# Personality Prediction using MBTI Dataset

Nalisha Rathod(gs9440)

Sujata Gorai (hc6837)

1. Overview of the Problem

A.1 Literature Review

Machine learning for predicting personalities has been a significant focus in Natural Lan- guage Processing research over the past few years. This is in part due to the availability of large corpora of online social interactions. (MBTI) Myers-Briggs Personality Type Daatset published by Kaggle competitions which have allowed researchers to gain access to datasets with 8600 training examples of post by different people around the world. In terms of methods, most research takes a text classification approach for personality prediction.

Carl Jungs theory of different people having different state of mind types states that random disparity in behavior is accounted for by the way people use judgement and perception and this is what Myers-Briggs Type Indicator (MBTI) is based on.

There are 16 personality types across four dimensions. Extraversion (E) vs Introversion (I) is a measure of how much an individual prefers their outer or inner world. Sensing (S) vs Intuition (N) differentiates those that process information through the five senses versus impressions through patterns. Thinking (T) vs Feeling (F) is a measure of preference for objective principles and facts versus weighing the points of view of others.

Finally, Judging (J) vs Perceiving (P) differentiates those that prefer planned and ordered life versus flexible and spontaneous. Note that these measures are not binary but rather on a continuum.

Mohammad Hossein Amirhosseini and Hassan Kazemian in march [2020](#ML1) “**Machine Learning Approach to Personality Type Prediction Based on the Myers–Briggs Type Indicator**” used natural language processing toolkit (NLTK) and XGBoost approach to discover personality type prediction based on MBTI personality type indicator. They have used Pandas, NumPy, re, Seaborn, Matplotlib and Sklearn .

# Zhanchen Reg , Qiang Shen , Xiaolei Diao ,Hao Xu in January 2021[[3]](#DLlink) “A sentiment-aware deep learning approach for personality detection from text“ used different deep learning models. Bidirectional Encoder Representation from Transformers (BERT) was used to generate sentence-level embedding for text semantic extraction. In addition, a sentiment dictionary is used for text sentiment analysis in order to consider sentiment information. Finally, input these semantic information and emotional information into the neural network to construct an automatic personality detection model

Machine techniques such as Naïve Bayes, kNN, mLR, Gaussian Process is given by Agarwal ([2014](#ML2))” **Personality detection from text: a review**”. Post 2014, deep neural network architectures and models picked up and started beating the state-of-the-art accuracy of these models.

B. Dataset Overview

C. Methods

Import the Dataset

In each model, we have imported dataset using python. The dataset is on .csv format.

Data Pre-processing

We have done several methods to preprocess the comments before passing to the training model.

* Replaced "|||" from text with " " Join all texts written by 1 Person
* Removed Links from text and replace them with 'Link'
* Removed punctuation from Text

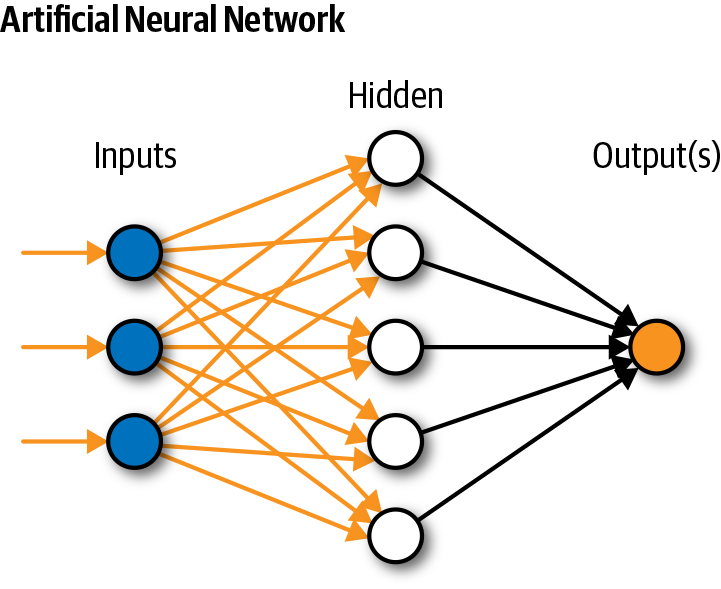
**C.5.2 Deep Neural Network Models**

We have used three deep neural network models for our dataset after the preprocessing it. The models are:

