CS585

Natural Language Processing

HW4 – Native Language Identification using pre-trained BERT (Bi-directional Encoder Representations from Transformers)

Yash Patel

A20451170

Ypatel31@hawk.iit.edu

Prof. Derrick Higgins (Instructor)

- # Instructions for running code

1. Requirements:

1) Python 3.6.0
2) Jupyter notebook
3) Libraries:
    - numpy
    - pandas
    - scikit-learn
    - matplotlib
    - seaborn
    - tensorflow==1.14.0
    - protobuf==3.6.0

2. Directory Structure

1) Project directory
    - BERT_BASE_DIR (cloned reps)
    - BERT_DATA_DIR (pre-trained BERT model)
    - bert_input_data
        - eval.txt
        - test.txt
        - train.txt
    - bert_output_data
        - eval.josnlines
        - test.jsonlines
        - train.jsonlines
    - data
        - lang_id_eval.csv
        - lang_id_test.csv
        - lang_id_train.csv
    - model-training-with-BERT-vectors.ipynb

- re-format.sh
- run_bert_fv.sh

3. Steps

1) Clone the BERT repo from git-hub and rename it with BERT_BASE_DIR.
2) Download pre-trained BERT model and rename it with BERT_BASE_DIR.
3) Download handout material which has data of native language and text of speaker. Run re-format.sh (shell script) file.

```
INPUT_DIR=bert_input_data
rm -rf $INPUT_DIR
mkdir -p $INPUT_DIR

sed 's/\([^,]*\),\(.*\)/\2/' ./data/lang_id_eval.csv > ./bert_input_data/eval.csv
sed 1d ./bert_input_data/eval.csv > ./bert_input_data/eval.txt
rm ./bert_input_data/eval.csv



sed 's/\([^,]*\),\(.*\)/\2/' ./data/lang_id_train.csv > ./bert_input_data/train.csv
sed 1d ./bert_input_data/train.csv > ./bert_input_data/train.txt
rm ./bert_input_data/train.csv



sed 's/\([^,]*\),\(.*\)/\2/' ./data/lang_id_test.csv > ./bert_input_data/test.csv
sed 1d ./bert_input_data/test.csv > ./bert_input_data/test.txt
rm ./bert_input_data/test.csv

echo "Done."
echo "Press Enter."
read x
```

This script first removes bert_input_data directory if one is created and then create new one. Then it reformates all the data files and save them in text files in newly create bert_input_data directory in such a way that they have only text of speakers. they don't include native languages and titles.

