Github link - https://github.com/drindustrial/Name-classification

Name classification

1. Introduction

Name classification models could be used to predict gender of a person who did not inform about his gender but filled out his name, we can get his gender by knowing only his name and work with this information, for example make better suggestions by knowing more information about person.

In this work was compared four different models for names classification by gender. All models can classify the gender by reading the name using a character level model.

2. Methodology

For making name classification model first we should extract the data and preprocess it to machine-readable format, in such a way that optimization algorithms could work with this data. Second step is picking the models, make train, and test dataset for the models, chose optimization algorithms and start training the models. For the better comparing results we should chose different type of models with their pros and cons then train and evaluate them to see where and why chose one model or another.

3. Data

2.1 Data representation

The dataset is a collection of strings of variable length. Data represents by table with two columns (name and gender) and 83 288 rows.

2.1 Data preprocessing

First part of preprocessing was filtered out names what belongs to two genders, in this part was filtered out 13 112 rows from train dataset and 826 rows from test dataset. After the filtering all names was transformed to machine-readable format, this format is as follows

```
[ 5 38 35 52 27 28 31 46 34  0  0  0  0  0  0]
```

Where each number is unique integer identifier for every charter what represent in train dataset and 0 is placed after each word less than 15 characters so that each sample has a length of 15.

3. RNN models

Was trained few RNN models with different hyperparameters, the following results were obtained:

Model 1 - LSTM layer (size 14), 1 Dense Layer (size 8)

Model 2 - LSTM layer (size 15), 1 Dense Layer (size 2)

Model 3. GRU layer (size 18), 1 Dense Layer (size 18)

Parameter	Model 1	Model 2	Model 3
# of LSTM layers	1	1	0
# of GRU layers	0	0	1
# of Dense layers	1	1	1
# of epochs	80	85	85
L1 regularization	2e-4	2e-4	2e-4
L2 regularization	2e-4	2e-4	2e-4
Dropout	0.2	0.2	0.15
# of parameters	1514	1560	1976
Train accuracy	0.8736	0.8632	0.8738
Test accuracy	0.8245	0.8181	0.8239

To avoid too much overfitting was applied l1 l2 regularization and Dropout layers.

4. Logistic Regression

In case, if we have low computational resources we can use logistic regression. Logistic regression have train accuracy 0.8030 and test accuracy 0.7694 in this task.

5. Feed forward models

For comparison RNN model with other types of models was trained and the following results were obtained:

Model 2.1 - 1 Dense layer (size 15)

Model 2.2 - 1 Global Max Pooling, 2 Dense layer (size 35)

	Model 2.1	Model 2.2
# of Pooling layers	0	1
# of Dense layers	1	2
# of epochs	80	50
L1 regularization	3e-4	2e-4
L2 regularization	3e-4	2e-4
Dropout	0.2	0.15
# of parameters	1421	1406

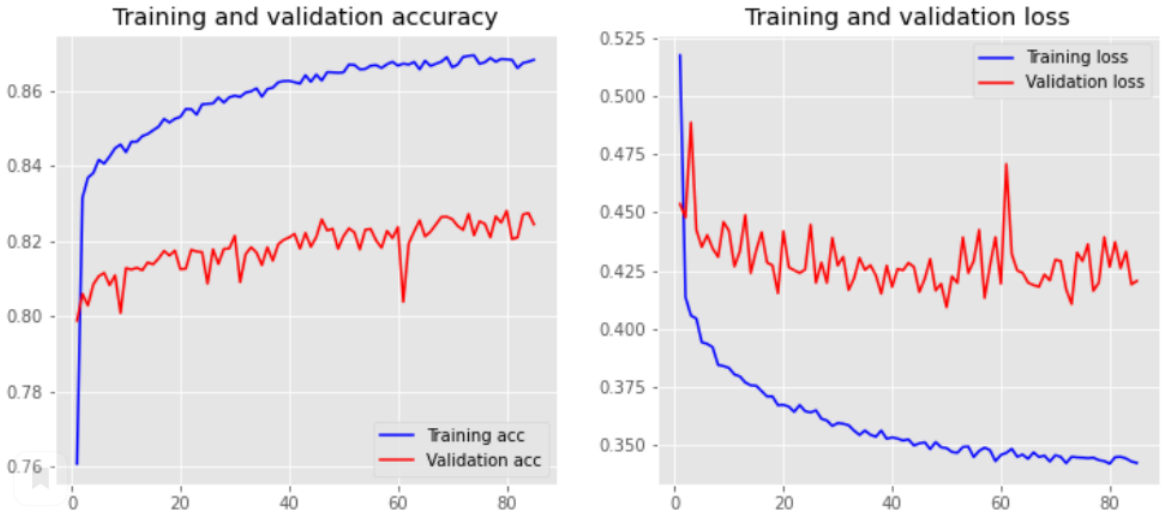
Train accuracy	0.8382	0.7144
Test accuracy	0.8013	0.6920

