almost 20% with MNB Classifier as compared to default configuration. On the other hand for SGD-Classifier there is a slight drop of around 1.5% in performance, which is most likely due to the limited hyperparameters used in our GridSearchCV trial.

1.4 Advantage of GridSearch Cross Validation

The main advantage of GridSearchCV is that it systematicaly tries out the different hyperparameters and performs the specified K-Fold Cross Validation for each of the combinations of hyperparameters. Thus, evaluating the best parameter set to train the model, where as the K-Fold cross validation helps us keep the test set uncorrupted. An additional bonus is the built in option of running the GridSearchCV with multiple workers to hopefully decrease the necessary time to run the computations.

1.5 Error analysis with confusion matrix



```
[
                    ...
 [   0,    0,    0,    0,    0,    0, 4631,   19,    0,    0,    0,    0,
     0,    0,    0,    2,    0,  108,    0,    0,    0,    0,    0,    0,
     0,    0,    7,    0,    0,    0,    0,    0,    0,    0,    1,    0,
     0,   54,    0,    0,    0,    0,    0],
                    ...
 [   0,    0,    0,    0,    1,    0,   72,    1,    0,    0,    0,    0,
     0,    0,    0,    7,    0, 2178,    0,    2,    0,    0,    0,    0,
     0,    0,    1,    0,    1,    0,    0,    0,    0,    0,    1,    0,
     0,  241,    0,    0,    0,    0,    0],
                    ...
  [   3,    0,    0,    0,    2,    0,  181,   74,    0,    0,    5,    0,
     0,    0,    0,   56,    2,  408,    0,    0,    0,    1,    0,    0,
     1,    1,   26,    0,    2,    0,    0,    0,    0,    5,    3,    3,
     0,  466,    0,    0,    0,    0,    0],
                    ...
 ]
```

Figure 1: Partial confusion matrix

Looking at the confusion matrix the most prominent examples are the following languages:

- undefined: Has the highest percentage of misclassifications, which is due to different combinations of languages and special symbols in the sample and causes confusion for the model.
- English: The highest misclassification of english was japanese, which was due to english words within the japanese samples. And the other prominent misclassification is the undefined class, which as mentioned before causes further confusion in the classification.
- japanese: Combination of japanese and english words, mostly misclassified by the undefined class.

1.6 Dealing with class imbalance

The following options are available to deal with class imbalance:

- Oversample minority class
  Can be done with the resample() method provided by sklearn

- Undersample majority class
  Analogous to the oversampling we remove samples from the majority class using the resample() method from sklearn
- Generate synthetic samples
  This approach is similar to oversampling, but generates new synthetic data for training the model

2 Part 2 - Multi Layer Perceptron

Training the model with the MLP Classifier and a multitude of parameters took considerable time.
The data was preprocessed in the same fashion as in the first part of the excercise.

The following pipeline definition was used for the GridSearchCV:

```
text_mlp = Pipeline([
('vect', CountVectorizer()),
('tfidf', TfidfTransformer()),
('mlp', MLPClassifier())
])
```

Which uses the CountVectorizer to vectorize the data and the labels and the TfidfTransformer for the frequency calculation. The to be used Classifier as mentioned before is the MLP classifier.

The pipline is then fed into the GridSearchCV to try out the different parameters as listend in Table 4.

2.1 Parameters

Parameter	Option	Description
solver	SGD lbfgs	Stochastic Gradient Descent Doesnt penalize intercept, Robust to unscaled datasets
max_iter	100, 150	Tries to converge the data within the given iterations
alpha	From 0.01, 0.001	Learning rate (larger rates omitted due to lecture)
hidden_layer_sizes	Range from 1 to 4	Defines how many hidden layers the model uses

Table 4: Parameters

The best performing hyperparameters for the MLPClassifier produces an accuracy of 86.4% on the training set, 77.9% on the test set and 77.5% on the validation set using the following combination of hyperparameters:

- Alpha: 0.01
- #iterations: 100
- solver: lbfgs
- hidden layer size: 4