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

Rio Pramana
Computer Science Department
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia
rio.pramana@binus.ac.id

Debora
Computer Science Department
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia
debora002@binus.ac.id

Enrico Fernandez
Computer Science Department
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia
enrico.fernandez@binus.ac.id

***Abstract*—Being able to predict the language of a certain text is important, especially for people who are in multilingual societies. Within this, it may help to solve their problems by building software that can help to identify the basis of the language of their surrounding uses. We have tried to achieve this by building different models that can perform multiclass classification to identify the language of a text. We created six different models using two algorithms, logistic regression and multinomial naive bayes. From these different models, we found that the Logistic Regression model paired with TfIdfVectorizer is the best model to use, reaching an accuracy score of 99.36%.**

Keywords—Language Identification, Natural Language Processing, Multiclass Classification, Logistic Regression, Multinomial Naive Bayes

I. Introduction

More people get access to the web, and more languages and dialects start to appear and need to be processed. Every human being may not be expected to know all the languages. There may be useful information on the digitalized world but set in a foreign language. So, to identify which language it corresponds to we need to track and identify the source of the language. Language identification is crucial in Natural Language Processing (NLP) that is able to automatically identify the natural language as to what the content is given and give the right auto-correct suggestions. It plays an important role especially for human life because language is a tool of human communication in everyday life. There are many people and diverse languages in this world, and some of them are interested in existing languages. Especially if someone was in another country where a foreign language is spoken there. Therefore, we need a model that can identify existing languages.

The prediction results can help people to identify what language is spoken or written. Language identification is challenging and accurate results must be taken because there are diverse languages in the world. There are many of those languages that are similar and there are also many that differ greatly.

This paper is organized as follows: the next section focuses on the literature review. The methodology utilized is given in section III. Section IV constitutes the results and discussions, and finally, the study's conclusions are given in section V.

II. Literature Review

A. Multiclass Classification

Multiclass classification is one of classification tasks in machine learning. The task is to classify each instance into one of three or more classes. To do multiclass classification, we can train a classifier using several algorithm options such as Naive Bayes, decision trees, and logistic regression. There are multiple multiclass classification techniques, one of them is transformation to binary [1].

Transformation to binary is essentially a strategy to reduce the problem of multiclass classification into multiple binary classification. One of the techniques used to achieve this is called One-vs.-rest (OvR), or often called One-vs.-all (OvA). This technique trains a classifier for each class with samples of that class being positive samples and other classes being negative samples [2].

B. Logistic Regression

Logistic regression model is a statistical model used to model the probability of one event happening out of two possible events. Logistic regression model is often used for predictive analysis and classification. It can estimate the probability out of two possible events, but can only output a number between 0 and 1 since the output will be a probability [3].

Despite only being able to perform binary classification, we can still apply various strategies to perform multiclass classification using logistic regression. One of the approaches is OvR where we will split the multiclass dataset into several binary classification problems and then train a classifier on each of these problems [4]. But one of the huge downsides of this is it could require a long time for large datasets or large numbers of classes because we will need to train a classifier for each one.

C. Multinomial Naive Bayes

Multinomial naive bayes is one of naive bayes variations. In multinomial naive bayes, each sample represents frequencies of each event that has been generated by a multinomial where each component of this multinomial represents the probability of an event occurring.

Rennie *et al.* discussed some problems in multinomial naive bayes implementation and proposed a few ways to solve these problems, including the use of tf-idf weights to produce a classifier that is competitive with Support Vector Machine (SVM) [5].

D. Vectorization (Machine Learning)

In machine learning, vectorization refers to the process of converting raw data into vectors of real numbers that can be used by machine learning models. Vectorization in machine learning is a step in feature extraction where we would like to extract distinct features from the data for the model to train on, by converting data to numerical vectors [6].

There are multiple techniques to perform vectorization such as Bag of Words, TF-IDF, and Word2Vec. In Python, we can use CountVectorizer() (for Bag of Words) and TfIdfVectorizer() (for TF-IDF) from scikit-learn to perform one of these techniques for our machine learning models.