1. Long Short-Term Memory (LSTM)
2. Pooled Gated Recurrent Unit (GRU)
3. Bidirectional Gated Recurrent Unit (GRU)

**Deep Learning**

Deep Learning is a subset of Machine Learning. It is a type of Machine Learning technique motivated by our Human Brain. Deep learning is just a term which describes certain types of Neural Networks (NN). It tries to work according to how a human brain can work. Both can learn and become expert in the area. Just like on our lifetime we see different things and learn it; the same way Neural Network learns when feed data into it.



The NN process the data through layers of non-linear transformation of an Input data in order to calculate the output. A NN can consist of as many Inputs with as many numbers of Hidden Layers and then finally after processing we get the out depending upon the type of input.

It is used in Natural Language Processing, Visual Recognition, Fraud Detection, etc.

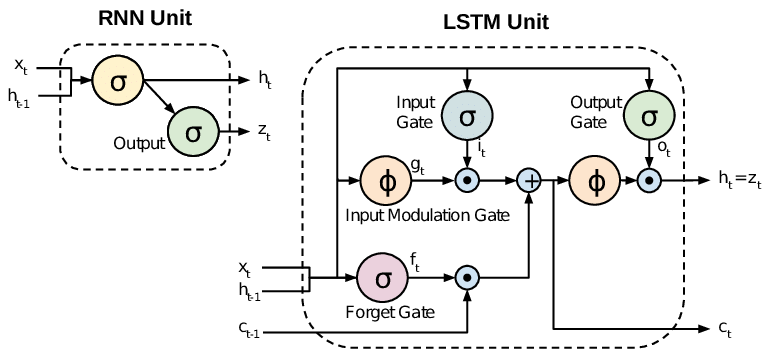
The problem with NN or even convolutional neural network (type of NN) it works for fixed grids or dimensions, so the input and output is fixed. Images of varying dimensions cannot be fed to CNN or NN and also there is no long-term dependency among the data.

**Recurrent Neural Network (RNN)**

* To overcome this limitation Recurrent neural networks (RNN) were based on David Rumelhart's work in 1986 as developed.
* RNN captures the sequential information present in the Input data i.e., dependency among the words in the text while making assumptions.
* But the drawback of RNN was it has vanishing gradient problem. We cannot train the network properly and this causes the RNN sequences not to retain in long term memory.
* To overcome the problems of RNN, Long Short Term Memory (LSTM) was proposed by Sepp Hochreiter and Jurgen Schmidhuber [[1]](#Ref1).
* For example:
* The sky is \_\_\_\_\_\_ (answer is blue)
* In the above sentence RNN model can easily predict that the answer can be “blue”. This is because the sequence of the sentence is not too long .
* But the in cases where the sentence sequence is too long like for eg:
* Fuji lived in Japan for 15 years. He loved the Japanese culture, and he is very keen of the people there. He is fluent in \_\_\_\_\_ (answer should be Japanese)
* This above sentence cannot be predicted by RNN as it’s a long sentence as it cannot remember relation between ling term sequences. In order to solve these kind of problems LSTM was introduced.

**Long Short-Term Memory (LSTM)**

LSTM consist of Input Gate, Forget Gate and Output Gate.

 Figure : RNN and LSTM [<https://rb.gy/ql43x9>]

In the above figure, we can see the comparison of LSTM with RNN. LSTM outdoes RNN in long sequence of text. As the MBTI dataset has posts and we wanted to do long sequence test classification, LSTM was the better choice than RNN. We have classified the post into 4 different personality types as discussed in the Data preprocessing section. (i.e., “is\_E”, “is\_T”, “is\_S”,”is\_J”) where LSTM can detect long sequences for each category of personality.

1. **glove.840B.300d** for word embedding was used
2. The parameters are fitted as:

-embed size = 50 (how big is each word vector)  
 -max features = 20000 (how many unique words to use)

-maxlen = 100

-loss=‘binary crossentropy’  
 -optimizer=‘adam’  
 -metrics=[‘accuracy’].

binary crossentropy’ is used as a loss function as we have classified it into0,1 for each personality type,‘binary crossentropy’ performs well which this type of classification.

