SelfExplain Text Classifier Reproducibility Report

Murray Kang, Xuweiyi Chen, Hao Luo
University of Washington
{haoqik, xuweic}@uw.edu

Reproducibility Summary

Scope of Reproducibility

There are three main claims we have tested for the model of SelfExplain. First, the stability of SelfExplain should be preserved within similar sentence. Second, we expect to see SelfExplain will not damage the performance of XLNet and RoBERTa. Third, we want to show these results are consistent across different datasets, which will also reinforce the second result.

Methodology

First of all, we used the author's code to train the model. However, the code provided by the author is far from complete. We encountered and investigate numerous bugs when we follow the instructions provided by the author. Even we successful trained the model with the provided code after we fixed all the bugs, the evaluation and test part of the code is also incomplete. We spent some extra time in order to follow the instructions from the paper.

Results

Based on each of our experiments, we found no conclusive evidence that LIL or GIL will boost the performance of XLNet or RoBERTa. However, LIL and GIL certainly will not harm the performance of XLNet and RoBERTa. These results can be infered from how the layers are constructed. We find the SelfExplain is stable, introducing SelfExplain will provide reasonable performance for different datasets.

What was Easy

Since the code we used was based on the authors', it helped us better understand the structure and ideas of SelfExplain, as well as all the concepts mentioned in the paper. For the transformers we used in the implementation, there are some examples and documentations of the API we need to use, so it really helped a lot in solving several problems about the codes. In terms of the ideas of the paper, using interpretable layer to interpret a sentence is intuitive. For both LIL and GIL, the original paper provide a intuitive explanation, which benefits us greatly for reproducing the code and understandings.

What was Difficult

We faced many difficulties in the process of reproducing the experiment. Maybe due to the difference and incompatibility of our PCs with the authors' code, it took us many days to revise and debug the original code to make it able to run on our end. Also, there are few documentations and information that could help us change the transformer to RoBERTa on the Internet, which is needed for our additional experiment. In terms of the ideas of the paper, although the construction of the two interpretable layers is intuitive, the reason why and how the interpretable helps improve the performance of the classifier is still hard to understand.

Communication with Original Authors

We have contacted the original authors though Dr. Yulia Tsvetkov. We are fortunate enough to discuss the paper with the author Dheeraj Rajagopal, who resolve some of our confusions and provide instructions on how we should move forward since part of the code for testing is unavaliable. His instructions have been proved as very effective. We want to thank Dr. Yulia Tsvetkov and Dheeraj Rajagopal for their swift responses.

1 Introduction

SelfExplain is a neural text classifier that can explain the text prediction using phrase-base concepts. The significance of this paper about SelfExplain is improving the interpretibilities of Neural network. In many areas including NLP, people always want to understand the model decision making procedure of Neural network models. In the area of Trustworthy AI, people found many bias that has been encapsulated in Neural network models. This paper combines two common approaches for better interpretibilities: (1) explain predictions based on the internal of model; (2) built-in interpretable layer. From our pespectives, this paper proposed two layers in order to provide better transparency for transformer-based NLP classifier: (1) Locally interpretable Layer analysis the contribution of each phrase to the final label distribution; (2) Globally interpretable Layer search the most influential concept from the training data for each input. Note that these two layers can be combined with transformer-based classifier and provide even better performance compared with only use transformer-based classifier. It suggests to people that these two layers facilitates better interpretibility and improve the trustworthyness.

It appears to us that this methodology might be able to convert for other Neural network if we consider each feature captured by YOLO as scores. However, it requires a lot of engineering and deeper knowledge. This paper suggests a innovative way of promote interpretibility of Neural model by introducing search-based and similarity-based layers. It is also interesting to discuss why the prediciton power increases by incorporating LIL and GIL layer.

2 Scope of Reproducibility

Since this paper is largely regarding on interpretibility, we want to see its interpretibility. However, we only have three people. The best practice is to test stability of the top interpreted words. In other words, we will train a model and utilize the model to predict a set of very similary sentences.

Then we want to test XLNet-Base model and RoBERTa-Base model with LIL and GIL layers. We want to see the performances of the models do not suffer as the paper claimed.

Since the original dataset provides only SST-2, we want to train this model across SST-5 and SUBJ and compare the accuracy I acquired and the data from the paper. If time permits, we will train this model with extra datasets.