III. Methodology

A. Data collection

The data set used in this paper is obtained from the huggingface.co platform at the following address (https://huggingface.co/datasets/papluca/language-identification accessed 1 May 2022). This dataset is specifically made for language identification tasks and consists of 90000 data, which are texts from 20 different languages, with information on 2 study variables as shown in Table I The data mainly shows which language the text interprets.

TABLE I. Dataset Descriptions

Categorical Data	Description
labels	Provide labels of the languages (For example: 'en' stands for English languages
text	Comprise of the sentences/words that briefly identify which labels it belongs to

Table II shows all 20 languages used for this dataset.

TABLE II. Sample Languages

Label	Language
ar	Arabic
bg	Bulgarian
de	German
el	Modern Greek
en	English
es	Spanish
fr	French
hi	Hindi
it	Italian
ja	Japanese
nl	Dutch
pl	Polish
pt	Portuguese
ru	Russian
sw	Swahili
th	Thai
tr	Turkish
ur	Urdu
vi	Vietnamese
zh	Chinese

This dataset already provided 3 splits (train, valid, and test) so we do not have to split the data ourselves. This dataset is also made to be perfectly balanced with every language having the same amount of samples in the dataset. Details of the dataset splits are shown in Table III.

TABLE III. Dataset Splits

	Train Set	Valid Set	Test Set
Total samples	70000	10000	10000
Total languages	20	20	20
Samples per language	3500	500	500

B. Preprocessing Data and Exploratory Data Analysis

For this project, we only use the train and test set. To proceed next, we set the dataset that has been split to the variables 'ds_train' and 'ds_test'. To carry out the model, a certain set of operations was carried out to prepare the data set.

1. Handling missing data

   We found out that the initial dataset, after a thorough checking, has no missing data. The target value (language) also has no missing data as seen in Fig. 1.

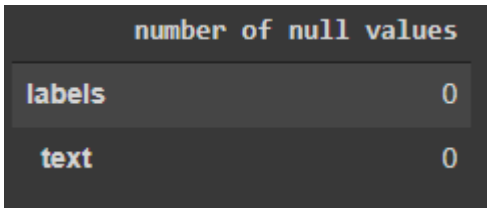

Fig. 1. Number of missing data in the dataset

2. Check data cardinality

When checking the cardinality of each categorical variable, the text contains a very high level of cardinality. The 'text' variable contains 68978 labels, which means there are probably 68978 unique text data as seen in Fig. 2.

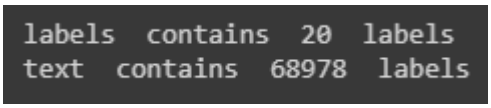

Fig. 2. Data cardinality

3. Text Preprocessing

To proceed onto the model proposed, the model needs to understand the raw text. This can be done by preprocessing the text. Preprocessing the text helps to fit them into the model. There are lots of text preprocessing that can be used. In this project, the texts preprocessing used are explained below.

a. Lower case conversion

Lowering the case of all text can help the model to interpret the set. As an example: the word "buy", if the user input the word as 'Buy', 'buy', and or 'BUY', the model may interpret these three as three different words. Lowering the case helps to identify the inputted text as one.

b. Removing unnecessary text

Oftentimes, plenty of characters/text in a dataset is unnecessary for the task that we are trying to perform, in this case, it is language identification. For example, we do not need punctuations to identify what language the text might be because any language can use punctuations, the use of punctuations do not imply the language of the text.

Therefore, we need to clean 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.+?`.

c. Encoding Categorical Data

In the dataset used for this project, it comprises two categorical data, they are the variable 'text' and 'labels'. Machine learning models can not interpret text to perform classification. Therefore, they needed to be converted into numbers first.

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 which will be explained next.

d. Vectorization

Vectorization helps in converting text to vector form in order for the model to understand. In this project 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.

To perform vectorization in Python, we used two vectorizers mentioned in the previous section; CountVectorizer and TfIdfVectorizer. We will use only one vectorizer for each model, but we will be testing and comparing both to find the best vectorizer that produces the best result. These vectorizers will be used in the model pipeline along with the classifier which will be further explained next.

