Language Recognition & First Sentence Prediction

Data: 22,000 paragraphs of 22 languages from Wikipedia

Goal 1

Predict the language of a text.

Goal 2

Predict whether a sentence is the first in its paragraph.

Language Recognition: My Approach

Step 1: Vectorize the text data using CountVectorizer and Tf-idf.

Step 2: Split the data into training and test data (75-25%/16,500-5,500 split)

Step 3: Train Logistic Regression and Naive Bayes models on the data.

Step 4: Perform cross validation to find the true accuracy of each model.

Step 5: Perform a grid search on the best model (CountVectorizer with Naive Bayes) to find the parameters that reduce overfitting.

Step 6: Test predictive ability of each model using a paragraph of text from the Internet in the 22 languages.

Step 7: Create a model that predicts well for Chinese and Japanese

Language Recognition: Findings

Training and Test Data Accuracies

		Training Data Accuracy	Test Data Accuracy
Count Vectorizer	Logistic Regression	0.999879	0.950182
	Naive Bayes	0.984242	0.954364
Tf-idf	Logistic Regression	0.986648	0.956136
	Naive Bayes	0.983750	0.954545

Cross Validation Scores of Each Model

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

Chinese (1) and Japanese (8) had lower F1 scores for CountVectorized models.

	precision	recall	f1-score	support
0	1.00	0.98	0.99	255
1	0.79	0.61	0.69	236
2	1.00	0.97	0.99	236
3	0.89	0.98	0.93	266
4	0.97	0.96	0.97	262
5	0.99	0.98	0.99	254
6	1.00	0.98	0.99	259
7	1.00	0.94	0.97	288
8	0.55	0.91	0.68	236
9	1.00	0.94	0.97	237
10	0.97	0.94	0.96	258
11	1.00	0.99	0.99	259
12	0.98	1.00	0.99	236
13	1.00	0.94	0.97	264
14	1.00	0.97	0.98	231

	precision	recall	f1-score	support
0	1.00	1.00	1.00	255
1	0.92	0.56	0.69	236
2	0.97	0.98	0.98	236
3	0.69	1.00	0.81	266
4	0.98	0.97	0.98	262
5	0.95	0.98	0.97	254
6	1.00	0.99	0.99	259
7	1.00	0.95	0.98	288
8	0.74	0.81	0.77	236
9	1.00	0.98	0.99	237
10	0.98	0.91	0.94	258
11	1.00	1.00	1.00	259
12	0.99	0.97	0.98	236
13	1.00	0.95	0.98	264
14	0.99	1.00	0.99	231
15	0.99	0.99	0.99	247

Chinese (1) and Japanese (8) had lower F1 scores for TF-idf models.

	precision	recall	f1-score	support
0	1.00	0.97	0.98	217
1	0.83	0.71	0.76	207
2	1.00	0.98	0.99	209
3	0.81	0.99	0.89	192
4	0.99	0.96	0.97	186
5	0.98	0.98	0.98	199
6	1.00	0.96	0.98	211
7	1.00	0.97	0.99	204
8	0.62	0.94	0.75	188
9	1.00	0.96	0.98	216
10	0.98	0.94	0.96	194
11	1.00	0.99	0.99	203
12	1.00	0.96	0.98	187
13	1.00	0.91	0.96	211
14	1.00	0.99	0.99	210
15	1.00	0.96	0.98	203
16	0.99	0.98	0.99	176
17	1.00	1.00	1.00	203

	precision	recall	f1-score	support
0	1.00	1.00	1.00	217
1	0.97	0.56	0.71	207
2	0.99	0.98	0.98	209
3	0.63	0.99	0.77	192
4	0.99	0.95	0.97	186
5	0.95	0.99	0.97	199
6	1.00	0.96	0.98	211
7	0.98	0.99	0.98	204
8	0.76	0.87	0.81	188
9	1.00	0.98	0.99	216
10	0.99	0.92	0.95	194
11	1.00	1.00	1.00	203
12	0.98	0.96	0.97	187
13	1.00	0.94	0.97	211
14	1.00	1.00	1.00	210
15	1.00	0.99	0.99	203
16	0.98	0.99	0.99	176
17	0.99	1.00	1.00	203
18	1.00	1.00	1.00	212

The Best Model (so far)

The best model was CountVectorizer with Naive Bayes with alpha set to 0.1. The accuracy was 99% on the training data and 96% on the test data.