2.1 Addressed Claims from the Original Paper

1. The transformer-based classifier with LIL and GIL layers can interpret sentences stably.
2. The transformer-based classifier with LIL and GIL layers perform consistently better across datasets with fixed hyperparameters with a few exceptions.
3. The transformer-based classifier with LIL and GIL layers perform better than only transformer based classifier in terms of accuracy

3 Methodology

We used the author's code, but the author's code is incomplete. We build upon the code provided by the author. We provide clear documentations for other people to use and we benchmark and develop our experiments on Google Cloud Platform. We will tailor our model to attu if time permits.

3.1 Model Descriptions

SelfExplain is a novel self-explaining model that explains a text classifier's predictions using phrase-based concepts. In the SelfExplain architecture, firstly, there is a transformer encoder that encoded the input and its relative non-terminals. After that, it used three layers after regularization to make prediction, a linear layer, a globally interpretable layer and a locally interpretable layer.
In the globally interpretable layer, it aims to figure out the most influential concepts on the model's prediction in the training set for each data sample. To be more specific, firstly, there is a concept store that contains all concepts in the training set, in which each candidate q_k is represented by a mean pooled representation of its constituent words.

$$q_k = \frac{\sum_{w \in q_k} e(w)}{len(q_k)}$$

Since the model is a finetuned for downstream task, all candidate representation q_k are re-indexed after every fixed number of steps. Next, the model used the Maximum Inner Product Search (MIPS) to retrieve the top K influenced

concepts $q_{k\,1:K}$ from the concept store based on the cosine similarity function:

$$d(x, Q) = \frac{\mathbf{x} \cdot q}{\|\mathbf{x}\|\|q\|} \qquad \forall q \in Q$$

After identifying the influential concepts from the training data, the model learned it end-to-end via back propagation. Our inner product model for this globally interpretable layer is defined as follows:

$$p(q|\mathbf{x}_i) = \frac{exp(d(\mathbf{u}_{\mathbb{S}}, q))}{\sum_{q'} exp(d(\mathbf{u}_{\mathbb{S}}, q'))}$$

In the locally interpretable layer, it aims to calculate the relevance score of each input concept that is relative to the prediction to quantify the contributions. Firstly, it computes a local relevance score for all input concepts from a sample. For each non-terminal nt_j, there is a score that qualifies the contribution of each nt_j to the label and then the model compared it to the contribution of the root node $nt_{\mathbb{S}}$. Finally, it chose the phrases $\mathcal{C}_L$ that contributed most to explain the predictions locally. To be more specific, given a encoder, LIL computes the contribution from nt_j to the final prediction, and then it built a representation of the input without contribution of phrase nt_j and use it to score labels:

$$t_j = g(\mathbf{u}_j) - g(\mathbf{u}_{\mathbb{S}})$$
$$s_j = \text{softmax}(\mathbf{W}_v \times t_j + \mathbf{b}_v)$$

where g is a $relu$ activation function, $t_j \in \mathbb{R}^D$, $s_j \in \mathbb{R}^C$, $\mathbf{W}_v \in \mathbb{R}^{D \times C}$, s_j is a label distribution without the contribution of nt_j.
Given this, the relevance score $\mathbf{r}_j$ is the given by the difference between the classifier score fore the predicted label based on entire input and the label score based on the input without nt_j:

$$\mathbf{r}_j = (l_Y)_{i|i=P_c} - (s_j)_{i|i=P_c}$$

SelfExplain is trained to maximize the conditional log-likelihood of predicting the class at final layers: LIL, GIL and linear (for label prediction). It has been found that regularizing models with explanation specific losses improves inherently interpretable models for local interpretability (Melis and Jaakkola, 2018). SelfExplain extends this idea for both global and local interpretable output for our classifier model, so we regularize the loss through GIL and LIL layers by optimizing their output for the end-task as well.