After obtaining the cleaned data from the data processing done above up too to step where the “posts” are cleaned and the personality types are divided into 4 types, we have prepossessed the data with padding and tokenization using Keras in Tensorflow.

The code snippet is attached in Figure below:

Text

Description automatically generated**LSTM Preprocessing**

We have also added a dropout Layer in out model. Dropout layer is used to reduce the problem of overfitting. In this model, we set one embedding layer, then bidirectional LSTM, two pooling layers before dropout layer.

**Bidirectional Gated Recurrent Unit (GRU)**

Gated Recurrent Unit is the modification of Recurrent network and the hidden Layer.It is similar to LSTM and it is much better at capturing long range connection and helps a lot with vanishing gradient problems.

It was invented in 2014 by Kyunghyun Cho et al. It is closely related to LTSM with lower parameter and easy to compute.

It is utilizing the gating mechanism as same as LSTM to manage and control the flow of information between the cells in the NN.

GRU has two gates: 1) Update Gate 2) Reset Gate

Gates helps in determining what has to be retained/passed or dropped.

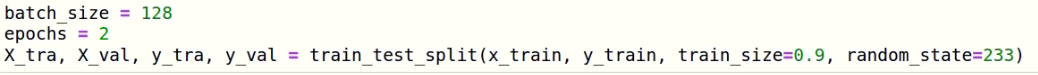
Update Gate: This gate helps in processing how much of the information needs to be maintained i.e., to be passed along to the future.

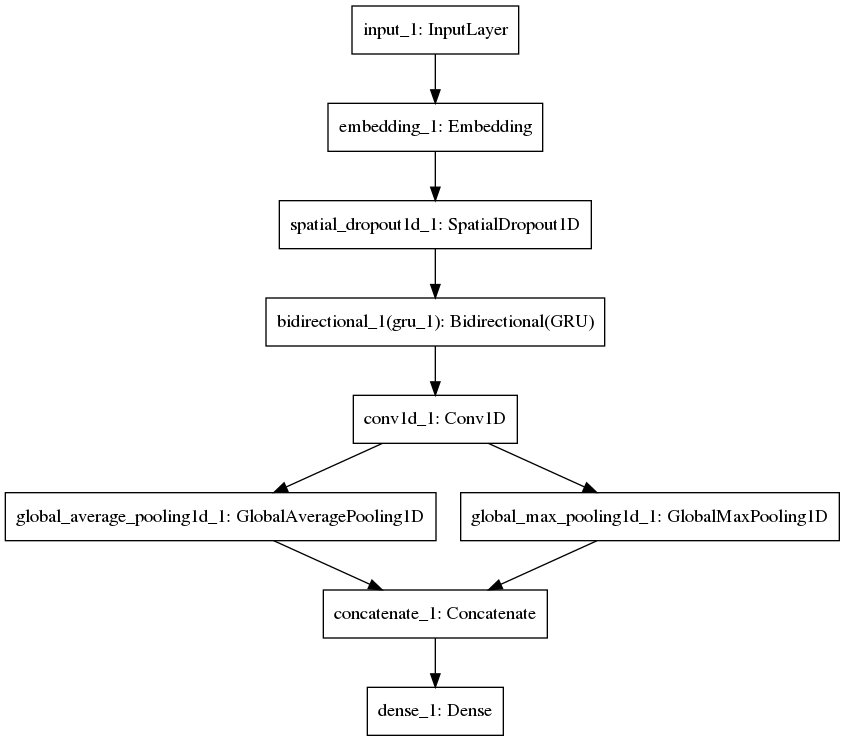
Reset Gate: This gate is used from the model o decide how much of the past information to forget.