All three of these models have the same number of parameters as RNN models – 1500 +- 100, to avoid too much overfitting was applied l1 l2 regularization and Dropout layers. From those four models the best is fully connected NN.

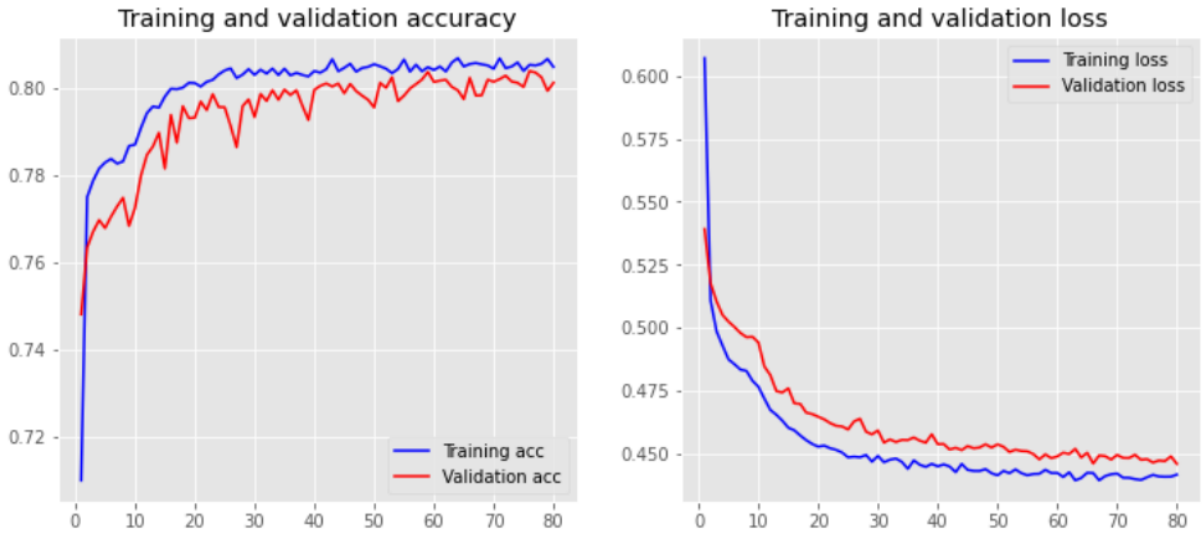
6. Graphs

In this paragraph presented graphs for three models:

RNN model with 1 LSTM layer (model 1 from 3 paragraph)



Feedforward model (model 2.1 from 4 paragraph)



Feedforward model with Global Max Pooling (model 2.2 from 4 paragraph)