For the LIL layer, we use a weighted aggregated representation over s_j and compute the log-likelihood loss as follows:

$$l_L = \sum_{j, j \neq \mathbb{S}} w_{sj} \times s_j, w_{sj} \in \mathbb{R}$$

and the overall loss of the whole training set is as follows:

$$\mathcal{L}_L = -\sum_{c=1}^{C} log(l_L) \times y_c$$

For the GIL layer, we firstly aggregate the scores over all the retrieved $q_{1:K}$ as a weighted sum, then pass it through an activation layer, a linear layer, and a softmax layer to compute the log-likelihood loss as follows:

$$l_G = \text{softmax}(\mathbf{W}_u \times g(\sum_{k=1}^{K} \mathbf{w}_k \times q_k) + \mathbf{b}_u)$$

and where $\mathbf{W}_u \in \mathbb{R}^{D \times C}$, $\mathbf{w}_k \in \mathbb{R}$ and g represents $relu$ activation function. Here, the global interpretable concepts are denoted by $\mathcal{C}_G = q_{1:K}$, and l_G represents the softmax for the GIL layer. The overall loss of the whole training set is as follows:

$$\mathcal{L}_G = -\sum_{c=1}^{C} log(l_G) \times y_c$$

In the training process, we optimize for the following joint loss:

$$\mathcal{L} = \alpha \times \mathcal{L}_G + \beta \times \mathcal{L}_L + \mathcal{L}_Y$$

where $\mathcal{L}_Y = -\sum_{c=1}^{C} log(l_Y) \times y_c$ is the overall loss of the whole training set for the linear layer . Here, α is and β are regularization hyper-parameters. All loss components use cross-entropy loss based on task label y_c.

Notice that we did not introduce the idea of concepts for GIL, phrase in LIL and labels used for prediciton because the model has some flexibilities on these choices. Concepts in GIL are samples from the train set. Phrases in LIL are tokens in the sentence, but we could use other pieces of sentence for LIL. Label used for prediction is the <CLS> tokens at the beginning of the last hidden layer of transformer model, which the paper has proposed to use RoBERTa [5] and XLNet [10], which contains 124 millions trainable parameters and 117 millions trainable parameters respectively.

3.2 Datasets

We used SST-2 [8], SST-5 [8], SUBJ [6] and CoLA [9] fine-grained classification dataset from the Stanford NLP, UWNLP, NYU and DeepAI website for benchmark and perform experiments for each of three claims we discussed in the section of Scope of Reproducibility. Note that SST-2, SST-5 and SUBJ have been tested in the original dataset. The CoLA dataset is an additional dataset that we want to test our hypotheses in order to reinforce our arguments. SST-2 dataset is a sentiment classification task which predict the sentiment of movie review sentences as a binary classification task. SST-5 dataset is a fine-grained sentiment classification task from Stanford as well as SST-2 dataset., which extend from 2 differnt classes to 5 different classes. SUBJ is another binary classification dataset based on both subjective and objective classifications. CoLA is another classification dataset from 23 linguistics publications, which is annotated by expert for acceptability (grammaticality). Note that CoLA dataset is very different from SST-2, SST-5 and SUBJ because CoLA is designed for linguistics acceptability classification and the other datasets are utilized for sentiment analysis. For the purpose of SelfExplain, we recognize the natural choice of task is sentiment analysis. However, we want to argue that SelfExplain will become even more convincing if SelfExplain could detect similar grammatical error through GIL and express the significant phrase through LIL. The statistics of datasets are presented in 1.

Dataset	C	L	Train	Test
SST-2	2	19	68222	1821
SST-5	5	18	10754	1101
SUBJ	2	23	8000	1000
CoLA	2	6	8551	527

Table 1: Dataset statistics, where C is the number of classes and L is the average sentence length.

Note that we may utilized different version of datasets from the original paper. Note for SUBJ, we utilized the first 8000 rows of dataset as training data and the next 1000 rows of dataset as test data. It has not been specified in the original paper.

3.3 Hyperparameters

In the paper, the original authors state the hyperparameters very clear. The parameter K is the number of influential concepts which will be chosen by GIL. The original paper consider $K = 5, 10$. For hyperparameters α and β for training, the original authors test $\alpha, \beta = \{0.01, 0.1, 0.5, 1\}$. It is determined that most experiment has been done based on $\alpha, \beta = 0.01$. Based on the caption of Table 2 from the original paper, we determined all the different hyperparameters, which is for our second test. However, there is one table without hyperparameters and we want to reproduce the table because its significance. The original authors claim that the hyperparameter K did not make noticeable difference. Based on reading the numbers from table 3 of the original paper, which we want to reproduce, $K = 5$, $\alpha, \beta = 0.01$ are being determined. Also, the learning have been specified as $2e - 5$. This value is utilized across all experiments and obtained from the original code in the existing repository. Another important hyperparameter is the number of max epochs. The original implementation has set it to be 5. It is appropriate because we have already utilized pre-trained transformer models. The PyTorch Lightning [3] will save the best three checkpoints. We will use the checkpoint with best validation accuracy to perform our experiments.

