Identify Language of a Text Using Logistic Regression and Multinomial Naive Bayes

Rio Pramana, Debora, Enrico Fernandez

Presentation Outline

01 Background Problem

02 Proposed Solution

03 Methodology

04 Demo

Background Problem

01 More people get access to the web, and more languages and dialects start to appear and need to be processed.

02 Language Identification (LID) is crucial to many NLP applications, such as to give the right auto-correct suggestions

03 Track and identify text/document containing multiple languages (multilingual)

Proposed Solution

Logistic Regression

Multinomial Naive Bayes

Methodology

01 Data Collection

Papluca Language Identification
(from Hugging Face)

Subset: papluca--language-identification

Split: train

labels (string)	text (string)
pt	Duas raças diferentes de cães castanhos e brancos brincam na praia.
es	Un producto de una calidad y capacidad increíbles que será el placer de todo amante de la tecnología

Link: https://huggingface.co/datasets/papluca/language-identification#additional-information

The Language Identification dataset is a collection of 90.000 samples and contains text in 20 languages

Methodology

02 Preprocessing Data and Exploratory Data Analysis

- Handling missing data
- Check data cardinality
- Text Preprocessing

Handling Missing Data

The initial dataset has no missing data as seen in Fig. 1.

	number of null values
labels	0
text	0

Fig 1. Number of mising data in the dataset

Check Data Cardinality

The 'text' variable contains 68978 labels, which means there are probably 68978 unique text data as seen in Fig. 2.

```
labels  contains  20  labels
text  contains  68978  labels
```

Fig 2. Data Cardinality

Text Preprocessing

01 Lower case conversion

Lowering the case of all text can help the model to interpret the set. This helps to identify the inputted text as one.

02 Removing unnecessary text

We are cleaning the text before we use it to train the model. We used a regex to remove unnecessary characters such as punctuations and website links. The regex we used is:

```
(@[A-Za-z0-9]+)|([!"#$%&\'()*+,-./:;<=>?@[\]^_`{|}])|(\w+://\S+)|^rt|http.+?
```

Text Preprocessing

03 Encoding categorical data

We performed Encoding using LabelEncoder() from scikit-learn on the target variable (labels) as the features (text) will be encoded in the process of vectorization.

04 Vectorization

We implement a couple of vectorizations, especially TF-IDF which creates vectors from text which contains information on the more important words and the less important ones as well. We used either CountVectorizer or TfIdfVectorizer for each model

Methodology

03 Modelling

```
[7] #Fit and save the model
    vectorizer = CountVectorizer()
    model = pipeline.Pipeline([
        ('vectorizer', vectorizer),
        ('clf', MultinomialNB())
    ])
    model.fit(x_train,y_train)
    with open("mnb_model.pkl", "wb") as f:
      pickle.dump(model, f)
```

Multinomial Naive Bayes

```
[9] #Fit and save the model
    vectorizer = TfidfVectorizer(ngram_range=(1,3), analyzer='char')
    model = pipeline.Pipeline([
        ('vectorizer', vectorizer),
        ('clf', LogisticRegression())
    ])
    model.fit(x_train,y_train)
    with open("lr_model.pkl", "wb") as f:
      pickle.dump(model, f)
```

Logistic Regression

Methodology

03 Modelling

- CountVectorizer()
- CountVectorizer(ngram_range=(1,3), analyzer='char')
- TfIdfVectorizer(ngram_range=(1,3), analyzer='char')

