Tweets Classification

Runyao Yu
Technical University of Munich
Chair for Human-Centered Assistive Robotics
Munich, Germany
runyao.yu@tum.de

***Abstract*—This research aims to classify the tweets from eight users. In this research, I applied three RNN models i.e. Vanilla Recurrent Neural Network (Vanilla RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) on the tweets dataset. Next, I added dropout on them. Then, I replaced Adam as AdamW. After comparing their performance combined with attention and bidirectional structure, I attempted different combinations of tricks and observed on the validation accuracy.**

***Index Terms*—NLP, Text Classification, RNN, LSTM, GRU, Dropout, AdamW, Attention, Bidirectional**

I. Introduction

Artificial Intelligence (AI) is constantly expanding and has been one of the most prominent research topics in recent decades. The goal of AI is to empower systems with intelligence capable of human-like learning and reasoning. It has numerous advantages and has been successfully implemented in a variety of industrial fields, such as Natural Language Processing (NLP).

With the help of NLP, we are able to analyse the huge number of texts and find their hidden relations. As the development of online social media, it becomes extremely convenient for people to post their opinions and minds on the internet. In this research, in order to identify the user ID based on their posts, I applied three text classifiers on them, i.e. RNN, LSTM, and GRU.

RNN is a class of artificial neural networks where connections between nodes form a directed or undirected graph along a temporal sequence. As a variant of RNN, LSTM can solve the problem of gradient vanishing to a large extent due to its forget gate and is widely used in time-series problems. GRU is a gating mechanism and it is quite similar to LSTM while having fewer parameters.

After I applied three models on the dataset, I then added dropout and changed the optimizer as AdamW to observe their performance. Adding dropout can usually increase the generalization ability, and trying different optimizer might lead the model to a better minimum. Moreover, I attempted to add attention mechanism and bidirectional structure on the selected best baseline model, respectively. The attention mechanism allows the model to concentrate on the important part of texts, and the bidirectional structure can make it easier for model to understand the context based on both directions.

II. No Free Lunch Theorem

When the performance of all optimization techniques is averaged across all feasible problems, the theorem states that they all perform equally well. It suggests that there is no one ideal machine learning algorithm for predictive modeling issues like classification and regression.

This is the reason why we need to always try different models with different hyperparameters on a new dataset, even though we all know that in general meaning, Transformer is better than Linear Regression.

III. Dataset

The dataset was preprocessed by Mrs. Hyemin Ahn. The dataset consists of eight classes from eight users, each text has been tokenized and assigned a numeric value, which can be retrieved in the meta.json as a dictionary. The goal is to train a NLP classifier to predict that the input text belongs to which user.

IV. Baseline Models

In this section, I applied three different models on the tweets dataset and compared their performance. For each model, I trained them with 100 epochs without any other tricks.

A. RNN

Recurrent neural networks (RNNs) are so named because they complete the same task for each element of a sequence, with the result being dependent on the prior computations. Due to this property, RNNs can process the time-series signals such as texts in the context of NLP.

B. LSTM

The problem with vanilla RNNs is computational in nature: when training a vanilla RNN using back-propagation, the long-term gradients which are back-propagated can vanish or explode. RNNs using LSTM units partially solve the vanishing gradient problem, because LSTM units allow gradients to flow unchanged. Therefore, LSTM has generally better performance than vanilla RNNs.