C. Modelling

The objective of the project is to analyze the possibilities of predictions using natural language algorithms as a tool for predicting language from text. In this project, the following algorithms were applied: logistic regression and multinomial naive bayes.

So, after the data has been preprocessed, we continue to instantiate the vectorizer that we are going to use. In this project, as stated before, we used two vectorizers that we are going to use and compare. The best vectorizer will then be used for our final model. There are three variations that we use for our models, they are:

1. CountVectorizer()

   This CountVectorizer has no modification to the parameters (using default parameters).

2. CountVectorizer(ngram_range=(1,3), analyzer='char')

   This CountVectorizer has two modifications to the parameters. The ngram_range is set to (1,3) and the analyzer is set to 'char'.

3. TfIdfVectorizer(ngram_range=(1,3), analyzer='char')

   This TfIdfVectorizer has two modifications to the parameters. The ngram_range is set to (1,3) and the analyzer is set to 'char'.

We used the three variations mentioned above to build and train our machine learning model using either Logistic Regression or Multinomial Naive Bayes, thus 6 (six) models are trained and evaluated for this project. In the end, we will pick the best performing Logistic Regression model and

Multinomial Naive Bayes model to be used in the final application. We give a unique name for each of these 6 models as described in Table IV which we will use to refer to a specific model for the rest of this paper.

TABLE IV. MODEL VARIATIONS NAME

Model	with TfIdfVectorizer(with parameters)	with CountVectorizer(default parameters)	with CountVectorizer(with parameters)
Logistic Regression	lr_model	lr_model_count	lr_model_param
Multinomial Naive Bayes	mnb_model_tfidf	mnb_model	mnb_model_param

We used pipeline (a Python package from scikit-learn) to make our model before it is being trained. The code used can be seen in Fig. 3.

```
[ ]  #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)
```

Fig. 3. Code for model with Logistic Regression using TfIdfVectorizer

D. Evaluation Metrics

To evaluate the efficiency of various algorithms being used, it can be done by using evaluation metrics. In this paper, we focus on the accuracy, precision, recall, f-measure and confusion matrix. These can be defined as follows:

Confusion Matrix

The confusion matrix is a performance measurement for machine learning classification and useful for measuring Accuracy, Precision, Recall and F1-Score as shown in Table IV.

TABLE V. CONFUSION MATRIX

		Actual	
		Positive(P)	Negative(N)
Prediction	Positive(P)	True Positive (TP)	False Positive (FP)
	Negative(N)	False Negative (FN)	True Negative (TN)

It provides different combinations of predicted and actual values, where:

1. TP is the total number of positive values that are equal to the predicted positive.
2. FP is the total number of negatively classified data that is positive falsely.
3. FN is the total number of positive values that have been classified as negative falsely.
4. TN is the total number of negatively values that is correctly classified.

Other evaluation metrics such as recall, precision, accuracy and F1 score can be deduced in a mathematical expression respectively in equations 1, 2, 3 and 4.

$$Recall = \frac{true\ positive}{true\ positive + false\ negative} \quad (1)$$

$$Precision = \frac{true\ positive}{true\ positive + false\ negative} \quad (2)$$

$$Accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ input\ samples} \quad (3)$$

$$F1\ Score = \frac{2\ x\ recall\ x\ precision}{recall + precision} \quad (4)$$

We then compared the results of each model with each other. An example of their differences is shown in the figure below (Fig. 4. and Fig. 5.).

