Optimizing Deep Learning Solution Performance by Custom Blending for Typical Natural Language Processing Business Problems.

Indranil Dutta

Principal Consultant and Lead Data Scientist

---***---

Abstract : *In this research will show case a very simple but effective mechanism to achieve high accurate results from standard Deep learning algorithms for typical NLP business problems. The analysis depicts how to optimize the performance of the models by stacking or blending one with other and the logic behind selection of each algorithms. Here we will simply create a neural network type structure where instead of each neural node we will be using either single Deep learning models or a stack of multiple deep learning models and pass through the results by other layers of advanced machine learning models called the Meta model layers and again throw the output through a soft voting classifier as the final layer. The key contrast of this mechanism is all the model(nodes) contributes the same weightages and there is no back propagation to update the weights through any gradient. Finally, will discuss how these architectures will optimize the overall performance with some standard metrices.*

Key Words: NLP, Deep Learning, Stacking, Meta Model, Classification

1.INTRODUCTION

In modern era of Artificial Intelligence Data scientists and engineers use many a state of art technologies like BERT, GPT, T5, XLNet, ALBERT, ROBERTa and many more to tackle the typical NLP problems like classification, relation embedding, sentiment analysis, question answering, summarization, language translation and so many. These methodologies are computation heavy and requires advanced configuration to perform the task. Also, a very small dataset sometimes shows high variances by using these SOTA models.

But the following analysis shows a very simple but tricky methodology to use the standard deep learning methodologies along with advanced machine learning algorithms to optimize the overall purpose of a business problem without using any transformer based or attention model. To avail these computations even no GPU is required. The methodology we called stacking or blending in the Machine Learning landscape, or if anyone is well versed with R, could be familiar with Super learning. A simple NLP binary classification problem to classify the email body from the signature block for a bunch of 1000 emails can be easily performed through these mechanisms and finally I am going to show you how the model performance improved with respect to the individual models. The architecture is very flexible and data scientist or ML engineers can play around these techniques by their own discretion and choice. Also, the same architecture can be fitted in GPU or TPU environment as well once the training dataset is heavy or Data scientist opt for a little complex architecture to achieve the business goal.

2. BUSINESS PROBLEM

Let's consider a very simple business problem to discuss the solution here. We have to create a binary classification model to segregate the email body (text) from the signature block. It means whenever we feed an email text file (single or multiple) in the model, it can successfully generate the tokens of each sentences/blocks and predict correctly on the basis of contents available in the tokens and thereby identify the signature block, which basically comprises of Name, email id, organization Name, Contact Number, Social Media details along with some HTML tags or logos.

3. DATA PREPERATION

Eventually, we have to maintain a series of data preprocessing and augmentation techniques to clean up the data and in order to better training inject some perturbation and random effects to the data to make it more generic and stable. As this is not the focus point of this research, will not discuss in detail that what are the different preprocessing and augmentation techniques have been used to get the data ready for the model consumption. Finally, we have labelled the data and cleaned it and augmented by n times to have a standard training set. We have kept a 10% of the data aside for out of bad validation and rests 90% for the key model building and out of that as well we make a 70:30 split to create a training and testing set.

4. SOLUTION DESIGN

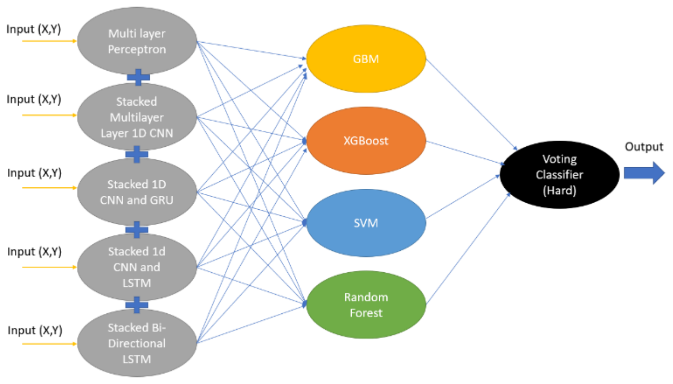
In simple Machine Learning Predictive use cases we have used the concepts like Stacking and Blending where multiple different classification/regressors stacked each other. The difference between stacking and blending is that Stacking uses out-of-fold predictions for the train set of the next layer (i.e meta-model), and Blending uses a validation set (let's say, 10-15% of the training set) to train the next layer. Most of the cases it's been observed that it achieves higher precision and accuracy that the individual model but sometimes it's suffered a high variance issue as well. Hence the most effective way of using this technique lying behind the selection of right algorithms to stack each other to optimize the performance.