A unidirectional GRU characteristically reads the input sequence from one direction. A bidirectional GRU consists of 2 vanilla unidirectional GRUs stacked side by side, but the second GRU reads the input sequence from the opposite direction

In this model, we have used glove.840B.300d for word embedding. Then we put max features=100000 (maximumfeatures) maxlen=150 (maximum length)  
embed size=300 (size of embedding)

Preprocessing as followings:  
Did tokenization and padding of each post and then convert it vector by using word embeddings. Bi-Directional GRU architecture is depicted in figure below. In this Bidirectional GRU model, we have set two pooling layers (average pooling and max pooling) layer after bidirectional GRU layer. Then we set dense layer. The batch size and number of epochs are 128 and 2 respectively. We have set binary cross entropy as loss function. We have done our experiment with 90% labeled data as training and 10% data as testing.





Bidirectional GRU model

**C.5.3 Pooled Gated Recurrent Unit (GRU)**

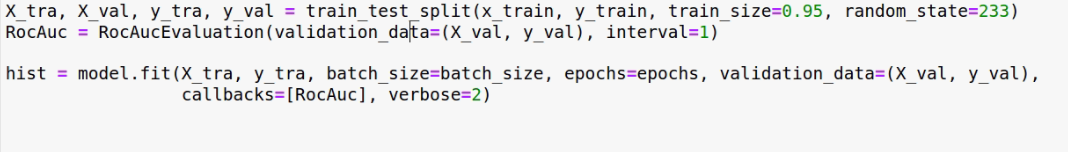
Pooled GRU is a simple GRU model with pooling layers.

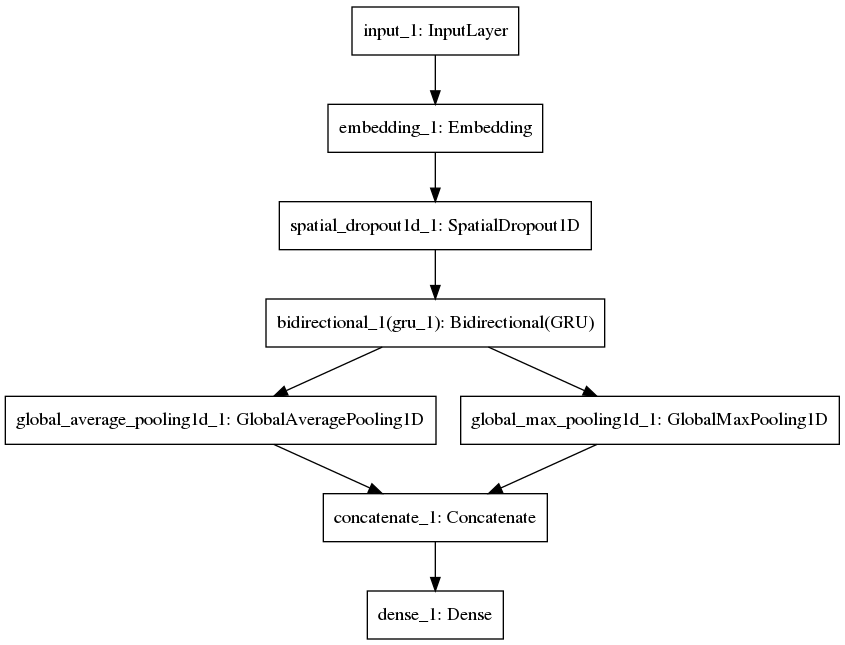
It is used to reduce the number of dimensions of the feature matrix. Thus it reduces the number of parameters to learn and the amount of computation performed in the network.

We also add dropout layer for reducing the overfitting problem.

For Preprocessing of Data

Did tokenization and padding of each post and then convert it vector by using word embeddings. Bi-Directional GRU architecture is depicted in figure below. In this Bidirectional GRU model, we have set two pooling layers (average pooling and max pooling) layer after bidirectional GRU layer. Then we set dense layer. The batch size and number of epochs are 32 and 2 respectively. We have set binary cross entropy as loss function. We have done our experiment with 90% labeled data as training and 10% data as testing.





Pooled GRU model

**D. Results and Comparison for Deep learning Models**

We have applied these deep learning models and compared the results among them. In Table 1, we can see the performance of our models.

We have split the train dataset into 90% as training size and 10% as validation size.