Methodology

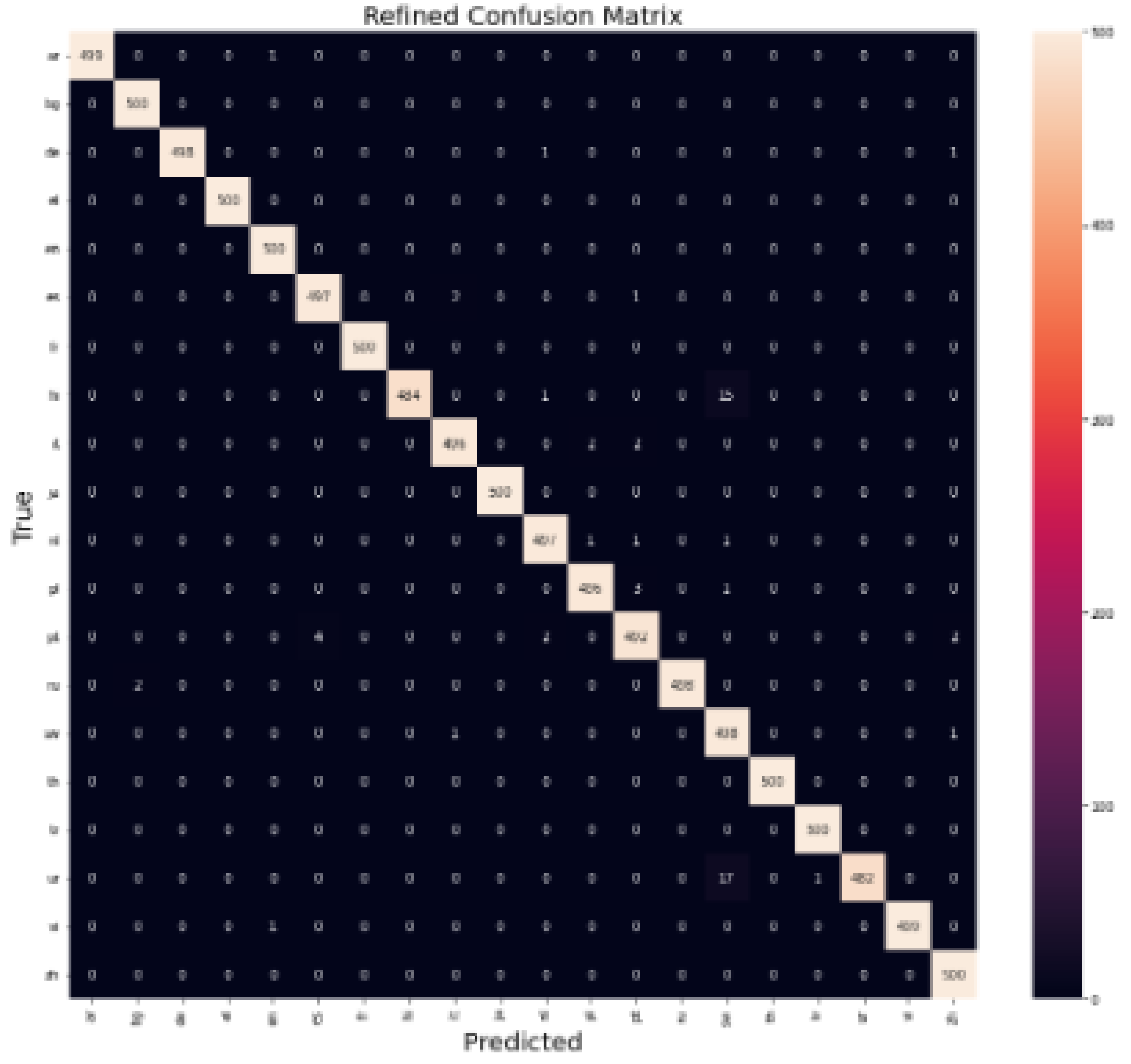
04 Evaluation Metrics

- Evaluation metrics: accuracy, precision, recall, f1-score and confusion matrix.

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	500
1	1.00	1.00	1.00	500
2	1.00	1.00	1.00	500
3	1.00	1.00	1.00	500
4	1.00	1.00	1.00	500
5	0.99	0.99	0.99	500
6	1.00	1.00	1.00	500
7	1.00	0.97	0.98	500
8	0.99	0.99	0.99	500
9	1.00	1.00	1.00	500
10	0.99	0.99	0.99	500
11	0.99	0.99	0.99	500
12	0.99	0.98	0.98	500
13	1.00	1.00	1.00	500
14	0.94	1.00	0.97	500
15	1.00	1.00	1.00	500
16	1.00	1.00	1.00	500
17	1.00	0.96	0.98	500
18	1.00	1.00	1.00	500
19	0.99	1.00	1.00	500
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

Accuracy:
Model accuracy score: 0.9936



Methodology

04 Evaluation Metrics

- We compared the results of each model with each other by their accuracy shown in Table below.

TABLE VI. MODEL ACCURACY COMPARISON (IN PERCENTAGE)

Model	with TfIdfVectorizer(with parameters)	with CountVectorizer(default parameters)	with CountVectorizer(with parameters)
Logistic Regression	99.36%	91.84%	99.20%
Multinomial Naive Bayes	98.97%	92.14%	98.81%

- From this result, we decided to go with lr_model and mnb_model_tfidf for our final application as they are the best performing model for logistic regression and multinomial naive bayes respectively.

Demo

Thank You