



# 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



## Create four different models

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

Step 2: Train Logistic Regression and Naive Bayes models on the data.

Step 3: Test predictive ability of each model using a paragraph of text from the Internet in the 22 languages.




# Language Recognition: Findings



Based on the accuracy, CountVectorizer with Naive Bayes was the best model

		Training Data Accuracy	Test Data Accuracy
Count Vectorizer	Logistic Regression	0.993750	0.933636
	Naive Bayes	0.991818	0.955682
Tf-idf	Logistic Regression	0.974489	0.954545
	Naive Bayes	0.983239	0.943409



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

	precision	recall	f1-score
0	0.99	0.94	0.97
1	0.77	0.53	0.63
2	1.00	0.99	1.00
3	0.91	0.96	0.93
4	0.98	0.93	0.95
5	0.99	0.99	0.99
6	1.00	0.96	0.98
7	1.00	0.96	0.98
8	0.51	0.94	0.66
9	1.00	0.91	0.95
10	0.94	0.95	0.95
11	1.00	0.98	0.99
12	0.98	0.94	0.96
13	1.00	0.97	0.99
14	0.99	0.97	0.98
15	0.98	0.93	0.95
16	1.00	0.98	0.99

	precision	recall	f1-score
0	1.00	0.99	1.00
1	0.95	0.54	0.69
2	0.99	1.00	0.99
3	0.75	1.00	0.86
4	1.00	0.96	0.98
5	0.98	1.00	0.99
6	1.00	0.99	0.99
7	0.99	0.97	0.98
8	0.67	0.92	0.78
9	1.00	0.97	0.99
10	0.99	0.93	0.96
11	1.00	0.99	1.00
12	1.00	0.94	0.97
13	1.00	0.98	0.99
14	1.00	0.98	0.99
15	0.98	1.00	0.99
16	1.00	0.99	1.00



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

	precision	recall	f1-score	support
0	1.00	0.98	0.99	194
1	0.50	0.98	0.66	188
2	1.00	0.97	0.99	195
3	0.78	0.97	0.86	197
4	0.99	0.98	0.99	199
5	0.96	0.99	0.98	222
6	1.00	0.97	0.99	207
7	1.00	0.98	0.99	209
8	1.00	0.39	0.56	213
9	1.00	0.96	0.98	191
10	0.98	0.90	0.94	204
11	1.00	0.98	0.99	199
12	0.98	0.93	0.95	199
13	1.00	0.94	0.97	196
14	0.99	0.99	0.99	170
15	1.00	0.96	0.98	210
16	1.00	0.96	0.98	198
17	1.00	0.99	0.99	216
18	1.00	0.98	0.99	190
19	1.00	0.98	0.99	194
20	1.00	0.97	0.99	207
21	1.00	0.98	0.99	202

	precision	recall	f1-score	support
0	1.00	0.99	1.00	194
1	0.93	0.54	0.68	188
2	0.98	0.98	0.98	195
3	0.66	0.99	0.80	197
4	0.99	0.98	0.98	199
5	0.94	0.99	0.97	222
6	1.00	0.97	0.98	207
7	0.98	0.98	0.98	209
8	0.99	0.57	0.72	213
9	1.00	0.97	0.98	191
10	0.98	0.91	0.94	204
11	1.00	1.00	1.00	199
12	0.98	0.94	0.96	199
13	1.00	0.95	0.98	196
14	0.59	0.99	0.74	170
15	1.00	0.98	0.99	210
16	0.97	0.98	0.98	198
17	0.99	1.00	0.99	216
18	1.00	0.99	0.99	190
19	0.99	0.98	0.99	194
20	0.99	0.99	0.99	207
21	1.00	0.99	0.99	202



## 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.



# Language Recognition: Ideas for Further Research



**Use packages like Jieba and Nagisa to properly tokenize Chinese and Japanese.**

**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 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.

Step 3: Do this for training and test data.



## **Part 4: Fit a Logistic Regression model to the data**

Step 1: Fit the training data to the model.

Step 2: Make predictions and evaluate the results.



## Part 5: 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 training data to the model.

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	0.673437	0.533291
	F1 Score	0.645661	0.443936
Tf-idf	Accuracy	0.695176	0.592190
	F1 Score	0.684098	0.547264



## Tf-idf LSA un-optimized and optimized model results

Optimizing C for F1 score actually makes the model worse.

		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



## Conclusion

Considering random guessing would yield about a 20% accuracy, I think this model is pretty 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.**