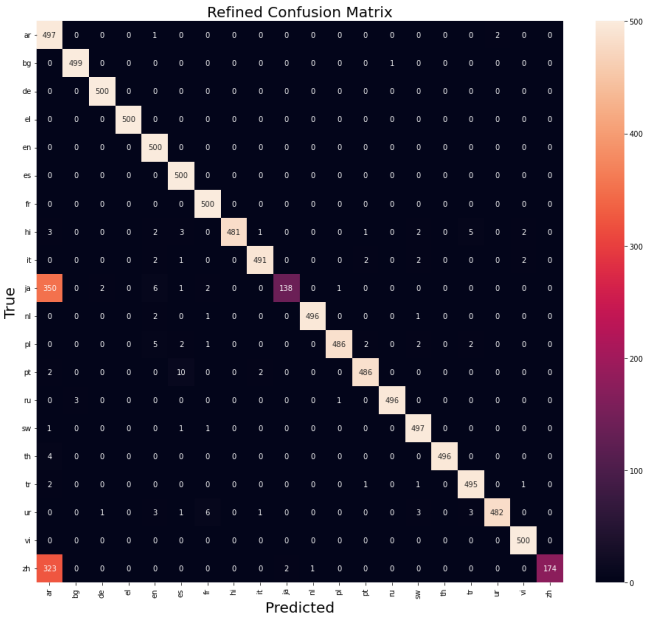

Fig. 4. Confusion matrix of mnb_model

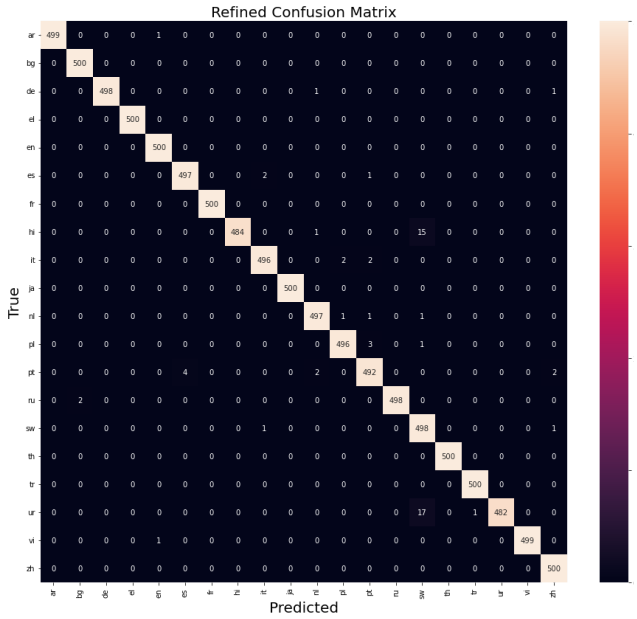

Fig. 5. Confusion matrix of lr_model

IV. Results & Discussion

The two (2) models namely: Logistic Regression (LR) and Multinomial Naive Bayes (MNB) were tested for accuracy, precision, recall, and F1-score. An example of the performance of the models is shown in the figures below (Fig. 6. and Fig. 7.).

```
Classification Report:
              precision    recall  f1-score   support

           0       0.42      0.99      0.59       500
           1       0.99      1.00      1.00       500
           2       0.99      1.00      1.00       500
           3       1.00      1.00      1.00       500
           4       0.96      1.00      0.98       500
           5       0.96      1.00      0.98       500
           6       0.98      1.00      0.99       500
           7       1.00      0.96      0.98       500
           8       0.99      0.98      0.99       500
           9       0.99      0.28      0.43       500
          10       1.00      0.99      0.99       500
          11       1.00      0.97      0.98       500
          12       0.99      0.97      0.98       500
          13       1.00      0.99      0.99       500
          14       0.98      0.99      0.99       500
          15       1.00      0.99      1.00       500
          16       0.98      0.99      0.99       500
          17       1.00      0.96      0.98       500
          18       0.99      1.00      1.00       500
          19       1.00      0.35      0.52       500

    accuracy                           0.92     10000
   macro avg       0.96      0.92      0.92     10000
weighted avg       0.96      0.92      0.92     10000

Accuracy:
Model accuracy score: 0.9214
```

Fig. 6. Classification report of mnb_model

```
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
```

Fig. 7. Classification report of lr_model

The summary of all six models' performance is presented comprehensively in Table VI.