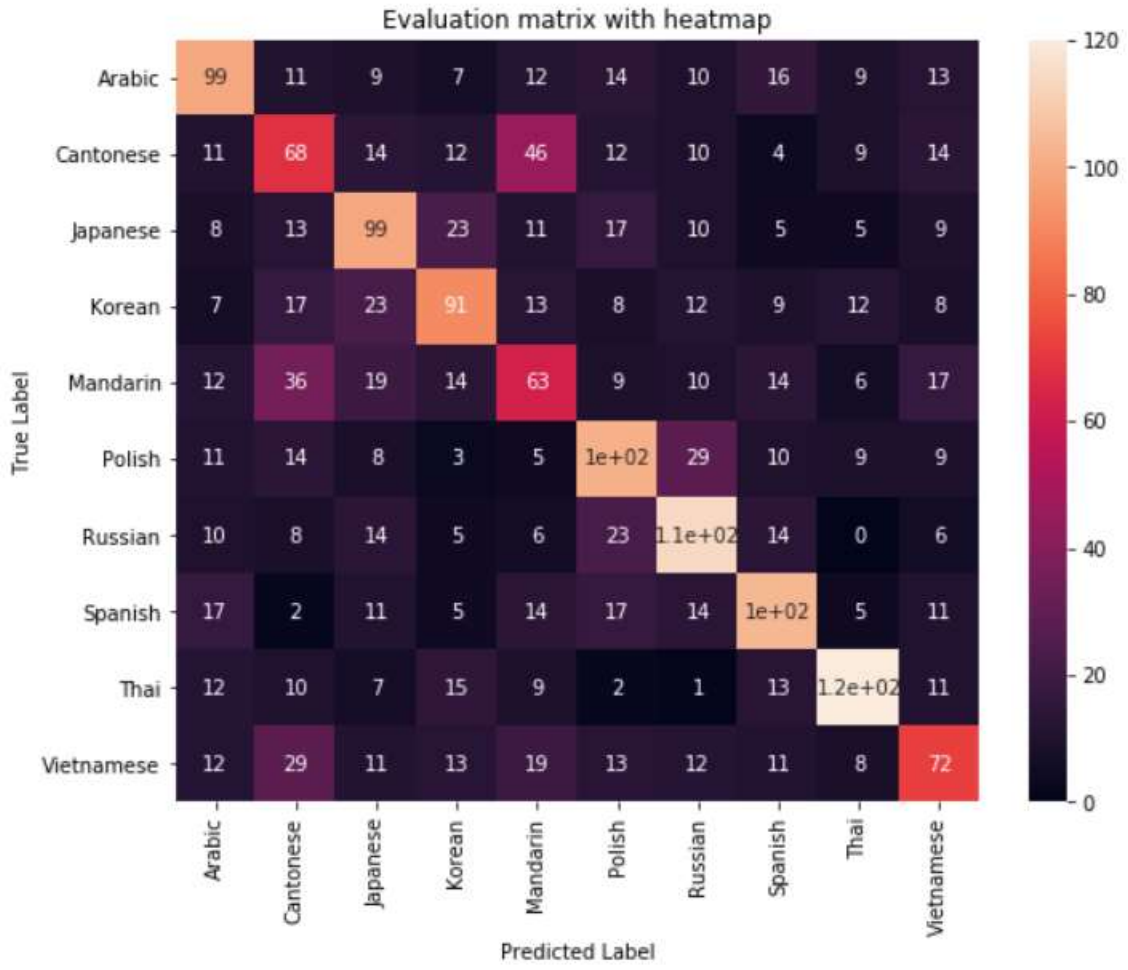
4) Then install tensorflow with 1.14 version and protobuf with 3.6.0 version to run the run_bert_fv.sh (shell script) which runs a extract_features.py python file to generate vectors. After installing run run_bert_fv.sh file. This will generate 7.5 GB of vector data from text of speakers. For input it will take re-formatted text files from bert_input_data directory and it generates vectors (.jsonlines) in bert_output_data directory.
5) Then run model-training-with-BERT-vectors.ipynb in jupyter notebook to train ,test and evaluate your model.

