How to use Reinforcement learning to valuate textual data

Ankit Kulshreshtha
AI Labs
Course5 Intelligence
Bengaluru, India
ankit.kulshreshtha@course5i.com

Sudhanshu Upadhyay
AI Labs
Course5 Intelligence
Bengaluru, India
sudhanshu.upadhaya@course5i.com

November 10, 2020

Abstract

Having an accurate dataset is a very important aspect for training neural networks. A model trained on corrupted and faulty data may result in poor performance and getting a clean dataset is not always possible. While manually cleaning the data may help if the size of the dataset is small, it isn't always possible nor an effective solution for training neural networks, which essentially require large amounts of data for training. In this paper, we present the application of reinforcement learning for valuing each data sample, based on its importance for training for a text classification task. We encode the textual data using Transformer-based model and feed it to the reinforcement learning agent to output a selection probability which ranks each data point on the basis of its importance. Data valuation using reinforcement learning increases the accuracy of our trained models by up to 7%.

***K*eywords** Reinforcement learning · Text Classification · Transformers

1 Introduction

Deep learning models have achieved a lot in the NLP domain. Neural networks can now classify text, convert text from one language to another, extract information from texts, etc. Many real life datasets that we come across include a significant number of samples which have either incorrect labels or noisy data which is not fit for training a deep learning model, as a deep learning model can only be as good as the data (training data) itself. If the data itself is faulty or has errors, then the model also learns from this faulty data. Hence, it's important to train the deep learning model, using data that is accurate. Owing to the large size of the dataset for such tasks and time constraints, it isn't always possible to manually go through each data sample and rectify or correct them. In this paper, we apply data valuation technique on our text dataset to remove faulty samples.

To accomplish this task, we use reinforcement learning based framework. The field of reinforcement learning (RL) comprises a set of techniques and algorithms for machines and software to learn from their environment using feedback, just like humans learn using positive and negative feedback. Reinforcement learning algorithms are being applied to a wide range of tasks, ranging from self-driving cars and industrial automation to gaming AI such as AlphaGo [1].

Reinforcement learning, though, can be applied to a variety of tasks, it comes with its own set of challenges. Training an RL agent is pretty easy for simple tasks, however, as the complexity of the problem increases, the training time and computational power required also increases significantly. In this paper, we discover the application of reinforcement learning for data valuation. We will be using reinforcement learning for valuating a dataset for a text classification task. Training an RL agent for this task, comes with its own set of difficulties. The environment for the agent is not defined, and representing the current state, defining what a single episode would comprise of, balancing exploration and exploitation policies, are some of the important parts worth considering, else it may result in a sub-optimal model

which is not able to clearly identify low and high value samples.

Specifically talking about the data valuation problems, the time and the computational power required is quite high. Data valuation tasks involve training the main model multiple times or using heavy complicated operations which are time consuming for calculating the importance of each training sample. Besides this, even training the RL agent takes a lot of time. RL agents for such tasks have to be constructed using neural networks, as simple RL agents fail to accomplish the task due to its sheer complexity. For data related to linguistic tasks, where the later part of a input sequence is dependent on the previous part, a complex state representation needs to be selected which contains information about this correlation, so that the agent used for data valuation can perceive the meaning of entire sequence.

Besides the problems faced in using RL, the results are quite promising if one can overcome these issues. To accomplish our goal, we feed every training sample to the RL agent which assigns a numeric value to each of this sample, where this numeric value represents the fitness of this sample for training our deep learning model. After the RL agent has assigned the fitness value to every sample, we can simply discard the samples having very low fitness. We use reinforcement learning to assign a value to each sample which describes the importance of that datum in training the model.

2 Related Work

Various data valuation techniques have been used in the past for selecting useful data samples to be used for training a neural network. The leave-one-out (LOO) method, measures the training value of each sample. Where the training value of a sample is calculated by subtracting the performance measure when the model is trained on all samples, except the one in consideration, from the performance measure of the model which is trained on the entire data. However, this requires training the model for 'N' times, where 'N' is the number of samples in the training set. This paper [2] proposes to estimate the model performance using influence functions. Obtaining the influence function requires evaluating the inverse of the Hessian for the loss function. However, the time complexity still depends on the number of training samples and the model parameters and becomes high for large neural networks. A method to approximate the influence function with the time complexity being proportional to the product of number of training samples and the model parameters was also introduced, which takes significantly less amount of time compared to the previous methods, but is still high for large networks.