These techniques get encouraged from the traditional stacking and blending architecture but follows a totally different architecture. Here we feed the training data to each one of the blocks. The first layer is a Modern Multi-layer perceptron which is nothing but a jungle of dense layers with a relu activation. But data scientists use few drop out layer as well to manage the overfitting issues for each one of the blocks. The second block is a one-dimensional convolution Neural Network which is also very effective for language classification. It's a totally different structure which uses a Kernel based scanning of the text block and convert a vector against each frame and then apply the max or average pooling to summarize the data and apply flatten layer to organize the data in a serialized manner and finally apply the dense layer with a sigmoid function to classify the class. Hence this has totally a different flavour than a traditional MLP structure.

We cannot judge the skill of the model from a single or double evaluation. The reason for this is that Neural Networks are stochastic, meaning that a different specific model will result when training the same model configuration on the same data. This is a feature of the network in that it gives the model its adaptive ability but requires a slightly more complicated evaluation of the model.

Hence, we introduce the hybrid structure models in subsequent nodes to give more complex evaluation on the given data. The gated recurrent Unit model follows a different structure altogether with a combination of a reset gate and update gate with sigmoid and tanh activation functions. Hence once we pass the data through a one-dimensional convolution layer and then through another GRU layer, the block will learn a new evaluation of the input data.

Figure -1: A custom blending set -1

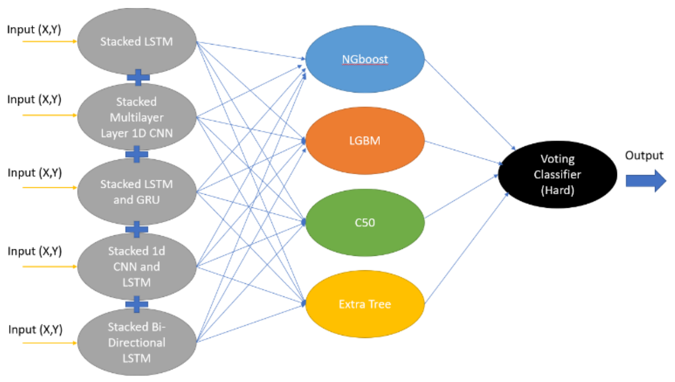


On the similar side we will create another block of a one-Dimensional CNN and a stacking of LSTM layer. As the LSTM also have a different architecture altogether, it allows the data to adapt through a different learning. And finally, keep another block of a bidirectional LSTM which can learn the sequence and dependency of words from left to right and right to left (both direction). This is obviously adding another value to the overall learning.

The tricks of the stacking is not to club two similar architecture in a same node to avoid the benefit of diversified leaning.

Now in the second layer the Meta model will absorb the output of these 5 individual blocks and train itself. It's always a good practice is to use different flavors of Meta Models to give a very generalized view of the overall learning. May be the bagging model doesn't fit the data properly, but the boosting does. Otherwise, both bagging and boosting ensembles do not fit the data properly but a simple KNN or naïve bayes does. Hence if we use a group of 4 Meta Models in the second layer try to distribute between different flavors and architectures. Like we did in first diagram a sequential ensemble, a parallel ensemble and a kernel-based model to test the learning. Finally use a voting classifier with a soft voting where we get the option to decide the weightage of individual Meta models and based on the higher vote of a particular class the final class has been defined.

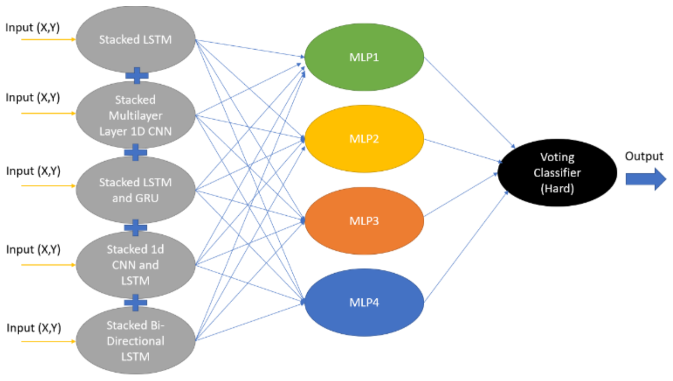
Figure -2: A custom blending set -2



Now refer the diagram no 1, we can see an overlapping of two boosting Algorithm of similar architecture on the second layer which is actually limits the learning efficiency of two nodes in a single learning. Similarly, if we observe the diagram no -2, the third node of first layer, the combination of LSTM and GRU which both are the extension of RNN architectures actually limits the learning potentiality of that particular block. This item has deliberately included in the diagram to show case where are the common mistakes happens and what are the best tricks to create a perfect architecture to gain the optimized results. Now comparing to the Diagram -1, the second architecture will suffer a little less diversified learning as it limits the learning in both of the layers. It has included the NGboost and LGBM both in the second layer which eventually acts as a single node.