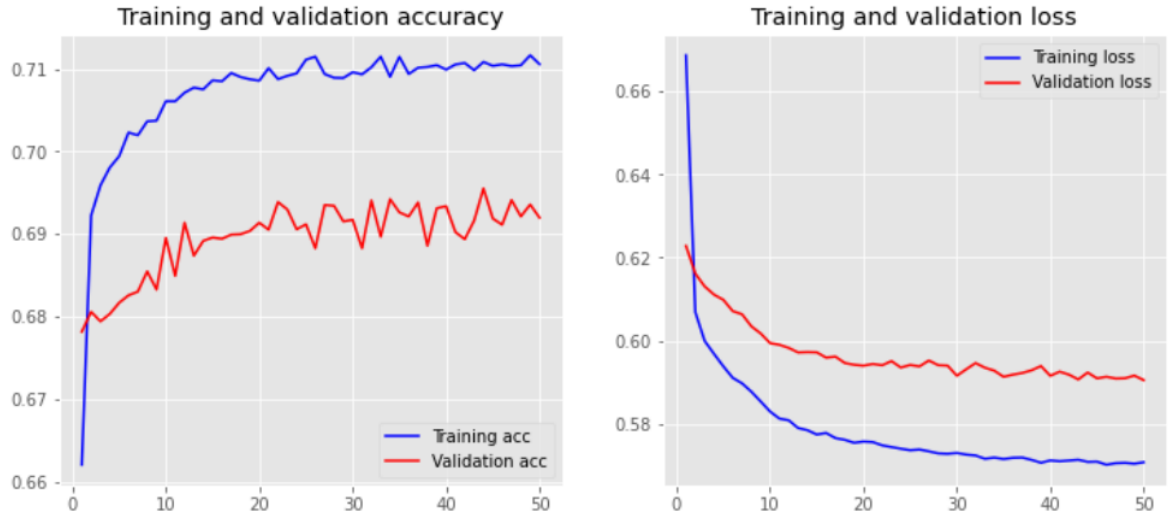
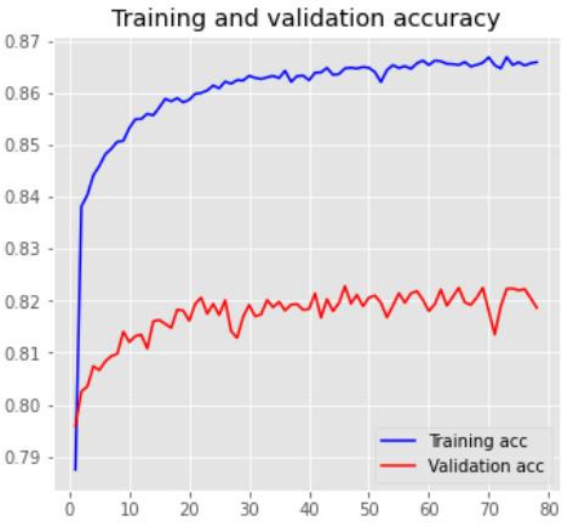


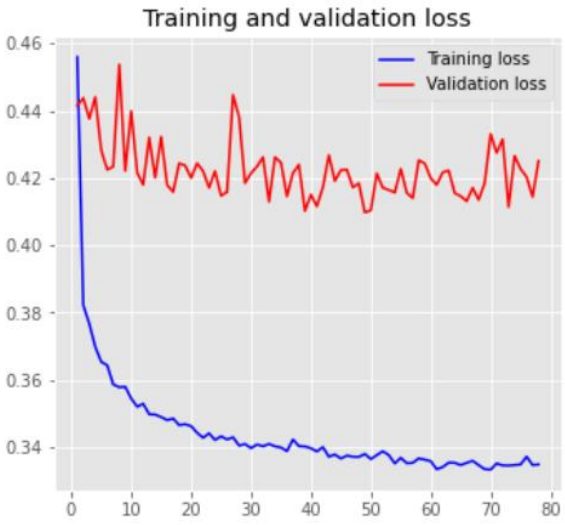
Table for comparison these three models:

Model	Train accuracy	Test accuracy
1 LSTM	0.8736	0.8245
2.1 Dense	0.8382	0.8013
2.2 Max pooling and Dense	0.7144	0.6920

From this table we can see that LSTM based model is better in terms of accuracy. Nevertheless, if we reduce number of parameters LSTM model will compute faster and still have same accuracy as fully connected model.

Here is model with almost twice less parameters, 1 GRU layer (10 size). Training Accuracy: 0.8682 and Testing Accuracy: 0.8187





7. Conclusion

From the results of each model, we can make these assumptions:

Logistic regression could be used there we work with big data or we have a little limit on computation resources. For the tasks there we have enough resources and accuracy of logistic regression not enough we can use RNN model with GRU layer. And if care only about accuracy we can use RNN model with LSTM layers and big number of parameter’s