Predictive ability of models

20 of the languages were predicted correctly by each model. However, all four models failed to predict Japanese and Chinese texts correctly. This makes sense since the F1 scores were lower for these languages.

The cause of the lower F1 scores is due to these languages not being properly tokenized since there are no spaces between words in these languages.

Model that Incorporates Chinese and Japanese

Accuracy score:
0.9875151515151516
Classification report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	750
1	0.99	0.99	0.99	750
2	1.00	1.00	1.00	750
3	0.81	1.00	0.90	750
4	1.00	0.98	0.99	750
5	0.99	0.99	0.99	750
6	1.00	0.98	0.99	750
7	1.00	0.98	0.99	750
8	1.00	0.99	0.99	750
9	1.00	0.99	0.99	750
10	0.99	0.96	0.98	750
11	1.00	1.00	1.00	750
12	1.00	0.98	0.99	750
13	1.00	0.96	0.98	750
14	1.00	0.99	1.00	750
15	0.99	0.99	0.99	750
16	1.00	0.99	0.99	750
17	1.00	1.00	1.00	750

Accuracy score:
0.9803636363636363
Classification report:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	250
1	0.99	0.99	0.99	250
2	0.98	0.99	0.99	250
3	0.78	0.99	0.87	250
4	0.99	0.95	0.97	250
5	0.97	0.99	0.98	250
6	1.00	0.98	0.99	250
7	0.99	0.98	0.99	250
8	1.00	1.00	1.00	250
9	1.00	0.98	0.99	250
10	0.98	0.94	0.96	250
11	1.00	1.00	1.00	250
12	0.98	0.95	0.97	250
13	1.00	0.97	0.98	250
14	0.99	0.98	0.99	250
15	0.98	0.99	0.99	250
16	0.99	0.98	0.98	250
17	0.99	1.00	0.99	250

Language Recognition: Ideas for Further Research

Try using a language dictionary instead of machine learning to solve the problem.

Expand the language corpus to include more languages.

Language Recognition: Recommendations

Incorporate the model into a translation app so that the app would know which language it should translate.

First Sentence Prediction: My Approach

Part 1: Create a new, self-supervised dataset containing the sentences and their labels for one language (Chinese).

Step 1: Split the data into training and test sets so as not to split apart paragraphs later.

Step 2: Create a function that takes a spaCy document object, splits the paragraphs into sentences, and labels each sentence as first in its paragraph or not.

Step 3: Use the function on the training and test data to create training and test dataframes.

Part 2: Random over sampling

Step 1: Because the data is highly imbalanced, use random over sampling on the minority class (first sentences).

Step 2: Do this for both training and test data.

Part 3: Latent Semantic Analysis

Step 1: Transform training and test data sentences into document term matrices using CountVectorizer and Tf-idf.

Step 2: Use Truncated SVD with 75 components to turn the document term matrices into latent semantic analyses.

Part 4: Fit a Logistic Regression model to the data

Step 1: Fit the model to the training data.

Step 2: Make predictions and evaluate the results.

Part 5: Try Different Numbers of Components

Step 1: Change the SVD n_components value to 100, retransform the test and training document term matrices, retrain the model, make predictions, and evaluate the results.

Step 2: Change the SVD n_components value to 50 and repeat the above process.

Step 3: Change the SVD n_components value to 25 and repeat the above process.

Part 6: Optimize the best model (Tf-idf with Logistic Regression) for F1 score

Step 1: Create a grid search with different values of C and the scoring parameter set to ‘f1.’

Step 2: Fit the model to the training data.

Step 3: Make predictions and evaluate the results.

First Sentence Prediction: Findings

CountVectorizer and Tf-idf LSA model results

Tf-idf LSA is the better model.

		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

Accuracies and F1 scores with different SVD components.

50 components had the best accuracy.

		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

Tf-idf LSA un-optimized and optimized model results

The optimal C value is 1, so there is no change to the previous best model.

		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

Conclusion

Considering random guessing would yield about a 20% accuracy, I think this model with 66% accuracy on the test data is very good for a first attempt.

First Sentence Prediction: Ideas for Further Research

Build a function that can take sentences of Chinese text and predict whether they are a first sentence in a paragraph. I could do this on Wikipedia articles.

Follow the same approach but do it using English text scraped from Wikipedia.

First Sentence Prediction: Recommendations

Build a list of most common first sentences and then analyze them to find similarities.