Note that this paper utilized pre-trained model based on RoBERTa [5] and XLNet [10]. As we find in the original papers, XLNet has a 0.2 dropout layer and RoBERTa has no dropout. Also there is no dropout in SelfExplain. The hidden size for RoBERTa-base pre-trained model is 768 and the hidden size for XLNet-base-cased pre-trained model is 512.

3.4 Implementation

We determine that we are between writing our own code and using exising code because the given code contains many bugs and it does not support training with RoBERTa. We fixed all the bugs related to preprocess as well as training procedure. Also we tailored the existing code for RoBERTa, which the original authors claim that it could only theorectically work and not being optimized. Based on our solution, the efficiency of training procedure for RoBERTa is identical to training with XLNet. We have contacted the original authors and will file a push request to the public repository for SelfExplain.

We made some changes based on the the author's code. Also, we used different dataset (CoLA) to reinforce the experiment mentioned in the paper. The original author's code repository can be found on the Github [1].

3.5 Experimental Setup

Following the paper, we used two different transformer encoders configurations as our base model: RoBERTa [5] - a robustly optimized version of BERT and XLNet [10] - a variant model based on Transformer-XL architecture. Then, we used the two encoders without the GIL and LIL layers as the baselines, and we generated parse tree to extract target concepts for the input. Also, we maintained the weights and hyperparameters from the pre-trained encoders, and fine-tuned the parameters on the GIL and LIL modules.

We experimented on two different numbers of global influential concepts K $= \{5, 10\}$, different transformer encoders, different combinations of layers used. We performed all experiments on two separated K-80 GPUs on Google Cloud Platform and 2 vCPU (Intel Broadwell) using 13 GB of memory.

We have cleaned the repository and provided detailed guidelines on our own repository [2]. We have explained and introduced the command to run each experiments in the ReadMe file. Note we require Linux system in order to train the code. Otherwise, there will be errors for Windows due to bugs within PyTorch Lightning library, which is critical because of its scalability for multiple GPUs.

3.6 Computational Requirements

The original paper had access to two NVIDIA V-100 GPUs, but we only have access to either RTX 3080 and two K-80 provided by GCPs. However, the model is too large for RTX 3080 and the code is not compatible with windows and therefore we have primary trained our experiments using K-80s. Average runtime for each epoch varies significant for different training set. For SST-2, the average runtime for each epoch is approximately 1 hour. For SST-5 and SUBJ, the average runtime for each epoch is approximately 30 minutes. For CoLA, the average runtime for each epoch is 10 minutes. We have performed more than 30 experiments, each with 5 epochs. The total training time is more than 24 hours without considering failures. We recommend the CPU have access to at least 12 GB of memory as well as for 12 GB memory for GPU. In order to store all models, it will require more than 200 GB of disk space.

Our team have access to 2 RTX 3080s, which we believe they are sufficient for the purpose of training. However, the existing code structure requires Linux operation system. We spent two days in order to separate the existing code struture with PyTorch Lightning, but the entire code structure has been adopted based on PyTorch Lightning which causes different bugs. Therefore, we seeks to use attu to train. At the end, we settle a solution of using two GCPs. First, we only have access to 1, but the training procedure takes a lot of time and therefore we ended up setting up two GCPs. According to the original authors, this exising code is only optimized for XLNet, and training using the RoBERTa of their method will exceed 48 hours. We modified the existing code, which theoretically works for RoBERTa, and the efficiency is almost identical to XLNet. This effort has reduced the training requirements greatly for RoBERTa.

4 Results

As we stated in the previous section: Scope of reproducibility, we want to test three claims from the original paper, which we will demonstrate our results in separate section. We believe our experiments are within $1 - 2$ % difference compared to the results from the original paper. Even some patterns of the data are different, we still believe there is no conclusive reason to reverse the claims made by the original paper. Note that this original data show consistency through all three experiments.