There could be another angle of limiting the learning potentiality of the model. If we refer the following diagram, we can see there are major drawbacks in the second layer. Though the configuring of each node of MLP is different but using the similar learning across all Meta Models actually limits the learning potentiality. Already the input data is passed through the neural network blocks from standard deep learning algorithms and already used the relu function and specific optimizers like Adam. If the same inputs again feeds through the same layer with relu activation there is nothing different to transform and nothing much to learn from the second layer. Hence all the layers are just replicating almost the same results that comes from the previous layers and there is no major purpose of using the voting classifier to expect a majority vote.

Figure -3: A custom blending set -3



Hence using this type of architecture actually drives a less accurate results inspite of using a complex structure.

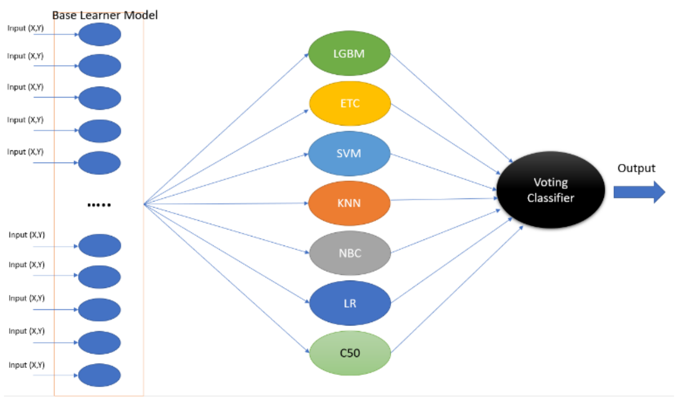
But practically once the real business scenario comes into the picture data scientist mostly follows the underlying architecture to get the best possible results. As of now whatever architecture we have discussed is just to demonstrate the core idea of the process. But from practical application point of view the underlying diagram is the most optimized. The rule of thumb says if there are 7 nodes in the second layer then we have to atleast keep 4X nodes i.e. (4*7) = 28 nodes in the first layers. Now we have to use the common nodes multiple time to occupy the 28 nodes. Here the trick is to use the following:

- If we use same CNN in multiple case use different Kernel with different strides, in some case use the maxpooling and some cases average pooling to keep a generic differentiation between the architectures.
- If we use the LSTM and GRU, must use the drop out layer with different percentages and use different batch size. The training time of LSTM and GRU is more so better to use a comparatively higher batch size.
- Use different learning rates and different optimizers like Adam, Nadam, RMSprop etc.
- In case of MLP we can use different combination of Inputs for each dense layer with different drop out ratio.
- Use different early call back criteria
- Use the metrices as accuracy in some cases and F1 score in some other cases.

This will give a diversified flavor in the Meta Model layers and model learns different scenarios from each input. This is a very powerful methodology. Now in the second layer here we utilize the most optimum combination of Meta learners, each one belongs to different class to enable the best learning capability. One from bagging, boosting, kernel-based learning, Bayesian Learning, simple logit classification, decision tree-based classifier, Distance based classifier. All of them learns based on their individual learning capacity. Now once we throw the outputs to a voting classifier after assigning equal weightage to all of the meta learners. We can surely see a variety of prediction comes and based on the majority class we can tag the class for the case. This is indeed a powerful technique.

We cannot use the cross validation of the overall structure together or making the hyper parameter tuning together. We need to perform the cross validation and hyper parameter tuning for all of the individual models one by one. But once we are using different models with different parameters, in that scenario hyper parameter tuning is not that required as the number of nodes in first layer is large enough. In case we are using only 4 -5 nodes in the first layer and each model is used once, we need to use the hyper parameter tuning using Keras tuner to get the best possible value.

Figure -4: A custom blending set -4



Obviously based upon the complexity of the architecture the processing time would also spikes. In this shape i.e., 28 nodes in first layer along with 7 nodes in second layer and we have 1000 email (after augmentation the size is 12000 emails). Out of 1200 are kept for out of bad evaluation and from rests 10800 emails 70% i.e., 7560 email content have been used for training. After tokenizing there are 37342 lines of records are there and we have created a vector embedding of these words where we considered the vocabulary size as 5000. So, the entire dataset is a matrix of 37342*5000 which is a challenging task for a local machine to handle with that type of architecture.

