Language Recognition and First Sentence Prediction

Contents

1. Introduction

Today, we take it for granted that Google can automatically detect a what language a given text is written in and then translate the text with a high degree of accuracy. The goal of my model is to recognize what language a given text is written in and output the language name.

I used a language identification dataset from Kaggle that comprises 22,000 paragraphs of 22 languages and is taken from the original WiLI-2018 Wikipedia language identification benchmark dataset that contians 235,000 paragraphs of 235 langauges. The languages in the dataset I used are:

- English
- Arabic
- French
- Hindi
- Urdu
- Portuguese
- Persian
- Pushto
- Spanish
- Korean
- Tamil
- Turkish
- Estonian
- Russian
- Romanian
- Chinese
- Swedish
- Latin
- Indonesian
- Dutch
- Japanese
- Thai

The data consist of two columns: a natural language text and a categorial label. The data contains 1,000 examples of each language for a total of 22,000 examples.

A secondary goal of this project is building a model that can predict whether a sentence is the first in a paragraph or not.

2. My Approach

Language Recognition

I built four different models by first vectorizing the text data using Count Vectorizer and Tf-idf, and then training Logistic Regression and Naive Bayes models on the training data. I then evaluated the performance of each model using accuracy score, precision, and recall. I also did cross validation for each model to make sure the accuracies I was getting were reasonable.

After finding that the Tf-idf model with Naive Bayes had the highest accuracy on the test data, I reduced overfitting by instantiating models with different alpha values and found that a model with alpha equal to 0.1 was ideal.

Predicting Language of Text Taken from the Internet

I further tested each model's predictive ability by building a function that would take in a paragraph of text in one of the 22 langauges and output its label. The data was taken from various news and other sites on the internet.

Lastly, I built a model that tokenized Chinese and Japanese and merged the document term matrices for these langauges with that of the other languages. I then made predictions and evaluated the model.

First Sentence Prediction

For the second half of this project, I built a model that can identify whether a sentence is the first in its paragraph or not. To do this, I first chose one language to work with, in this case, Chinese. I also chose Chinese because it was the only data which still had punctuation in it.

Next, I needed to create a new self-supervised dataset containing individual sentences and a label indicating if the sentence is the first of a paragraph or not. To this end, I constructed a function that would use spaCy to create spaCy document objects from the paragraphs, split the data up into sentences using the sents method on the document object, label each sentence as the first in its paragraph or not, and construct the new dataframe containing all of the individual sentences from each paragraph and their labels.

To keep the sentneces from being taken out of context during the train-test split, I first did train-test split on the individual paragraphs. Then, I applied the function to create new dataframes for training data and test data.

Because I was dealing with a highly imbalanced dataset, I did random oversampling on the minority class (the first sentneces).

At this point, I was ready to do a latent semantic analysis (LSA) on the data in order to create document vectors that could then be fed to a machine learning model for making predictions. To do this, I used

CountVectorizer and Tf-idf to transform the sentences into a bag of words. I then used Truncated SVD with 75 components to turn the document term matrices into latent semantic analyses.

Once I had the latent semantic analysis vectors, I fit a Logistic Regression model on the data, made predictions on the training and test sets, and evaluated the results using accuracy score, precision, and recall.

In order to prevent overfitting and improve accuracy, I tried creating models with different n_components values for the SVD, namely, 25, 50, and 100. I then fit models to these new datas, made predictions, and evaluated the results.

After getting the results, I found that Tf-idf with LSA with 50 SVD components performed the best. So far, the model has been optimizing for accuracy. But in a classification problem with a miority class, it is more important to optimize for F1 score. So, I created a grid search with the scoring parameter set to f1 in order to optimize C for f1 score instead of accuracy.

3. Findings

Language Recognition

Below is a table showing the accuracy scores of the models. Count Vectorizer with Naive Bayes performed the best.

```
In [10]: %store -r accuracy_df
         accuracy_df
```

Out[10]:

		Training Data Accuracy	Test Data Accuracy
Count Vectorizer	**Logistic Regression**	0.993295	0.939773
	Naive Bayes	0.991023	0.959318
Tf-idf	**Logistic Regression**	0.974034	0.941364
	Naive Bayes	0.983920	0.939545

Below are the cross validation scores.

```
In [3]: %store -r cross_val_scores
        cross_val_scores
```

Out[3]:

		Cross Validation Score
Count Vectorizer	**Logistic Regression**	0.947773
	Naive Bayes	0.955409
Tf-idf	**Logistic Regression**	0.955000
	Naive Bayes	0.954545

Predicting Language of Text Taken from the Internet

Most languages were predicted correctly, however Japanese and Chinese were sometimes predicted as other languages, such as Japanese as Russian or Chinese as Japanese. Sometimes, when a prediction was made on the same language multiple times, the algorithm would predict different languages from one time to the next. I believe the reason for this is because Chinese and Japanese do not contain spaces and therefore cannot be tokenized.

Therefore, I built a model that tokenized Japanese and Chinese and merged that data with the rest of the language data before training the previous best model (Count Vectorizer with Naive Bayes, alpha=0.1) on the data and making predictions. I found that the accuracy was 98% on the test and training data, and the F1 scores for Chinese and Japanese were vastly better than before at around 99%.

First Sentence Prediction

The Tf-idf LSA model performed better than the CountVectorizer model. When I change the SVD components parameter, I found that 50 was the ideal value. Below are the results.

In [7]:
```
%store -r accuracy_f1
accuracy_f1
```

Out[7]:

		Training Data	Test Data
Count Vectorizer	**Accuracy**	0.645660	0.608365
	F1 Score	0.608329	0.558414
Tf-idf	**Accuracy**	0.689931	0.621673
	F1 Score	0.678198	0.619139

In [6]:
```
%store -r tf_svd_25_50_75_100
tf_svd_25_50_75_100
```

Out[6]:

		Training Data	Test Data
25 components	**Accuracy**	0.665799	0.618821
	F1 Score	0.655820	0.630415
50 components	**Accuracy**	0.662326	0.629753
	F1 Score	0.645138	0.638850
75 components	**Accuracy**	0.689931	0.621673
	F1 Score	0.678198	0.619139
100 components	**Accuracy**	0.659549	0.614068
	F1 Score	0.628669	0.565310

When I optimized C for F1 score, I found that the ideal value of C was still 1, so the accuracies didn't change when compared with the previous best model (Tf-idf with 50 components SVD).

In [8]:
```
%store -r tf_optimized
tf_optimized
```

Out[8]:

		Training Data	Test Data
Un-optimized	**Accuracy**	0.662326	0.629753
	F1 Score	0.645138	0.638850
Optimized for F1	**Accuracy**	0.662326	0.629753
	F1 Score	0.645138	0.638850

4. Ideas for Further Research

Language Recognition and Predicting Langauge of Text Taken from the Internet

- Try to solve the same problem but instead of using machine learning, use language dictionaries.
- I could expand the langauges corpus to include more languages.

First Sentence Prediction

- I could try inputting some sentences from paragraphs gathered on the Chinese web and see how well the model predicts the sentence labels.
- Follow the same process but for Wikipedia data in English. To do this, I would need to scrape Wikipedia.

5. Recommendations

Language Recognition and Predicting Langauge of Text Taken from the Internet

The language recognition model could be incorporated into an app that recognizes a language in order to translate it correctly.

First Sentence Prediction

It could be used to build a list of most common first sentences, which could then be characterized.