• Summary of evaluation finding

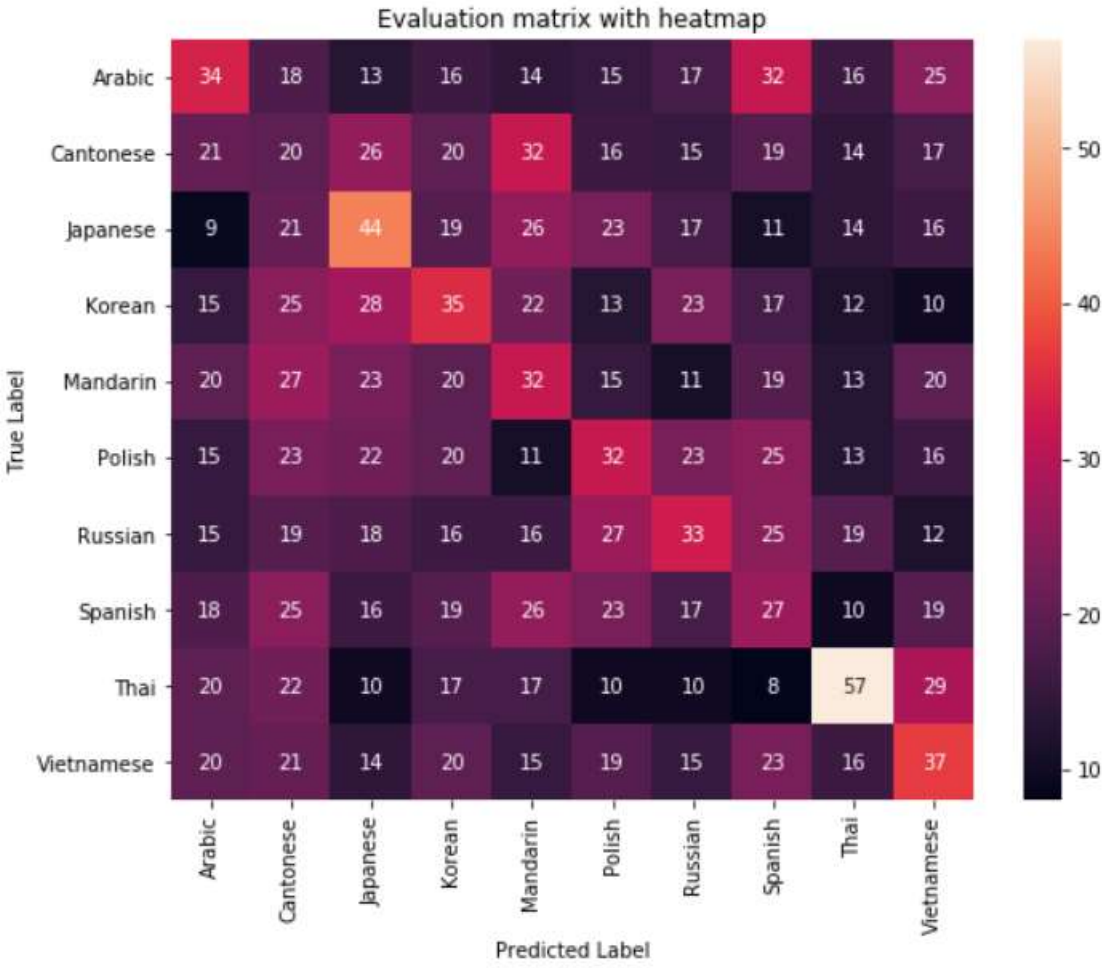
1. Load native language for training and testing from csv files in data directory as targets. Load BERT generated vectors (.jsonlines) for training and testing from bert_output_data directory as features.
2. Train your own model with this training vectors as features and training native language as targets. Here, I used Logistic Regression model, Decision Tree model and Support Vector Machine model.
3. Then we found four performance measurements,
   - Accuracy
     - Logistic Regression:
       - Training accuracy: 0.7348333333333333
       - Testing accuracy: 0.466
     - Decision Tree:
       - Training accuracy: 1.0
       - Testing accuracy: 0.1755
     - Support Vector Machine:

- Training accuracy: 0.7658333333333334
- Testing accuracy: 0.426

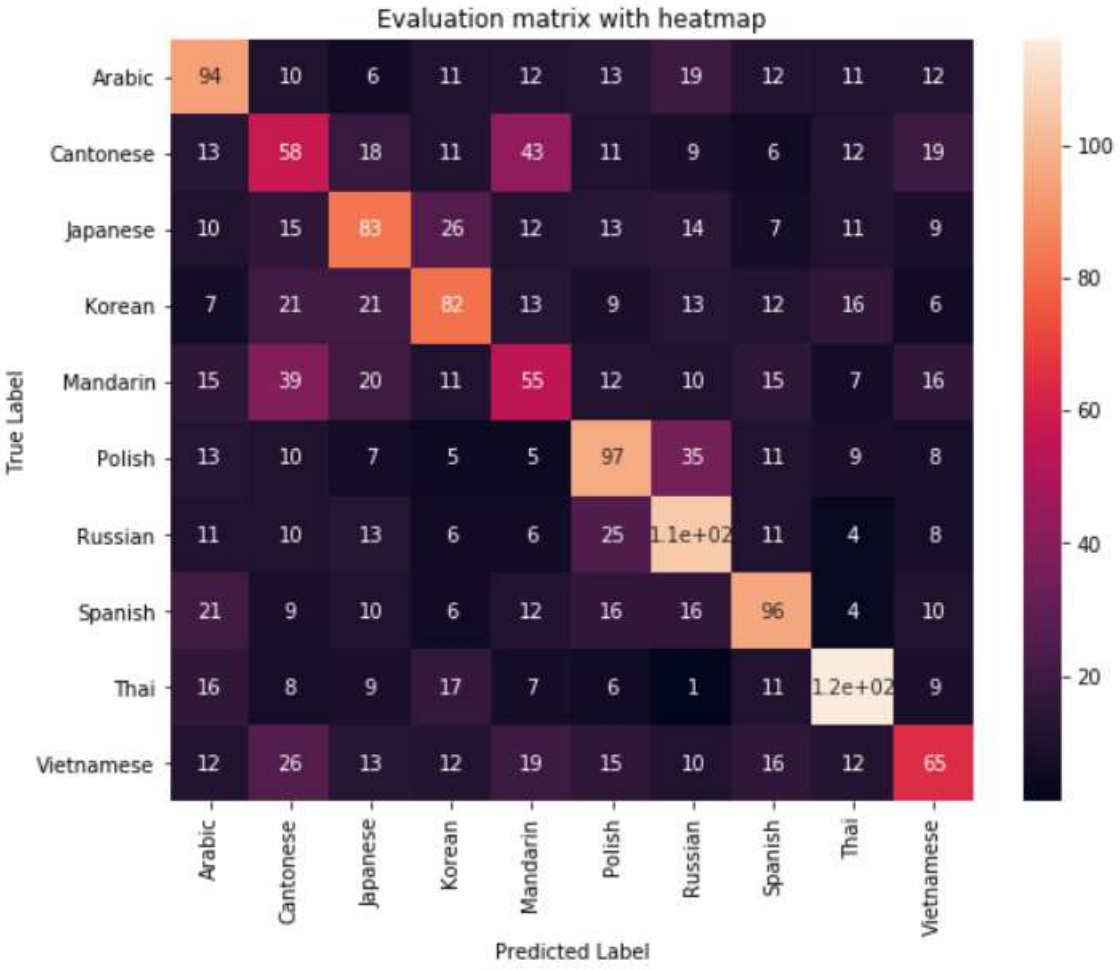
- Confusion Matrix
  - Logistic Regression:



  - Decision Tree:



➢ Support Vector Machine:



- Misclassification for each class
  - Logistic Regression

	language	misclassification	precision	recall	f1 score	support
0	Arabic	10.05	0.497487	0.495	0.496241	200
1	Cantonese	13.60	0.326923	0.340	0.333333	200
2	Japanese	10.85	0.460465	0.495	0.477108	200
3	Korean	10.30	0.484043	0.455	0.469072	200
4	Mandarin	13.60	0.318182	0.315	0.316583	200
5	Polish	10.65	0.470046	0.510	0.489209	200
6	Russian	9.70	0.513514	0.570	0.540284	200
7	Spanish	9.60	0.520000	0.520	0.520000	200
8	Thai	7.15	0.655738	0.600	0.626632	200
9	Vietnamese	11.30	0.423529	0.360	0.389189	200

  - Decision Tree

	language	misclassification	precision	recall	f1 score	support
0	Arabic	15.95	0.181818	0.170	0.175711	200
1	Cantonese	19.05	0.090498	0.100	0.095012	200
2	Japanese	16.30	0.205607	0.220	0.212560	200
3	Korean	16.60	0.173267	0.175	0.174129	200

4	Mandarin	17.35	0.151659	0.160	0.155718	200
5	Polish	16.45	0.165803	0.160	0.162850	200
6	Russian	15.75	0.182320	0.165	0.173228	200
7	Spanish	17.60	0.131068	0.135	0.133005	200
8	Thai	13.50	0.309783	0.285	0.296875	200
9	Vietnamese	16.35	0.184080	0.185	0.184539	200

