Package requirements

Python 3.7

- numpy>=1.17.4
- pandas>=0.25.3
- scikit-learn==0.21.3
- tensorflow==1.15.0

Included files

- run_bert_fv.sh: Supplied script for generating feature vectors that has been customized to a specific directory structure and file nameing scheme
- bert_prep.sh: Bash script that separates a two-column comma-separated file into two separate files by column and trims any leading or trailing quotation marks, optionally removing the first line
- model.py: Python code that performs the analysis using the *tensorflow* and *sklearn* modules

Required directory tree

```
.
  |-bert
     |-models
         |-uncased_L-12_H-768_A-12
             |-bert_input_data
                |-prep.sh
                |-lang_id_eval.csv
                |-lang_id_test.csv
                |-lang_id_train.csv
                |-eval_data.txt
                |-eval_label.txt
                |-test_data.txt
                |-test_label.txt
                |-train_data.txt
                |-train_label.txt
             |-bert_output_data
                |-eval.jsonlines
                |-test.jsonlines
                |-train.jsonlines
  |-model.py
  |-logger.py
  |-run_bert_fv.sh
```

Instructions to reproduce

1. Create the 'bert_input_data' and 'bert_output_data' directories as shown in the required tree.
2. Move the BERT files, provided data files, and attached files to their assigned places in the tree.
3. Run 'prep.sh' three times (once for train, test, and eval) with the 'output_file_prefix' option set to 'train', 'test', and 'eval' respectively. The output files will be named [dataset]_label.txt and [dataset]_data.txt and saved to the locations shown in the tree.
4. Run 'run_bert_fv.sh' to extract the feature vectors, which will be named [dataset].jsonlines and saved to the locations shown in the tree.
5. Run the Python analysis by calling 'python model.py' from the base of the tree; the seed is set for the relevant modules before execution so the results should be similar to those shown below.

Model: neural network

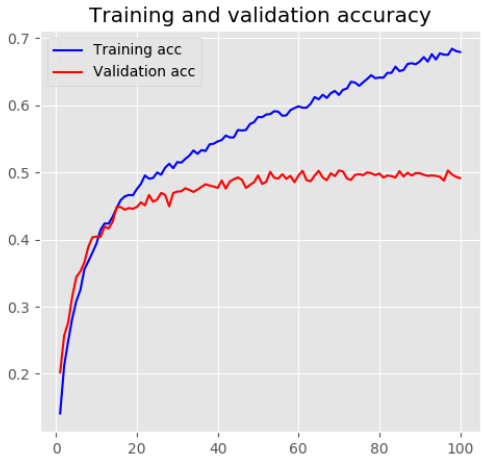
Model specification

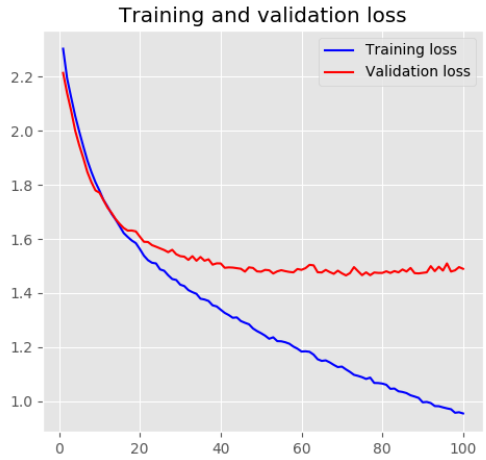
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	98432
activation (Activation: relu)	(None, 128)	0
dropout (Dropout: 0.2)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1290
activation_1 (Activation: softmax)	(None, 10)	0

optimizer=SGD(lr=0.001, momentum=0.9)

100 epochs, batch_size=32

Model fit results





Training Accuracy: 0.7473

Testing Accuracy: 0.4805

Confusion matrix (rows=actual, columns=predicted):