4.1 Result 1

We shall restate our hypothesis: The transformer-based classifier with LIL and GIL layers can interpret sentences stably. For interpret sentences stably, we mean that LIL and GIL should find similar concepts and phrases for similar sentences. If two sentences are very closely related to each other, we should expect GIL report similar concepts from the the train set. We should also expect the same from LIL.

In order to construct the experiments, we have constructed similar sentences and analysis them using LIL and GIL. We have provide two pairs of similar sentences in 2.

[1] https://github.com/dheerajrajagopal/SelfExplain

[2] https://github.com/XuweiyiChen/SelfExplain

Input	Top LIL interpretations	Top GIL interpretations
enter the beautiful and mysterious secret agent petra schmitt	the beautiful and mysterious secret agent	(1) passion and attitude, (2) passionate and truthful
enter the mysterious and beautiful secret agent petra schmitt	the mysterious and beautiful secret agent	(1) passion and attitude, (2) passionate and truthful
visually imaginative , thematically instructive and thoroughly delightful , it takes us on a roller-coaster ride from innocence to experience without even a hint of that typical kiddie-flick sentimentality	to experience without even a hint of that typical k idd ie - fl ick sentiment ality	(1) serenity and poise (2) annoying
thoroughly delightful , thematically instructive and visually imaginative , it takes us on a roller-coaster ride from innocence to experience without even a hint of that typical kiddie-flick sentimentality	to experience without even a hint of that typical k idd ie - fl ick sentiment ality	(1) serenity and poise (2) annoying

Table 2: Interpretations Comparisons of two pairs of similar sentences. $\alpha, \beta = 0.1, K = 5$ for hyperparameters.

The outcome of the experiment is very promising and convincing. We should expect even more clear and smaller pieces of LIL interpretations based on Byte Pair Encoding (BPE). For first pair of sentence, we swapped the word "beautiful" with "mysterious" and it reflect in the LIL interpretations because it is most relevant to the sentiment of the sentence. For the latter example, the swapped part does not appear in the LIL interpretations but it is also stable across similar sentences. For GIL interpretations, it appears very stable based on the MIPS algorithm. Note that this result is obtained by RoBERTa pre-trained model. This behavior is expected because LIL perform a algorithm to find the most significant absence out of all option (phrases) and GIL perform a MIPS algorithm. With this evidence, we will conclude that the LIL and GIL can support transformer-based classifier with stability. Note that we performed ten pairs of similar sentences and the performances are consistent.

4.2 Result 2

In this section, we want to provide our results in order to test the second claim made by the original authors: the transformer-based classifier with LIL and GIL perform consistently better across datasets with fixed hyperparameters with exceptions. For results, we present all the data in the 3.

Model	SST-2	SST-5	SUBJ	CoLA
XLNet	93.23	52.4	95.50	69.26
SELFEXPLAIN-XLNet (K=5)	92.89	52.31	95.80	69.26
SELFEXPLAIN-XLNet (K=10)	93.23	51.95	96.00	69.26
RoBERTa	93.07	54.90	95.73	87.10
SELFEXPLAIN-RoBERTa (K = 5)	93.58	54.99	95.83	87.47
SELFEXPLAIN-RoBERTa (K = 10)	93.92	54.99	95.70	86.91

Table 3: performance comparison of models with and without GIL and LIL layers. All experiments used the same code architecture. We use the development set for SST-2 results and test sets for SST-5 and SUBJ. $\alpha, \beta = 0.1$ for all the experiments

For nearly all cases, the original paper provide measurements that are consistently better than our experiments. We determine this is not a major issue for four reasons: the current code has been optimized for efficiency, which we acquired this information through our communication with the original author; the learning rate might be different: we select different seeds, which might lead to different efficiency; the $1 - 2$ % differences of performances are not important because we should focus on patterns within the data we acquired through experiments because we care more about explainability.