Another widely used method is the Shapley value-based methods, which is derived from game theory. In this method, we calculate the Shapley value for each training point which in game theory concept, is a method to distribute the total gains generated by the coalition of all players. We calculate the Shapley values for each training sample by finding the average marginal contribution of this sample to all possible subsets of the entire training data formed by other training points. However, this method is also practically expensive, because the time complexity increases exponentially as the number of training samples increases. Besides, it also requires to re-training of the model. Two approaches to calculate the approximate Shapley values were introduced in [3]. Both of these methods used Monte Carlo approximation to calculate the expected contribution of a training point to a random subset. This Monte Carlo simulation approach, however, still requires to re-train the model for each subset and hence these are not efficient methods which can be used as practical approaches.

Another method introduced, uses K-nearest neighbor (KNN) algorithm to get efficient data valuation for large training sets and large deep learning models [4]. The complexity of this method is independent of the model size. The KNN classifier finds the locality structure in the training data by calculating the similarity between the data points, which in turn leads to efficient calculation of data values. This method can be used with both LOO and Shapley value calculation.

3 Methodology

We employ a meta-learning framework called DVRL [5] which tries to jointly learn the data values with the target task predictor model. In order to process text sequences, we use an approach which consists of two modules.

The first module is a text classifier which outputs the probability of the text belonging to each class. This classifier takes in input as encoded representation which has been generated from passing the text sequences through a transformer encoder. This encoded representation is then pooled to obtain a flattened representation of hidden-dimension size (usually, 768). This pooled output is then fed into a fully-connected neural network with 1 linear layer + softmax activation (predictor model) to produce the probability score class labels. This predictor model is trained to minimize

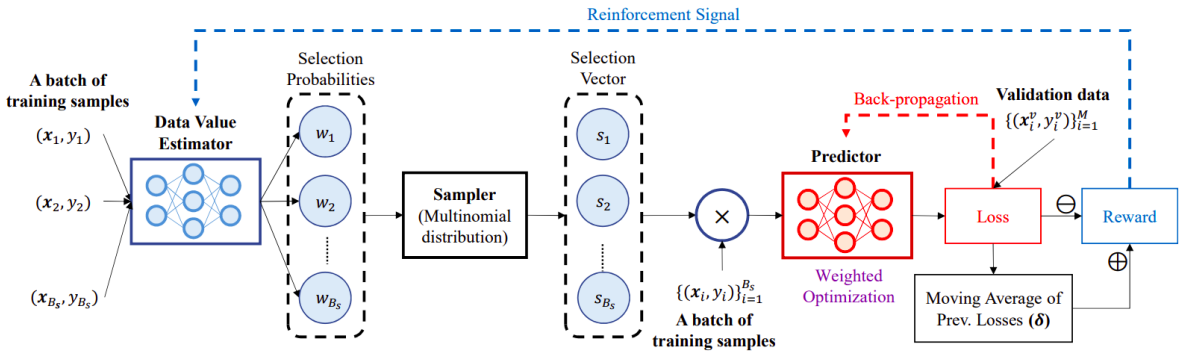

Figure 1: Block diagram of the DVRL framework for training. Adapted from [5] pg.3

cross-entropy loss ($\mathcal{L}_f$) on training set $\mathcal{D}$.

$$f_\theta = \arg\min_{\hat{f}\in\mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} h_\phi\left(\mathbf{x}_i, y_i\right)\cdot \mathcal{L}_f\left(\hat{f}\left(\mathbf{x}_i\right), y_i\right)$$

where, f_θ can be any trainable function with parameters θ, such as a neural network. The data value estimator model $h_\phi : \mathcal{X}\cdot\mathcal{Y} \to [0,1]$, on the other hand, is optimized to output weights that determine the distribution of selection likelihood of the samples to train the predictor model f_θ. We formulate the corresponding optimization problem as:

$$\begin{aligned} \min_{h_\phi} \quad & \mathbb{E}_{(\mathbf{x}^v, y^v)\sim P^t}\left[\mathcal{L}_h\left(f_\theta\left(\mathbf{x}^v\right), y^v\right)\right] \\ \text{s.t.} \quad & f_\theta = \arg\min_{\hat{f}\in\mathcal{F}} \mathbb{E}_{(\mathbf{x},y)\sim P}\left[h_\phi(\mathbf{x},y)\cdot\mathcal{L}_f(\hat{f}(\mathbf{x}),y)\right] \end{aligned}$$

The second module which is part of DVRL is a multi-layer perceptron (data value estimator), which acts as our agent in the reinforcement learning task. This network determines the distribution of selection likelihood of the samples to train the predictor model f_θ. This selection probability is used in the sampler which, in turn, generates the batch of training samples which are given as an input to the predictor model.

We use the REINFORCE algorithm [6] to optimize the policy gradients, with the rewards obtained from a small validation set that approximates performance on the target task.

3.1 Algorithm

We utilize data value estimation of DVRL algorithm for training text classification model using the framework as provided in algorithm 1.