	Arabic	Cantonese	Japanese	Korean	Mandarin	Polish	Russian	Spanish	Thai	Vietnamese
Arabic	**105**	4	10	8	11	18	5	15	12	12
Cantonese	15	**57**	19	18	46	13	4	2	14	12
Japanese	11	8	**107**	23	14	11	9	2	10	5
Korean	7	12	30	**95**	9	5	13	2	14	13
Mandarin	9	31	15	21	**73**	5	12	10	9	15
Polish	11	4	9	9	5	**104**	31	14	9	4
Russian	8	4	12	6	8	28	**118**	7	2	7
Spanish	23	1	11	5	7	19	20	**95**	5	14
Thai	17	8	6	12	4	2	3	4	**127**	17
Vietnamese	20	7	12	16	14	16	13	6	16	**80**

Pairwise sensitivity by class (rows=actual, columns=predicted):

	Arabic	Cantonese	Japanese	Korean	Mandarin	Polish	Russian	Spanish	Thai	Vietnamese
Arabic	**0.465**	0.018	0.044	0.035	0.049	0.080	0.022	0.066	0.053	0.053
Cantonese	0.110	**0.419**	0.140	0.132	0.338	0.096	0.029	0.015	0.103	0.088
Japanese	0.048	0.035	**0.463**	0.100	0.061	0.048	0.039	0.009	0.043	0.022
Korean	0.033	0.056	0.141	**0.446**	0.042	0.023	0.061	0.009	0.066	0.061
Mandarin	0.047	0.162	0.079	0.110	**0.382**	0.026	0.063	0.052	0.047	0.079
Polish	0.050	0.018	0.041	0.041	0.023	**0.471**	0.140	0.063	0.041	0.018
Russian	0.035	0.018	0.053	0.026	0.035	0.123	**0.518**	0.031	0.009	0.031
Spanish	0.146	0.006	0.070	0.032	0.045	0.121	0.127	**0.605**	0.032	0.089
Thai	0.078	0.037	0.028	0.055	0.018	0.009	0.014	0.018	**0.583**	0.078
Vietnamese	0.112	0.039	0.067	0.089	0.078	0.089	0.073	0.034	0.089	**0.447**

Pairwise recall by class (rows=actual, columns=predicted):

	Arabic	Cantonese	Japanese	Korean	Mandarin	Polish	Russian	Spanish	Thai	Vietnamese
Arabic	**0.525**	0.020	0.050	0.040	0.055	0.090	0.025	0.075	0.060	0.060
Cantonese	0.075	**0.285**	0.095	0.090	0.230	0.065	0.020	0.010	0.070	0.060
Japanese	0.055	0.040	**0.535**	0.115	0.070	0.055	0.045	0.010	0.050	0.025
Korean	0.035	0.060	0.150	**0.475**	0.045	0.025	0.065	0.010	0.070	0.065
Mandarin	0.045	0.155	0.075	0.105	**0.365**	0.025	0.060	0.050	0.045	0.075
Polish	0.055	0.020	0.045	0.045	0.025	**0.520**	0.155	0.070	0.045	0.020
Russian	0.040	0.020	0.060	0.030	0.040	0.140	**0.590**	0.035	0.010	0.035
Spanish	0.115	0.005	0.055	0.025	0.035	0.095	0.100	**0.475**	0.025	0.070
Thai	0.085	0.040	0.030	0.060	0.020	0.010	0.015	0.020	**0.635**	0.085
Vietnamese	0.100	0.035	0.060	0.080	0.070	0.080	0.065	0.030	0.080	**0.400**

Postmortem

Model accuracy and loss followed the expected pattern over epochs; there didn't appear to be any indication of convergence issues or model overtraining. However, the learning rate did need to be lowered from the default (0.01) to help improve convergence. The relatively low accuracy can in part be attributed to the data being minimally preprocessed; only the leading and trailing quotation marks were stripped, no other changes were made.

Concerning the prediction results, misclassifications tended to occur most often between languages primarily spoken by countries regionally close to one another or that share the same language family.The model had the most trouble differentiating texts written by those with Mandarin and Cantonese as their native languages; similar difficulties were observed to a lesser degree between the other asian languages (with the exception of thai and vietnamese) and between the slavic languages.