Here are our findings. In the original paper, the authors claims that SELFEXPLAIN-RoBERTa and SELFEXPLAIN-XLNet consistently show competitive performance compared to the base module except for some special case about TREC-6 [7]. We do not quite agree with this claim. From our experiments, we see performances drop by a little when increasing K for other dataset such as SST-5 using XLNet. For different datasets such as SUBJ, the performances are improved with LIL and GIL. Even though the trend remain unclear in our experiments, we do find that the introduction of LIL and GIL with different K does not harm the performance of XLNet and RoBERTa significantly. Our observation accords with the claims made by the original authors at a certain level, and we do not find conclusive evidence that introducing LIL and GIl in the transformer-based model will always provide better performances across different datasets. However, we cannot reverse their statement as well. The original authors foresee the possibility of exceptions and we do not observe significant loss due to the introduction of LIL and GIL.

Notice the accuracy for CoLA are significant different between XLNet based classifier and RoBERTa based classifier. We believe RoBERTa as a variant of BERT have a natural advantange for grammer related task. We also observed the accuracy is the same for XLNet based classifier. We will discuss this matter in section 4.4 because CoLA is a dataset that is not utilized in the original paper.

4.3 Result 3

In this section, we want to test the third claim of the original paper: the additional LIL and GIL can be corporated with transformer-based classifier with no performance lost and even boost the performance of the classifier in terms of accuracy. We followed the experiments peformed by the original paper. We trained additional model in order show each single layer of interpretable layer: LIL or GIL will not harm the performance of the model for both XLNet and RoBERTa. We want to present the data we acquired in the 4.

Dataset	Accuracy
XLnet-Base	92.23
SELFEXPLAIN-XLNet + LIL	93
SELFEXPLAIN-XLNet + GIL	93.69
SELFEXPLAIN-XLNet + GIL + LIL	92.89
RoBERTa-Base	94
SELFEXPLAIN-RoBERTa + LIL	93.75
SELFEXPLAIN-RoBERTa + GIL	93.07
SELFEXPLAIN-RoBERTa + GIL + LIL	93.58

Table 4: Comparisons of Transformer-based classifier with Interpretable Layers. Hyperparameter $K = 5$, $\gamma = 0.1$, $\lambda = 1$ if the corresponding layer exists.

Based on the data in 4, we see sometimes the introduction of LIL or GIL will influence the performace negatively and sometimes influence the performance positively. However, it does not influence the accuracy significantly. The original paper claims that K does not influence the performance, and we choose $K = 5$. The main hypothesis we tested here is consistent with what the original authors claimed. The introduction of LIL or GIL to transformer-based classifier will cause no significant performance losses and even boost the performance of the classifier.

4.4 Additional Results not Present in the Original Paper

For addtional result, we introduced a new dataset CoLA. It has been treated as a normal dataset along with other dataset utilized in the original paper. In RoBERTa, we see that introducing LIL and GIL has similar affect for other datasets. However, the accuracy remain as 69.26 for XLNet even we performed many other experiments. First, we believe that XLNet require smaller concepts and thus we use $K = 2$ for Hyperparameter. Then we utilized $\lambda = 0.01, 0.1, 1$ as well as $\gamma = 0.01, 0.1, 1$. The accuracy remains as the same. However, we find that XLNet is introduced with purpose of dealing with longer context [2]. XLNet will have worse performance at dealing with short context. It is possible that the Implementation of XLNet-base pretrained model will filter all the short sentences. This is quite interesting observation and which we will discuss the impact of this observation further in the next section.

Also, the original authors did not provide code for RoBERTa because they do not have time to optimize for RoBERTa. We followed the advice of the original authors and successful implement RoBERTa-based training process with almost identical training efficiency as XLNet.

For hyperparameters, since the max epoch is 5, we wonder if the limited epoch restrain us of finding the performance deduction of LIL and GIL, we performed the training using 20 epochs for SUBJ and SST-5 because training with SST-2

is not feasible due to the limitation of the time. The accuracy is consistent from the first to the last epoch within 1.5 % range.

Based on all the additional results we have performed, we believe that the results from the original paper are quite convincing even though we may disagree when they claim GIL and LIL will always boost the performance in terms of accuracy. However, they also foresee some exceptions.

