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. It will take document vectors taken created by a SVD transformation.

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 data. I then evaluated the performance of each model using accuracy score, precision, and recall.

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.

I built a predict function using each of the four trained models and fed each one 22 different paragraphs, one for each language, taken from the Internet, primarily news sites, in order to predict what language the paragraph was.

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 (75 components), 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.

After getting the results, I found that Tf-idf with LSA 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

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.

First Sentence Prediction

The Tf-idf LSA model performed better than the CountVectorizer model. But when I tried to optimize C parameter for f1 score, the model's accuracy and f1 score actually dropped. Results are below.

```
In [16]:  %store -r accuracy_f1
          accuracy_f1
```

Out[16]:

		Training Data	Test Data
Count Vectorizer	**Accuracy**	0.673437	0.533291
	F1 Score	0.645661	0.443936
Tf-idf	**Accuracy**	0.695176	0.592190
	F1 Score	0.684098	0.547264

```
In [17]:  %store -r tf_optimized
          tf_optimized
```

Out[17]:

	Training Data	Test Data

Un-optimized	**Accuracy**	0.695176	0.592190
	F1 Score	0.684098	0.547264
Optimized	**Accuracy**	0.696604	0.580026
	F1 Score	0.688193	0.528058

4. Ideas for Further Research

Language Recognition and Predicting Langauge of Text Taken from the Internet

- Use packages like jieba (Chinese tokenizer) and nagisa (Japanese tokenizer) to tokenize Chinese and Japanese, then combine them with the rest of the tokenized langauge data before doing the modeling.
- 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.