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%

By comparing each model by their accuracy shown in Table VI, we can see that logistic regression model with TfIdfVectorizer(ngram_range=(1,3), analyzer='char'), called lr_model, achieved the highest accuracy compared to the rest of the models, especially the ones using multinomial naive bayes. But to train each of the logistic regression models, it took us around six minutes. While it only took us around 30 seconds to train a multinomial naive bayes model.

From Table VI, we can also imply that logistic regression models work better than multinomial naive bayes when it uses TfIdfVectorizer and CountVectorizer with modified parameters = {ngram_range=(1,3), analyzer='char'}. But multinomial naive bayes win over logistic regression if both models use a CountVectorizer with default parameters.

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. To make our application, we used the Gradio library. We also used the Pickle module to save our model so it can be reused in any application.

For our application, we give users the freedom to choose between our two final models to be used for language identification. Users can then input the text that they want to identify the language of, then our application will use the selected model to identify the language of the text. We made sure to apply preprocessing steps needed on the text input before feeding it to the model. We also made a function to make the output of the model understandable to the user. For example, the output would be "English" instead of its label, "en" (seen in Fig. 8.).

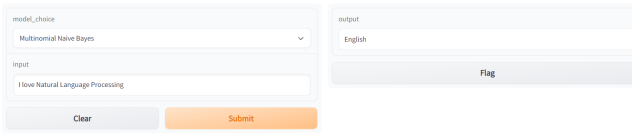

Fig. 8. Example of language identification using mnb_model_tfidf and Gradio

But we also encountered a problem despite the models having 99% accuracy score. The problem is that the model usually outputs the wrong language if the input is short (e.g. only consists of 2 words). We suspect that it might be because the words used for this type of input appeared in various languages and the dataset only has a few samples where the text is short.

V. Conclusion

In this project, we have made an attempt to build a Natural Language Processing (NLP) model that can predict and identify which language the text corresponds to accurately to help the user in identifying what language the information is written in. We used two (2) different algorithms, Logistic Regression and Multinomial Naive Bayes, and built six (6) different models to perform multiclass classification and identify the text's language.

After carefully preprocessing the data and training each model, we found that the logistic regression model with TfIdfVectorizer (with modified parameters) works best as it reaches the accuracy of 99.36%, while the multinomial naive bayes model only reached a maximum accuracy of 98.97% when using the same TfIdfVectorizer.

But we used a transformation to binary technique to use logistic regression for multiclass classification, thus making its models longer to train compared to multinomial naive bayes models. On average, our logistic regression models took around 6 minutes to train, while our multinomial naive bayes only needed around 30 seconds. So the multinomial naive bayes model might come out on top if training speed is a priority as it produces similar results compared to the logistic regression model.

Based on the result of this project, we can conclude that we have successfully built a natural language process model that can identify which language the text corresponds to in an accurate way. But the dataset that we used is still limited to only 20 languages with a few notable languages not available such as Bahasa and Korean. We also encountered a problem where the output is usually not correct for short input. Therefore, we suggest data collection for other languages in the future so we can also utilize this model to identify more languages, and future research on making a model that can identify the language of a short input more accurately.

References

[1] Aly, M. (2005). *Survey on multiclass classification methods*. Neural Netw, 19(1), 9.

[2] Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4, No. 4, p. 738). New York: springer.

[3] *What is logistic regression*? (n.d.). IBM. Retrieved June 14, 2022, from https://www.ibm.com/id-en/topics/logistic-regression

[4] Brownlee, J. (2021, April 26). *One-vs-Rest and One-vs-One for Multi-Class Classification. Machine Learning Mastery*. Retrieved June 14, 2022, from https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

[5] Rennie, J. D., Shih, L., Teevan, J., & Karger, D. R. (2003). *Tackling the poor assumptions of naive bayes text classifiers*. In Proceedings of the 20th international conference on machine learning (ICML-03) (pp. 616-623).

[6] Jha, A. (2021, December 31). *Vectorization Techniques in NLP [Guide]*. Neptune.Ai. Retrieved June 15, 2022, from https://neptune.ai/blog/vectorization-techniques-in-nlp-guide