- Support Vector Machine

	language	misclassification	precision	recall	f1 score	support
0	Arabic	11.20	0.443396	0.470	0.456311	200
1	Cantonese	14.50	0.281553	0.290	0.285714	200
2	Japanese	11.70	0.415000	0.415	0.415000	200
3	Korean	11.15	0.438503	0.410	0.423773	200
4	Mandarin	13.70	0.298913	0.275	0.286458	200
5	Polish	11.15	0.447005	0.485	0.465228	200
6	Russian	11.05	0.454936	0.530	0.489607	200
7	Spanish	10.25	0.487310	0.480	0.483627	200
8	Thai	8.50	0.574257	0.580	0.577114	200
9	Vietnamese	11.60	0.401235	0.325	0.359116	200

- Misclassification for each pair of classes

- Logistic Regression:

	True Label	Predicted Label	Misclassification
0	Arabic	Cantonese	11
1	Arabic	Japanese	8
2	Arabic	Korean	7
3	Arabic	Mandarin	12
4	Arabic	Polish	11
5	Arabic	Russian	10
6	Arabic	Spanish	17
7	Arabic	Thai	12
8	Arabic	Vietnamese	12
9	Cantonese	Arabic	11
10	Cantonese	Japanese	13
11	Cantonese	Korean	17
12	Cantonese	Mandarin	36
13	Cantonese	Polish	14
14	Cantonese	Russian	8
15	Cantonese	Spanish	2
16	Cantonese	Thai	10
17	Cantonese	Vietnamese	29
18	Japanese	Arabic	9
19	Japanese	Cantonese	14
20	Japanese	Korean	23
21	Japanese	Mandarin	19
22	Japanese	Polish	8
23	Japanese	Russian	14
24	Japanese	Spanish	11
25	Japanese	Thai	7
26	Japanese	Vietnamese	11
27	Korean	Arabic	7
28	Korean	Cantonese	12
29	Korean	Japanese	23
30	Korean	Mandarin	14

31	Korean	Polish	3
32	Korean	Russian	5
33	Korean	Spanish	5
34	Korean	Thai	15
35	Korean	Vietnamese	13
36	Mandarin	Arabic	12
37	Mandarin	Cantonese	46
38	Mandarin	Japanese	11
39	Mandarin	Korean	13
40	Mandarin	Polish	5
41	Mandarin	Russian	6
42	Mandarin	Spanish	14
43	Mandarin	Thai	9
44	Mandarin	Vietnamese	19
45	Polish	Arabic	14
46	Polish	Cantonese	12
47	Polish	Japanese	17
48	Polish	Korean	8
49	Polish	Mandarin	9
50	Polish	Russian	23
51	Polish	Spanish	17
52	Polish	Thai	2
53	Polish	Vietnamese	13
54	Russian	Arabic	10
55	Russian	Cantonese	10
56	Russian	Japanese	10
57	Russian	Korean	12
58	Russian	Mandarin	10
59	Russian	Polish	29
60	Russian	Spanish	14
61	Russian	Thai	1
62	Russian	Vietnamese	12
63	Spanish	Arabic	16
64	Spanish	Cantonese	4
65	Spanish	Japanese	5
66	Spanish	Korean	9
67	Spanish	Mandarin	14
68	Spanish	Polish	10
69	Spanish	Russian	14
70	Spanish	Thai	13
71	Spanish	Vietnamese	11
72	Thai	Arabic	9
73	Thai	Cantonese	9
74	Thai	Japanese	5
75	Thai	Korean	12
76	Thai	Mandarin	6
77	Thai	Polish	9
78	Thai	Russian	0
79	Thai	Spanish	5
80	Thai	Vietnamese	8
81	Vietnamese	Arabic	13
82	Vietnamese	Cantonese	14
83	Vietnamese	Japanese	9
84	Vietnamese	Korean	8
85	Vietnamese	Mandarin	17
86	Vietnamese	Polish	9
87	Vietnamese	Russian	6

88	Vietnamese	Spanish	11
89	Vietnamese	Thai	11

➢ Decision Tree:

	True Label	Predicted Label	Misclassification
0	Arabic	Cantonese	21
1	Arabic	Japanese	9
2	Arabic	Korean	15
3	Arabic	Mandarin	20
4	Arabic	Polish	15
5	Arabic	Russian	15
6	Arabic	Spanish	18
7	Arabic	Thai	20
8	Arabic	Vietnamese	20
9	Cantonese	Arabic	18
10	Cantonese	Japanese	21
11	Cantonese	Korean	25
12	Cantonese	Mandarin	27
13	Cantonese	Polish	23
14	Cantonese	Russian	19
15	Cantonese	Spanish	25
16	Cantonese	Thai	22
17	Cantonese	Vietnamese	21
18	Japanese	Arabic	13
19	Japanese	Cantonese	26
20	Japanese	Korean	28
21	Japanese	Mandarin	23
22	Japanese	Polish	22
23	Japanese	Russian	18
24	Japanese	Spanish	16
25	Japanese	Thai	10
26	Japanese	Vietnamese	14
27	Korean	Arabic	16
28	Korean	Cantonese	20
29	Korean	Japanese	19
30	Korean	Mandarin	20
31	Korean	Polish	20
32	Korean	Russian	16
33	Korean	Spanish	19
34	Korean	Thai	17
35	Korean	Vietnamese	20
36	Mandarin	Arabic	14
37	Mandarin	Cantonese	32
38	Mandarin	Japanese	26
39	Mandarin	Korean	22
40	Mandarin	Polish	11
41	Mandarin	Russian	16
42	Mandarin	Spanish	26
43	Mandarin	Thai	17
44	Mandarin	Vietnamese	15
45	Polish	Arabic	15
46	Polish	Cantonese	16
47	Polish	Japanese	23

48	Polish	Korean	13
49	Polish	Mandarin	15
50	Polish	Russian	27
51	Polish	Spanish	23
52	Polish	Thai	10
53	Polish	Vietnamese	19
54	Russian	Arabic	17
55	Russian	Cantonese	15
56	Russian	Japanese	17
57	Russian	Korean	23
58	Russian	Mandarin	11
59	Russian	Polish	23
60	Russian	Spanish	17
61	Russian	Thai	10
62	Russian	Vietnamese	15
63	Spanish	Arabic	32
64	Spanish	Cantonese	19
65	Spanish	Japanese	11
66	Spanish	Korean	17
67	Spanish	Mandarin	19
68	Spanish	Polish	25
69	Spanish	Russian	25
70	Spanish	Thai	8
71	Spanish	Vietnamese	23
72	Thai	Arabic	16
73	Thai	Cantonese	14
74	Thai	Japanese	14
75	Thai	Korean	12
76	Thai	Mandarin	13
77	Thai	Polish	13
78	Thai	Russian	19
79	Thai	Spanish	10
80	Thai	Vietnamese	16
81	Vietnamese	Arabic	25
82	Vietnamese	Cantonese	17
83	Vietnamese	Japanese	16
84	Vietnamese	Korean	10
85	Vietnamese	Mandarin	20
86	Vietnamese	Polish	16
87	Vietnamese	Russian	12
88	Vietnamese	Spanish	19
89	Vietnamese	Thai	29

➢ Support Vector Machine:

	True Label	Predicted Label	Misclassification
0	Arabic	Cantonese	13
1	Arabic	Japanese	10
2	Arabic	Korean	7
3	Arabic	Mandarin	15
4	Arabic	Polish	13
5	Arabic	Russian	11
6	Arabic	Spanish	21
7	Arabic	Thai	16
8	Arabic	Vietnamese	12

