Knowledge based systems and language technology

Natural Language Processing

Software Project

Faculty of Mathematics and Computer Science

Romanian sub-dialect identification

Emanuel Bîscă

Applied Computational Intelligence, M.Sc. I

Contents – Romanian sub-dialect identification

1. Problem statement

The main task I want to solve for this project, is to properly discriminate between the Moldavian and the Romanian dialects across different text genres (news and tweets).

I will train my model mainly on tweets. Therefore, the model is built for a in-genre binary classification by dialect task, in which a classification model is required to discriminate between the Moldavian (label 0) and the Romanian (label 1) dialects.

The training data is composed of 7757 samples. The validation set is composed of 2656 samples. All samples are preprocessed in order to replace named entities with a special tag: NE. Another important remark is that the tweets are encrypted.

2. Proposed solution

2.1. Theoretical aspects

File description

- *train_samples.txt* – the training data samples (one sample per row);
- *train_labels.txt* – the training labels (one label per row);
- *validation_samples.txt* – the validation data samples (one sample per row);
- *validation_labels.txt* – the validation labels (one label per row);
- *test_samples.txt* – the test data samples (one sample per row);
- *sample_submission.txt* – a sample submission file in the correct format.

Data format

The data samples are provided in the following format based on TAB separated

```
112752  sAFW K#xk}t fH@ae m&Xd >h& @# l@Rd}a @Hc liT ehAr@m Xgmz !}a }eAr@m Be g@@m efH RB(D Ehk&
107227  X;d:N qnwB Acke@m m*g lvc& ggcp ht*A mat; }:@ HA&@m HA@e hZ Er#@m
101685  #fEw w!ygF dDB XwfE| HrWe@mH
```

values:

Each line represents a data sample where:

- the first column shows the ID of the data sample.
- the second column is the actual data sample.

The labels are provided in the following format based on TAB separated values:

```
1   1
2   0
```

Each line represents a label associated to a data sample where:

- the first column shows the ID of the data sample.
- the second column is the actual label.

2.2. Application

Mainly the proposed model solves the task of identifying the two dialects considered in the hypothesis, Romanian and Moldavian. Even though the task comes from the natural language processing field, I extensively utilized a large amount of knowledge from the machine learning field, when designing the so-called application proposed for this software project.

In the next section I will give more insights on the approach I've been applying to solve the problem stated in the beginning.

3. Implementation

I implemented my model in Python. I'll start by mentioning the libraries I've been using to solve the task. I utilized `numpy`, to manipulate the arrays, the `csv` to write the file in the desired format, `TfidfVectorizer` for feature extraction, `f1_score`, `confusion_matrix`, `accuracy_score`, `classification_report` because I had to evaluate the performances and `MultinomialNB`.

```
import numpy as np
import csv
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB
```

Regarding the way in which I handled reading the data, I believe that a good choice was with `open`, using the `readlines` method. It should also be mentioned that I concatenated the train data and the validation data, with the goal of obtaining to the highest possible accuracy and to train the model on as much data as possible.

On the samples side I needed to put only the text part (which belonged to each sentence) in a variable, getting this done by separating the text ID using the split function with the *\t* separator. All this being inside a for, every new sentence was appended to the rest. The exact same procedure was applied to train samples as well as to validation samples; the resulting vectors were concatenated.

```
file = open("C:/Users/Emanuel/Desktop/NLP-RDI/train_samples.txt", encoding="utf8")
train1_samples = file.readlines()

file = open("C:/Users/Emanuel/Desktop/NLP-RDI/validation_samples.txt", encoding="utf8")
train2_samples = file.readlines()
```

```
samples1 = []
for x in train1_samples:
    samples1.append(x.split("\t")[1])

samples2 = []
for y in train2_samples:
    samples2.append(y.split("\t")[1])
```

As you can predict, the same mechanism is in place for the labels, this time separating the ID and the label (label 0 or label 1), concatenating the result in the end.

```
file = open("C:/Users/Emanuel/Desktop/NLP-RDI/train_labels.txt", encoding="utf8")
train1_labels = file.readlines()
file = open("C:/Users/Emanuel/Desktop/NLP-RDI/validation_labels.txt", encoding="utf8")
train2_labels = file.readlines()
```

```
labels1 = []
for x in train1_labels:
    labels1.append(x.split()[1])
labels2 = []
for y in train2_labels:
    labels2.append(y.split()[1])
```

```
data_samples = []
data_samples = samples1 + samples2
data_labels = []
data_labels = labels1 + labels2
```

When it comes to feature extractions, I wanted to use `TFIDF Vectorizer` (stands for term frequency – invers document frequency) because it basically transforms the text into characteristic vectors that will become then the input for the model. I chose this vectorizer because it was proved to offer a better performance, compared to `CountVectorizer`, another widely known equivalent.

Regarding the classifier, I must affirm that it's been difficult to choose the better option, because there are too many. On this specific dataset, the best performance was obtained with the `MultinomialNB`, so I utilized it in designing the model of this software project.

4. Experiments and Results

To test the accuracy of the proposed model, I opted for `f1 score`, being interpreted as a weighted average of accuracy and recall, where the best score is 1 and the weakest reaching point 0.

I also used confusion matrix, accuracy score and classification ratio. For example, for the final model, `MultinomialNB`, the situation is shown below:

```
C:\Users\Emanuel\AppData\Local\Programs\Python\Python39\python.exe
Confusion Matrix :
[[1001  300]
 [ 169 1186]]
Accuracy Score : 0.8234186746987951
Report :
              precision    recall  f1-score   support

           0       0.86      0.77      0.81      1301
           1       0.80      0.88      0.83      1355

    accuracy                           0.82      2656
   macro avg       0.83      0.82      0.82      2656
weighted avg       0.83      0.82      0.82      2656
```