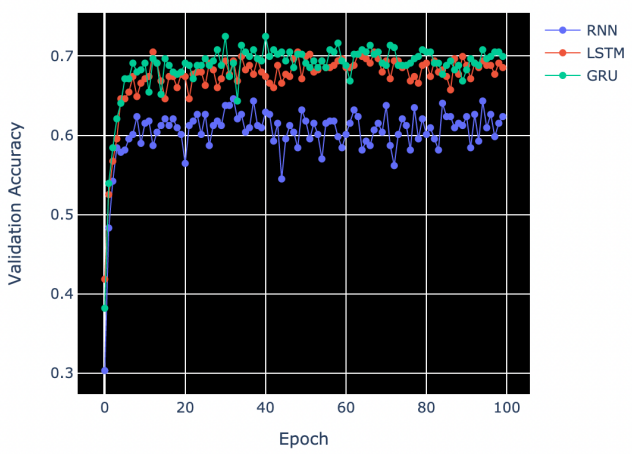

Fig. 1. Baseline Models

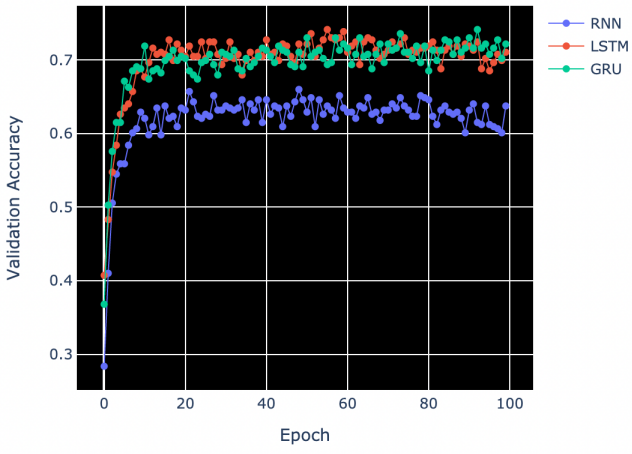

Fig. 2. Baseline Models with Dropout

C. GRU

The GRU is similar to a long short-term memory (LSTM), but it lacks an output gate. Thus, it has fewer parameters. GRU's performance was found to be comparable to that of LSTM in some cases. On some smaller and less frequent datasets, GRUs have been found to perform better.

As shown in the Fig. 1, after training 100 epochs, the validation accuracy of RNN, LSTM, and GRU reached to its highest value i.e. 64.60%, 70.51%, and 72.47%, respectively.

V. Tricks

In this section, I first added dropout on the three models. Then, I replaced the optimizer from Adam to AdamW. After I concluded the best baseline model, I combined it with attention mechanism. In parallel, I added the bidirectional structure on the best baseline model, and compared their performance.

A. Dropout

Dropout is easily implemented by randomly selecting nodes to be dropped-out with a given probability. With the help of dropout, we can reduce the complexity of DNN models and thus avoid overfitting problem.

In the following experiment, I added one dropout layer after the embedding layer and set the dropout rate as P=0.2.

```
# in the _init_ function
self.dropout = nn.Dropout(p=0.2)

# in the forward function
embed = self.embedding(token_id)
embed = self.dropout(embed)
out, _ = self.rnn(embed)
```

As shown in the Fig. 2, after training 100 epochs and after adding one dropout layer after the embedding layer, the validation accuracy of RNN, LSTM, and GRU reached to its highest value i.e. 66.01%, 74.16%, and 74.16%, respectively. It turned out that using dropout can indeed increase the generalization ability of the model and therefore increase the accuracy. Also, it seemed like LSTM and GRU do not have much difference. Next move, I will try to change the optimizer and observe their performance.

B. Optimizer

Based on the first experiment, I replaced the optimizer from Adam to SGD and AdamW, respectively. Training with SGD led to a very slow convergence. As I trained the model with 150 epochs, the validation accuracy increased slowly to around 47%. Even though many papers showed that training with SGD will eventually converge to the global minimun, it is infeasible for me to train three models with hundreds or thousands epochs. Therefore, I gave upon the experiment with SGD and only continued to train the model with AdamW

AdamW is a stochastic optimization method that modifies the typical implementation of weight decay in Adam to combat Adam's known convergence problems by decoupling the weight decay from the gradient updates.

In the following experiment, I added one dropout layer after the embedding layer and set the dropout rate as P=0.2, meanwhile, I changed the optimizer from Adam to AdamW.

```
# in the block for model definition
optimizer = optim.AdamW(...)
```

As shown in the Fig. 3, after training 100 epochs and after adding one dropout layer with the replacement of optimizer, the validation accuracy of RNN, LSTM, and GRU reached to its highest value i.e. 66.29%, 75.00%, and 76.12%, respectively. Three models were improved due to the replacement of the optimizer. Especially, for GRU, the validation accuracy became 2% higher, which is significant.

After observing these three experiments (baseline models, baseline models with dropout, baseline model with dropout and AdamW), I found that RNN is generally worse than LSTM and GRU. Also, GRU outperformed the LSTM (this might be a coincidence, but since their performance are approximately