Algorithm 1 DVRL Training for Text Classification

Input: Text Sequences split into train, valid and test sets

1: Sample a mini-batch of size 'N' from the dataset $[(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n))]$
2: Encode the sequences in mini-batch by passing through encoder of Transformer-based model
3: Feed this mini-batch as input to the DVRL model, and get the selection probability $(S_1, S_2, \ldots S_N)$ as output
4: Using a sampler with multinomial probability distribution sample 'k' elements $(S_1, S_2, \ldots S_k)$ from the selection probability given by the DVRL model
5: Get the training samples corresponding to these 'k' elements $[(x_1, y_1), (x_2, y_2) \ldots (x_k, y_k))]$ selected by sampler
6: **for** $i = 1 \ldots k$ **do**
7: Do a forward propagation on this example on the predictor model
8: Calculate the loss and backpropagate it through the text classifier model to update its parameters
9: **end for**
10: Update the DVRL model parameters by using the loss of Text classifier model as the reward function

3.1.1 Predictor Model Architecture

We use the encoder of the following pre-trained models to generate sequence representation for the predictor model. The predictor model includes a combination of linear layers + activation functions to generate a probability score for the class labels.

BERT$_{BASE}$ [7]: 12-layer BERT-base model which was pre-trained on BookCorpus and English Wikipedia and released by Google; We used case-senstive version of the model.

RoBERTa [8]: 12-layer RoBERTa model which was pre-trained on multiple datasets, viz. BookCorpus, English Wikipedia, CC-News, OpenWebText, and WebText dataset. This was released by Facebook.

Longformer [9]: Transformer model for long sequences which uses a combination of a sliding window (local) attention and global attention. We use the BASE model which can handle sequences of maximum length of 4096. This was released by AllenAI.

MiniLM [10]: 12-layer multilingual pre-trained model which uses deep self-attention distillation for model compression. This was released by Microsoft.

4 Experiments

4.1 Data

To validate the effectiveness of DVRL approach, we have used custom Course5 dataset for Sequence classification. The sequences in this data are truncated to a maximum of 512 token length and further used for model training. We have truncated the sequences to a set a maximum sequence length of 512 tokens and have padded the smaller text sequences for equal lengths. The dataset is split into train, validation and test sets which account to 2200, 600 and 1100 data points respectively. We have added 20% noise to training data by randomly mapping the sequences with incorrect labels for experiment purposes.

4.2 Experimental Setup

We use pre-trained models of BERT, RoBERTa, Longformer and MiniLM which were fine-tuned on specific datasets to encode the representations of our input sequences. These models have been fine-tuned with learning rate of 4e-5 and then apply AdaX [11] for optimization with linear schedule warmup strategy. For the sequence classification task, we apply single linear layer + softmax activation on top of the encoded representations for classification. This model is passed as a predictor model in DVRL and 100 iterations are done with a mini-batch size of 256. For the data value estimator model, we use a fully connected network with 5 layers for which the parameters are trained. All experiments are performed on a server with two Nvidia Tesla V100 32 GB GPUs.

4.3 Identifying and Removing High-Low value samples

Identification of high-low value samples can help in certain ways:

1. Removing of low-value samples in the training data can improve the predictor model performance and especially in the cases where the training dataset contains corrupted/noisy samples.
2. Removing of high-value samples from the training data would deteriorate the overall performance of the predictor model.

As noisy samples hurt the performance of the predictor model, an optimal data value estimator with a clean validation set should assign lowest values to the noisy samples. With the removal of samples with noisy labels, the performance should either increase, or at least decrease much slower, compared to removal of samples with correct labels. We train the data value estimator along with the predictor model and plot the change in accuracy for various data encoder models in 2. We find that different encoder models perform much differently in similar environments. For most of the models, the accuracy doesn't seem to decrease to a large extent with removal of low-value samples, whilst reduces significantly when we remove high-value samples. Longformer achieves best performance without much change in accuracy while BERT performs badly when we remove most of the samples.

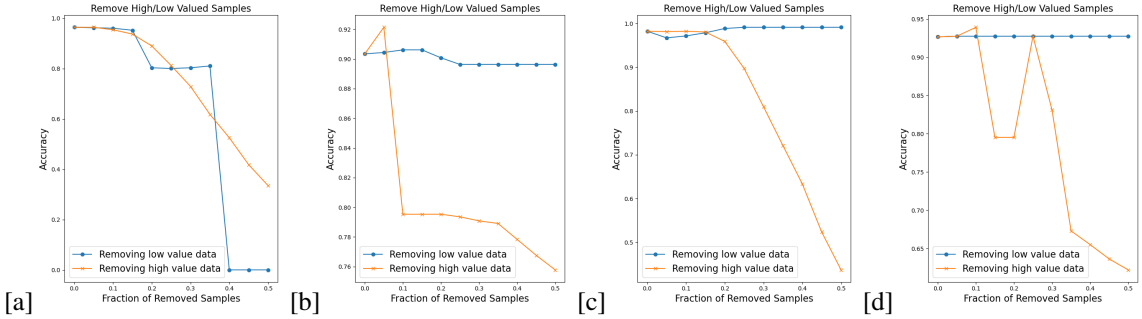

Figure 2: Identification of high-low value samples - (a) BERT(Base)-cased (b) RoBERTa (Base) (c) Longformer (d) MiniLM