In that case we need the distributed computing or the power of GPU processing to cater the requirement. But using a lesser number of first layer nodes with a small vocab size this can be easily doable in a local system with 4 cores and 16GB RAM.

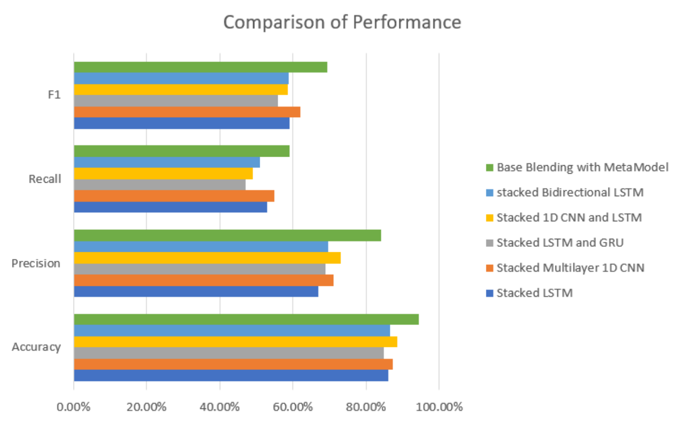
5. RESULT COMPARISON

Let's see the result comparison of the models and how to improvise the efficiency using that architecture. The following table shows the clear comparison of all the individual models as well the Blended one and we can see the Blended one outperformed the other models by significant extent. The efficiency can be increased by increasing its nodes. Here the Hybrid model results showed for the model of diagram no -1. This has been performed in the local system only. Now you imagine how the performance could be if you check the model with diagram n0 -4. There could be a potentiality of overfitting but using the drop out layers carefully with little higher percentages across all models would easily remove the issues.

Table -1: Result comparison

Model Type	Model Name	Accuracy	Precision	Recall	F1
Individual Base Model	Stacked LSTM	86.13%	0.67	0.53	0.592
	Stacked Multilayer 1D CNN	87.20%	0.71	0.55	0.620
	Stacked LSTM and GRU	84.90%	0.69	0.47	0.559
	Stacked 1D CNN and LSTM	88.50%	0.73	0.49	0.586
	stacked Bidirectional LSTM	86.45%	0.697	0.51	0.589
Hybrid Model	**Base Blending with MetaModel**	**94.30%**	**0.84**	**0.59**	**0.693**

Figure -5: A visual comparison of the individual model and the blended one with metamodels and voting classifier



6. BENEFITS

- Higher accuracy and precision within limited resources.
- No need for any GPU/TPU if we use simple 5 or 6 nodes first layer. But gives better results than any standalone model.
- No need to train any Transformer or attention model for getting better results.
- Simple coding exercise for data scientist or engineers rather than using any high ended SOTA models for simple NLP use cases.
- Less memory consuming and less parameters. Unline SOTA models it doesn't show millions parameter occupancy.

7. LIMITATIONS

This architecture is used to overcome the performance of individual models at good extent within limited resources, but it has some limitation as well.

- Unlike Transformer architecture it has never focuses on any positional encoder
- There is no option for specific attention for any specific characters.
- With the higher number of nodes in first layer the computation effort would be heavy.
- In very small dataset high chances of over fitting.
- If we use higher no of LSTM, Bidirectional LSTM or GRU modules, it directly impacts the overall training time of the entire blended architectures.

8. CONCLUSION

Finally, the entire module can be used as an importable package for readymade and quick usage anywhere in any environments. Once developed as a module it can serve as a low code no code solution for any of your NLP classification problems. There is a high affinity among Data science aspirants to directly use the SOTA models from sources like hugging face or John Hopkins SparkNLP. It unnecessary consumes the resources and memory to tackle simple problems. Whereas this are a very wide used techniques in Machine learning landscape and shows promising results over decades. The research is to encourage data science aspirants to explore the option to get a good accurate model with the use of the above-mentioned architectures and tricks.

REFERENCES

1. Ensemble learning with Stacking and Blending | What is Ensemble Learning (mygreatlearning.com)

2. Illustrated Guide to LSTM's and GRU's: A step by step explanation | by Michael Phi | Towards Data Science

3. 1D Convolutional Neural Network Models for Human Activity Recognition (machinelearningmastery.com)

BIOGRAPHIES

Indranil Dutta is a Principal consultant and Lead Data scientist with more than 11 years of rich experience in Data science and Artificial Intelligence in various industries. His core competency is in delivering scalable Data science and AI projects in Big data and Cloud environments.