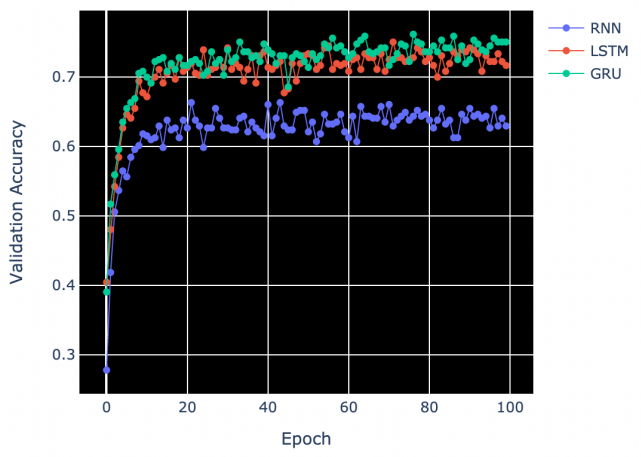

Fig. 3. Baseline Models with Dropout and AdamW

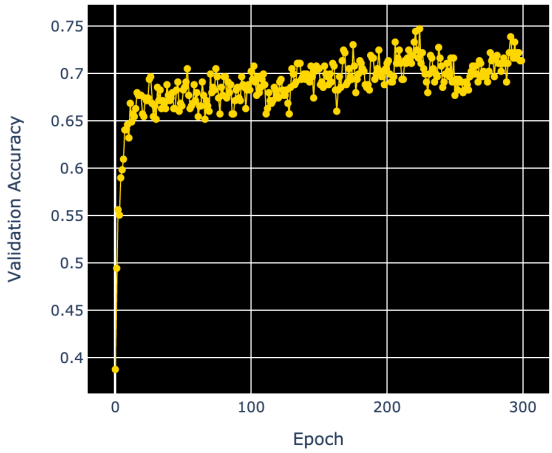

Fig. 4. GRU with Dropout, AdamW, and Attention

the same in the general case, and GRU possesses less parameters than LSTM), considering the GPU power and the time limitation, in the following experiments, I decided only to use GRU for training.

C. Attention Mechanism

A neural network is considered to be an effort to mimic human brain actions in a simplified manner. Attention Mechanism is also an attempt to implement the same action of selectively concentrating on a few relevant things, while ignoring others in deep neural networks. Adding attention mechanism into the RNNs models would generally improve the performance.

Here, I used multi-head attention with head=1 (the number of head requires to be divided by the batch size as an integer, and I used batch size=1). The required inputs of multi-head attention should be Q, K, and V, where they are denoted as Queries, Keys, and Values, respectively. For the encoder, the Q, K, and V are the same. I set a dropout rate as P=0.1 inside of the multi-head attention.

```
# in the _init_ function
self.attn = nn.MultiheadAttention(rnn_dim, 1, dropout=0.1)

# in the forward function
embed = self.embedding(token_id)
out, _ = self.rnn(embed)
out, _ = self.attn(out, out, out)
```

As shown in the Fig. 4, based on all previous tricks, after training 300 epochs and after adding attention mechanism, the maximum validation accuracy of GRU went down from 76.12% to 74.72%, which disobeyed my expectation. One possible reason for causing this result might be because that the model was already complicated enough to conduct this classification task. Therefore, keeping adding more fancy functions to the model will not help much.

However, a hard-working student will never kneel down to a setback! I decided to try other complicated models and to see if this is consistent with my assumption.

D. Bidirectional Structure

Bidirectional recurrent neural networks (BRNN) connect two hidden layers of opposite directions to the same output. With the help of this connection, the output layer can get information from the past and future states, simultaneously.

Here, to control variable, I changed the unidirectional GRU to bidirectional GRU based on all previous tricks except adding the attention. The importance is that noting after the bidirectional GRU layer, the length of the output will be doubled. Thus, I need to also double the input dimension of the last linear layer in order to adapt the new structure.

```
# in the _init_ function
self.rnn = nn.GRU(..., bidirectional=True)
self.out_linear = nn.Linear(2*rnn_dim, ...)
```

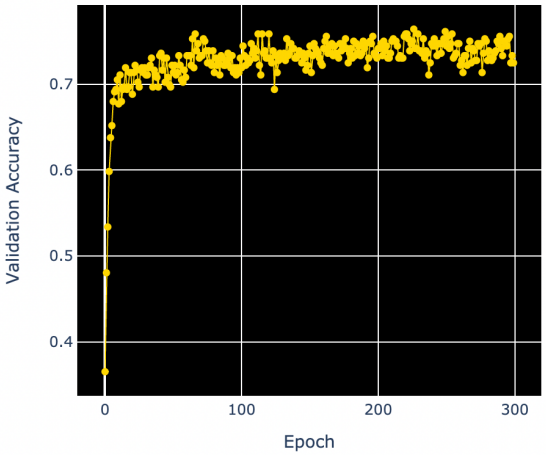

Fig. 5. GRU with Dropout, AdamW, and Bidirectional Structure

As shown in the Fig. 5, based on all previous tricks, after training 300 epochs and after adding bidirectional structure, the maximum validation accuracy went up slightly from 76.12% to 76.40%.