Overall, LSTM gives the best performance with AUC score 0.796 which is more than 80% accurate for the balanced class “is\_T”. Also, for unbalanced data i.e., “is\_S” class, the accuracy seems to be too low i.e., 0.474 for LSTM (which is highest among the three models). After balancing the class “is\_S” using random oversampling it is observed that Pooled GRU outperforms the other two models with the accuracy of 0.751 which is almost 75%.

Pooled GRU gives slightly better accuracy than Bidirectional GRU model.

LSTM also gives the best result for loss. It minimizes the loss to 0.5451 whether Pooled GRU gives maximum loss 0.573 for balanced class “is\_T”. For class “is\_S” after balancing we see that Pooled GRU minimizes the loss by 0.3578 and Bidirectional GRU gives maximum loss of 0.4399.

1. **For balanced data i.e., “is\_T” class**

|  |  |  |  |
| --- | --- | --- | --- |
|  | LSTM | Pooled GRU + Fast Text | Bidirectional GRU |
| ROC AUC | 0.796 | 0.773 | 0.779 |
| Loss | 0.5451 | 0.5731 | 0.5636 |

ROC AUC Score for class is\_T (balanced) Loss for class is\_T(balanced)

1. **For unbalanced data i.e., “is\_S” class**

|  |  |  |  |
| --- | --- | --- | --- |
|  | LSTM | Pooled GRU + Fast Text | Bidirectional GRU |
| ROC AUC | 0.474 | 0.473 | 0.457 |
| Loss | 0.3311 | 0.3312 | 0.3765 |

ROC AUC for class is\_S(imbalanced) Loss for class is\_S(imbalanced)

1. **After Random Over-Sampling for class “is\_S”**

|  |  |  |  |
| --- | --- | --- | --- |
|  | LSTM | Pooled GRU + Fast Text | Bidirectional GRU |
| ROC AUC | 0.634 | 0.751 | 0.661 |
| Loss | 0.4017 | 0.3578 | 0.4399 |

ROC AUC for class is\_S (balanced) Loss for class is\_S(balanced)

**D. Analysis Results and Comparison**

We have done validation on training dataset. The training dataset is labeled. We measure the results in Receiver Operating Characteristic (ROC) curve. The Area of Curve (AUC) is an effective measurement for binary classification. It plots the true positive rate against the false positive rate in various thresholds. So, we compare our models using the AUC values from each model’s output.

**Environment Setup:**

We have done our experiments using following environments:

python=3.7

TensorFlow 2.1  
Jupyter Notebook  
GPU: NVIDIA TITAN RTX

**D.4 Limitations and Future Work**

* The Natural language processing (NLP) classification to domain specific.
* The meaning of text may defer for different domain.
* Soo we are not sure if out model can perform well for different domain dataset.
* In Future we would like to evaluate our model on different dataset to see the accuracy and performance.
* We would also like to incorporate different other models to make it more reliable.
* Also use better data preprocessing techniques along with making the dataset balanced for all the classes to have much better accuracy.

**E. Conclusion**

We have implemented six different models to perform the classification of different personality types. We have found that deep neural network models outperform the conventional machine learning models. Moreover, Pooled GRU with FastText embedding gives the best result among 6 models. Before passing the comments to each, we work on different prepossessing to find a good result. We have compared the models by using AUC score.

We propose to use Pooled GRU model for toxic content classification.

**Repository**

Link [[2]](#RepoLink) to get our code and output.

**Member Contribution**

* Nalisha Rathod (GS9440) worked on literature review, Data preprocessing for Deep learning models and building deep neural network models
* Sujata Gorai(hc6837) worked on preprocessing the dataset and building conventional machine learning models
* Both of us worked on analysis part

**References**

[1]   Sepp Hochreiter and Ju ̈rgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[2] <https://github.com/nalisharathod01/Myers-Briggs-Personality-Prediction>

[3]  [https://www.sciencedirect.com/science/article/abs/pii/S0306457321000406#](https://www.sciencedirect.com/science/article/abs/pii/S0306457321000406)!

[4] Agarwal B (2014). Int J Comput Syst 1:1–4

[5] Machine Learning Approach to Personality Type Prediction Based on the Myers–Briggs

Type Indicator

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre- training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[7] Anaıs Ollagnier and Hywel Williams. Classification and event identification using word embedding. neural networks, 6:7, 2019.

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.