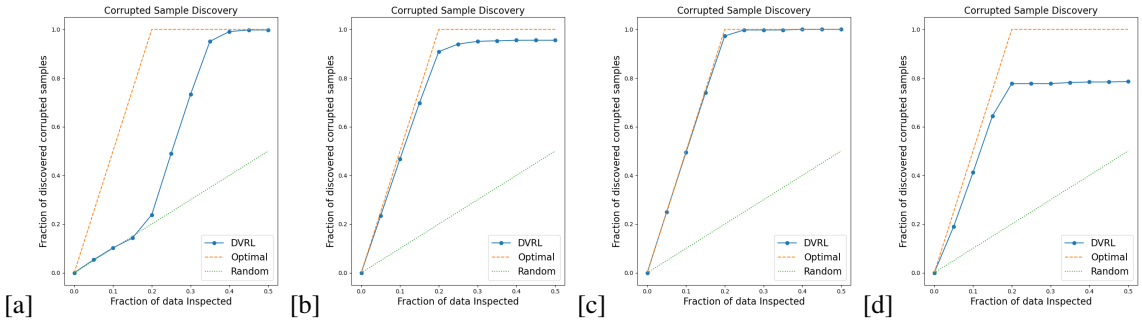

Figure 3: Corrupted sample discovery - (a) BERT(Base)-cased (b) RoBERTa (Base) (c) Longformer (d) MiniLM

4.4 Corrupted Sample Discovery

We added noise to training data by corrupting 20% of the training samples in order to reflect noisy data which may usually happen in case of cheap label collection techniques / human inconsistencies. An automated corrupted sample discovery method would be highly beneficial for distinguishing samples with clean vs. noisy labels. Data valuation can be used in this setting by having a small clean validation set to assign low data values to the potential samples with noisy labels. With an optimal data value estimator, all noisy labels would get the lowest data values. We find the corrupt samples using different predictor models and similar architecture of data value estimator, and observe that most of the predictor models are able to identify those samples correctly. Longformer, in this case, is close to optimal point and identifies almost all the corrupted samples within 20% of data inspected (See figure 3).

4.5 Robust learning

DVRL can be used to do robust learning which provides good predictor model behaviour on noisy as well as clean datasets. Ideally, noisy samples should get low data values as DVRL converges and a high performance model can be returned even without removal of low data values. We compare the different encoders of the predictor model for change in accuracy with DVRL which enables us to generate better predictor models even in case of both noisy as well as clean data (See table 1).

Predictor Model	Accuracy without DVRL	Accuracy with DVRL
BERT	0.9257	0.9643
RoBERTa	0.9058	0.9097
Longformer	0.9405	0.9884
MiniLM	0.8614	0.9392

Table 1: Robust learning with noisy labels in training data

5 Discussion and Future Work

In our experiments, we tried multiple transformer models to encode text data and observed their performance on the classification task. Using the figure 2, we can infer that removing low-value data points does not have any significant impact on the accuracy, whilst, removing high-value data points results in degraded performance of the models. Using this data value estimator, we are able to correctly identify the corrupt and noisy samples from the training data (See figure 3). The use of DVRL improves accuracy by up to 7% on most of our classification models (See table 1).

For future work, we would want to use the encoder models with classification head directly as a predictor model in DVRL framework as opposed to our current approach of just using the classification layer on encoded representations. In addition, we would also want to perform an ablation study of various parameters of our framework including the noise rate, length of sequences and volume of the data.

References

[1] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[2] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions, 2017.

[3] Amirata Ghorbani and James Zou. Data Shapley: Equitable valuation of data for machine learning, 2019.

[4] Ruoxi Jia, Xuehui Sun, Jiacen Xu, Ce Zhang, Bo Li, and Dawn Song. An empirical and comparative analysis of data valuation with scalable algorithms, 2019.

[5] Jinsung Yoon, Sercan O Arik, and Tomas Pfister. Data valuation using reinforcement learning. *arXiv preprint arXiv:1909.11671*, 2019.

[6] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[7] 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.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[9] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[10] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[11] Wenjie Li, Zhaoyang Zhang, Xinjiang Wang, and Ping Luo. Adax: Adaptive gradient descent with exponential long term memory. *arXiv preprint arXiv:2004.09740*, 2020.