VI. Complicated Combination

In this section, I combined different tricks to form more advanced models and ran multiple experiments.

A. GRU + Multi-Dropout + AdamW

Adding multi-dropout will not change the number of parameters. I didn't change the dimension rnn_dim=1024. At this point, the total number of parameters will be 11577864 and won't exceed the limitation.

In this combination, I simply inserted one more dropout layer between the GRU layer and the last linear layer.

B. GRU + Multi-Dropout + AdamW + Bidirectional Structure

To increase the ability of the model, I increased the dimension as rnn_dim=1244. At this point, the total number of parameters will be 19986552 and won't exceed the limitation.

In this combination, I inserted one more dropout layer between the GRU layer and the last linear layer. It should note that the input dimension of the last linear layer should be doubled.

C. GRU + Multi-Dropout + AdamW + Attention + Bidirectional Structure

It is impossible to simultaneously add attention and bidirectional structure while maintaining the original dimensions due to the limitation of the number of parameters. I changed the embed_dim=256, and rnn_dim=833. At this point, the total number of parameters will be 19997506 and won't exceed the limitation.

In this combination, it is crucial to know that after changing the structure from unidirectional GRU to bidirectional GRU, the input dimension of attention layer and the last linear layer needs to be accordingly doubled.

D. Multi-Layers GRU + Multi-Dropout + AdamW + Multi-Attention

It is impossible to simultaneously add multiple layers and attention while maintaining the original dimensions due to the limitation of the number of parameters. I changed the rnn_dim=947. At this point, the total number of parameters will be 19980773 and won't exceed the limitation.

In this combination, it is important to understand that after going through the first GRU layer, the input dimension of the second GRU layer needs to match the output dimension of the previous one. Another thing needs to be pointed out is that after multi-layers GRU and its attention layer, we need to concatenate the output of each attention layer and then input it to the last linear layer.

As shown in the Fig. 6, in this four types, the most simple one (type A) has the best performance (type A: 79.49%). After stacking bidirectional structure on type A, the performance became worse (type B: 77.25%). If keeping adding attention layer on type B, the performance became even worse (type C: 74.72%). Additionally, adding more GRU layers and attention layers based on type A will also not help (Type D: 75.28%). Thus, my best model is type A.

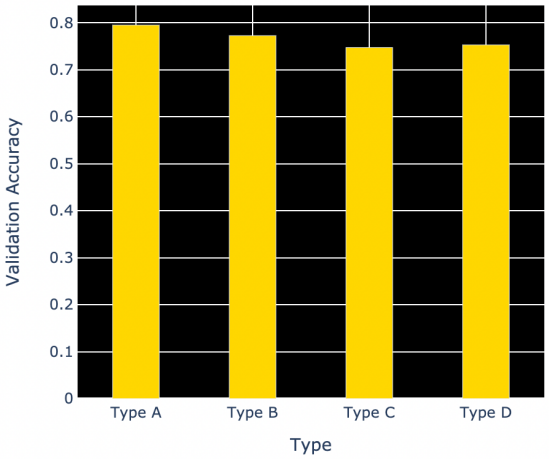

Fig. 6. Complicated Combination

VII. Discussion

As we saw in the previous experiments, the replacement of certain optimizer can significantly influence the test accuracy. This is due to that the optimizer helps the model to converge to a better minimum. Trying other optimizer such as RMSProp or AdaGrad might also lead to an improvement of the model.

In NLP problem, applying the data augmentation such as EDA or Back-Translation would usually improve the accuracy. In our case, this classification problem aims to classify the texts based on (somewhat) user's writing style or often used words etc. Applying Back-Translation might break the original style and/or change the custom usage of words. However, this still worth a try.

VIII. Conclusion

After I conducted all of these experiments, from the most simple RNNs, to more complicated and advanced models with dropout, AdamW, attention, and bidirectional structure, we can conclude that for some case, a simpler model might give us a higher accuracy and better generalization ability. Blindly stacking various tricks and increase the number of parameters of the model will probably not help.

The choice of optimizer can indeed influence the convergence of the model. After I changed the optimizer from Adam to AdamW, the validation accuracy of all three models got improved.

Another significant finding is that dropout played a crucial role in our case. As I kept adding more dropout, the performance became better. However, keeping adding hundreds dropout will also not help. This is essentially the bias-variance trade-off. We need to always avoid underfitting and overfitting problems in the context of machine learning.