5 Discussion

Based on our reproduced experiments, we find that the claims made the orignal authors are quite convincing. We are able to reproduce most of the results. Our approach is quite strong when it comes to test their claims written in the original paper. However, we do not have time to perform experiments on those claims that are not directly written in the original paper. We would like to test 100 epochs for the training process and see how the accuracy will behave. However, 100 epochs will take all our time to perform all the tests we want to run. Also, we want to train model to test different hyperparameters specifically γ and λ. With that being said, we do generally agree that SELFEXPLAIN could potentially offer some new pespective of explainability of NLP neural network.

5.1 What was Easy

First, we want to appreciate code from the original authors, which built up the foundation of the SelfExplain model, when we faced any confusion about the concepts mentioned in the paper, we can refer to the codes to deepen our understanding of the model. Also, the code style of the authors is clear and easy for us to understand the ideas of each part. Next, the dataset used in the experiments are easy to obtain since they are all open sources on the websites. In the training step, the authors used PyTorch lightning library to build the Trainer, and there were a lot of clear examples and API documentation which helped us better understand the structure of the codes.

5.2 What was Difficult

We experienced several challenges when reproducing this paper. In the process of reading through the whole paper, there are some concepts and intuitive that are hard for us to understand. Using maximum inner product search (MIPS) to retrieve the top relevant K concepts from the concept store is one of them. Therefore, we had to read the paper of MIPS [4], and found out the methodology and intuitive of it. The next one concerning the methodology of the paper is the reason of using regularization with the log-likelihood loss from LIL and GIL in training the linear classification layer. So, we read through the paper regarding the regularizing models with explanation losses to figure it out [1].

In terms of the difficulty of coding, there were still some challenges we faced. Firstly, the authors' codes were broken, and we had to fix it in the a couple of weeks. Also, we established two computing K-80 GPUs on Google Cloud Platform to train the models. Also, the codes for RoBERTa transformer is not completed, and there are few useful information we can find on the websites that are suitable to our case. Also, the authors cannot access to their original codes for RoBERTa. Thus, it took us many days to figure out how to use RoBERTa in the SelfExplain model by multiple searching and experiments.

5.3 Recommendations for Reproducibility

In terms of the concepts, ideas, and intuitive in the paper, We recommend that it would be better to read the paper [1] that discussed and proved why regularizing models with explanation certain losses can improve interpretable models for local interpretability in the area of computer vision. The SelfExplain model smartly extended this idea for both global and local interpretable output for the classifier model, and showed the actual improvement of doing this.

Since there were many incompatibilities happened with the authors' code and our local Python interpreter, for the researchers who would use the author's code, in which some methods and packages are out-of-date, we recommend you need to install the same version of the packages used if possible. For those packages that the specific old version cannot be installed anyone, I recommend you can refer to our codes to revise them on your end.

Also, for the researchers who do not have access to any powerful computing GPU, we recommend you can use Google Cloud Platform to train the model, espcifally for the dataset of SST-2, which has 60,000 sentences, since it will took a long time.

Communication with Original Authors

We have contacted the original authors though Dr. Yulia Tsvetkov. We are fortunate enough to discuss the paper with the author Dheeraj Rajagopal, who resolve some of our confusions and provide instructions on how we should move forward since part of the code for testing is unavaliable. His instructions have been proved as very effective. We want to thank Dr. Yulia Tsvetkov and Dheeraj Rajagopal for their swift responses. Additionally, Dheeraj Rajagopal speaks language that is clear and understandable. We learned that some hyperparameters are left out after training procedure. The stage of infer only utilized MIPS algorithm and transformer-based last hidden state. With this information, we really understand the importance of the claims of the orignal paper. We proposed the idea of SelfExplain space and the idea of kernel with Dheeraj Rajagopal. He thought it is a good idea to take advantage of those parameters which has been trained. We will discuss how to improve the idea of SelfExplain further and we have agreed to speak on phone. It is a quite a amazing communication, and we have benefited from this project greatly.

References

[1] David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 7786–7795, Red Hook, NY, USA, 2018. Curran Associates Inc.

[2] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019.

[3] William Falcon et al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019.

[4] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. *ArXiv*, abs/2002.08909, 2020.

[5] 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. *CoRR*, abs/1907.11692, 2019.

[6] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[7] Dheeraj Rajagopal, Vidhisha Balachandran, Eduard H Hovy, and Yulia Tsvetkov. SELFEXPLAIN: A self-explaining architecture for neural text classifiers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 836–850, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[8] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[9] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

[10] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019.