9	Cantonese	Arabic	10
10	Cantonese	Japanese	15
11	Cantonese	Korean	21
12	Cantonese	Mandarin	39
13	Cantonese	Polish	10
14	Cantonese	Russian	10
15	Cantonese	Spanish	9
16	Cantonese	Thai	8
17	Cantonese	Vietnamese	26
18	Japanese	Arabic	6
19	Japanese	Cantonese	18
20	Japanese	Korean	21
21	Japanese	Mandarin	20
22	Japanese	Polish	7
23	Japanese	Russian	13
24	Japanese	Spanish	10
25	Japanese	Thai	9
26	Japanese	Vietnamese	13
27	Korean	Arabic	11
28	Korean	Cantonese	11
29	Korean	Japanese	26
30	Korean	Mandarin	11
31	Korean	Polish	5
32	Korean	Russian	6
33	Korean	Spanish	6
34	Korean	Thai	17
35	Korean	Vietnamese	12
36	Mandarin	Arabic	12
37	Mandarin	Cantonese	43
38	Mandarin	Japanese	12
39	Mandarin	Korean	13
40	Mandarin	Polish	5
41	Mandarin	Russian	6
42	Mandarin	Spanish	12
43	Mandarin	Thai	7
44	Mandarin	Vietnamese	19
45	Polish	Arabic	13
46	Polish	Cantonese	11
47	Polish	Japanese	13
48	Polish	Korean	9
49	Polish	Mandarin	12
50	Polish	Russian	25
51	Polish	Spanish	16
52	Polish	Thai	6
53	Polish	Vietnamese	15
54	Russian	Arabic	19
55	Russian	Cantonese	9
56	Russian	Japanese	14
57	Russian	Korean	13
58	Russian	Mandarin	10
59	Russian	Polish	35
60	Russian	Spanish	16
61	Russian	Thai	1
62	Russian	Vietnamese	10

63	Spanish	Arabic	12
64	Spanish	Cantonese	6
65	Spanish	Japanese	7
66	Spanish	Korean	12
67	Spanish	Mandarin	15
68	Spanish	Polish	11
69	Spanish	Russian	11
70	Spanish	Thai	11
71	Spanish	Vietnamese	16
72	Thai	Arabic	11
73	Thai	Cantonese	12
74	Thai	Japanese	11
75	Thai	Korean	16
76	Thai	Mandarin	7
77	Thai	Polish	9
78	Thai	Russian	4
79	Thai	Spanish	4
80	Thai	Vietnamese	12
81	Vietnamese	Arabic	12
82	Vietnamese	Cantonese	19
83	Vietnamese	Japanese	9
84	Vietnamese	Korean	6
85	Vietnamese	Mandarin	16
86	Vietnamese	Polish	8
87	Vietnamese	Russian	8
88	Vietnamese	Spanish	10
89	Vietnamese	Thai	9

- Summary: For text classification we can use decision tree, support vector machine, logistic regression, neural network etc. In test data there are 2000 data instance and 200 per language class. According to me this logistic regression model might optimum to find native language of the speaker as per the accuracy and misclassification rate. Accuracy on test data using logistic regression is around 46%. Misclassification for each language class is around 10% (±3%). Accuracy on test data using decision tree is around 17%. Misclassification for each language class is around 15% (±3%). Accuracy on test data using support vector machine is around 42%. Misclassification for each language class is around 11% (±3%). So, true label probability for each class is 0.1. but when you test on

logistic regression model, it decreases to 0.05 (approx.) which is 50% of true label probability. On decision tree, it decreases to 0.02 (approx.) which is 20% of true label probability. On support vector machine, it decreases to 0.04 (approx.) which is 40% of true label probability. Here, decision tree has low accuracy. Reason for that might be it defines its depth by its own or large number of features or it predicts for very similar data but here text might change person to person or overfitting or complex rules. Here, support vector machine also has low accuracy. Reason for that might be it's difficult for it to know the pattern it relies on.

4. **Further improvements:** We can train others models and evaluate it. Like, we can train neural network with different numbers of hidden layer, neurons per layers, activation functions, etc. Any difference in hidden layer or neurons per layers or activation functions can give different accuracy. And we can choose best of them. We can define decision tree's depth. It can give us different accuracy as we use different type of encoding like one-hot, binary, categorical, numeric and we can use on of the best encoding method as per the accuracy.

- ## Notes on challenges

1. How to define depth of the decision tree based on the features type (categorical or nominal).
2. How to reduce dimensions of feature vector if possible, in this case.
3. How to know that which kind of model understand which kind of patterns to learn data.

4. How to decide the range of hidden layers, neurons and activation function for neural network architecture based on features and targeted classes dimensions, classification and regression process.