Investigating the functionality of Key, Query, Value submodules in Attention Mechanism

Ziyi Xie, Letian Jiang
Center for Data Science
New York University
zx1153@nyu.edu, lj2136@nyu.edu

Abstract

Transformer is the state-of-the-art model while the functionality of its attention submodules is still widely discussed. To gain more insights of the attention mechanism, we investigated the importance of individual submodule component in the Transformer model by two methods: 1) adversarial removal of linguistic information on the submodules; 2) rewinding the weights of the submodules. By observing the training curve and importance scores, we concluded that V component is more important and attacking or freezing submodules causes overfitting and underfitting.

1 Introduction

Attention-based models have been one of the latest inventions that improve deep learning model performance in the field of Natural Language Processing. The solely attention-based Transformer model was first proposed in the paper 'Attention is All You Need' [1]. The attention mechanism proposed in this paper contains three major components: Key (K), Query (Q), and Value (V). As shown in figure 1, those three components are used to calculate the Scaled Dot-Product Attention that tells the model how much 'attention' it should pay to certain inputs.

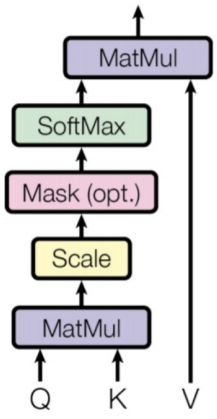

Figure 1: Scaled Dot-Product Attention [1]

There have been many other papers that propose improvements over this original transformer model structure to generalize the use of transformers. However, there haven't been many investigations on

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

analyzing the respective roles of Key, Query, and Value components/submodules. Therefore, in this paper, we will use two prevalent methods to analyze how limiting Transformer's K, Q, V access to linguistic information at different layers in training can affect the overall efficiency of training and performance of the model. The first method involves comparing model performance before and after removing linguistic information from the submodules. The second method involves directly changing submodule parameters in several different ways.

2 Setup

2.1 Model

2.1.1 Transformer

For the transformer model, we use the similar structure from the paper 'Attention Is All You Need' [1], containing a 5 layers and 5 heads encoder and a single layer linear decoder. The task of this transformer model is to assign a probability distribution for the word following a sequence of words in a sentence, or simply next word prediction.

2.2 Data

The dataset we will be using is WikiText-2 [4]. This (silver) dataset contains tokens collected from the set of verified Good and Featured articles on Wikipedia. We then use Spacy to parse every sentence to produce a dataset that contains the Part-of-Speech (POS) and sentence Dependency (DEP) labels for the WikiText-2 dataset. All the datasets will contain a train set and a validation set. There are 102,499 sentences in the train set, and 21,441 sentences in the validation set. During training, the batch size is set to be 20, and the evaluation batch size is set to 10. So there are about 2.4M masked word prediction pairs.

2.2.1 Intermediate Layer Representation Access

Since we are studying how changing K, Q, V could influence the transformer model, we need to know how to extract those intermediate representations. Following the road map in figure 2, we are able to extract K, Q, V representations in each transformer encoder layer and the corresponding matrices W_K, W_Q, W_V that produce the representations

2.3 Evaluation

For evaluation, we mainly focus on looking at the loss and accuracy. The loss is calculated as cross-entropy losses, and the accuracy is calculated by comparing the most likely next word predicted by the model versus the gold standard word.

3 Methodology

3.1 Experiment 1: Adversarial Attacking Method

In this experiment, we want to investigate how removing syntactic information from intermediate representation inside a transformer affects the training of the transformer. In particular, we will be removing the POS or DEP information from the K, Q, V representations at different layers of the transformer model. To remove such information, we use an adversarial model[3] to remove the information. The adversarial model contains 5 fully connected linear layers with leaky relu and batch normalization in between. The task of this adversarial model is to predict the POS or DEP of the next word given intermediate representations from the transformer model.

To better explain how these two models work together to achieve the information removal , lets denote

- F as the transformer model

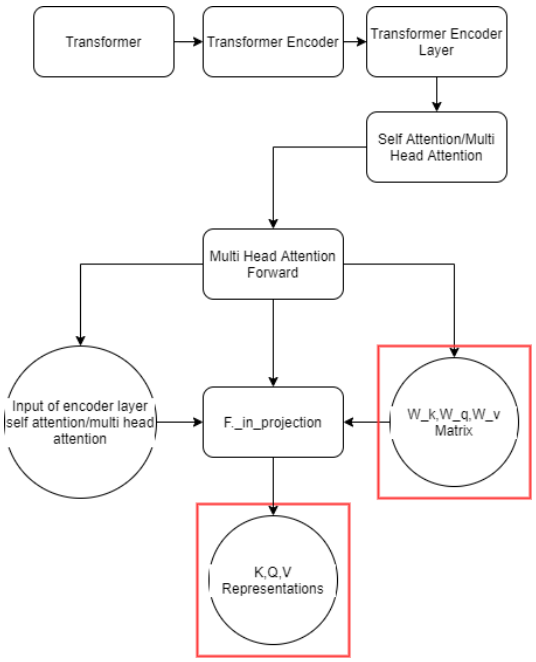

Figure 2: The method used to extract K,Q,V representations and the associated matrix operations to calculate the representations (marked in red boxes)

- A as the Adversarial model
- x as the input of the transformer model (sentences)
- x' as intermediate representation value in the transformer model
- y as the correct label for the next word prediction task
- y' as the correct label for the POS or DEP of the next word prediction task

Now we can define some cross entropy loss functions:

$loss_{adversarial} = Loss(A(x', y'))$,

$loss_{transformer} = Loss(F(x, y))$,

$loss_{total} = loss_{transformer} - \lambda * loss_{adversarial}$, where λ is some scalar constant that represents how strong the presence of the adversarial network is.

The two models will be trained alternatively. So within each training epoch, two backpropogations are done. When backpropagation is done on $loss_{total}$, the transformer model is optimized to decrease the loss on the next word prediction task and change its intermediate representations to increase the loss of the adversarial model because of the minus sign. When backpropagation is done on $loss_{adversarial}$, the adversarial model is optimized to decrease its loss on predicting the POS or DEP of the next word.

So there is a back and forth process between the training of the transformer model and the adversarial model: the adversarial model is trying to learn the POS or DEP information of the next word from the transformer intermediate representations, while the transformer is trying to change its intermediate representations to prevent the adversarial model from effective learning. In this way, the adversarial model is able to prevent the intermediate layer representations in the transformer to contain POS or DEP information.

The choice of x' is the K, Q, V representations at layer 1 through layer 5 of the transformer model. We say that we are attacking the K representations at layer 1 of the transformer if the input of the adversarial model is the K representations at layer 1 of the transformer.

For each choice of intermediate representations, we train both models for 100 epochs and look at the loss of the transformer model for evidence about the importance of POS or DEP information in different sub-modules (K, Q, V) of a transformer model during training. We also train a baseline transformer model without the adversarial model as a comparison.

3.2 Experiment 2: Submodule Weights Editing

In this experiment, we investigate the module/submodule importance by rewinding submodule weights and freezing submodule weights. More specifically, we explore the model performance drop as we modify a specific submodule (K, Q, V) by: 1) rewinding the final submodule weights back to previous submodule weights (before convergence); 2) rewinding the model weights of the submodules back to the convex combination of the initial weights and the final weights; 3) freezing the submodule weights during the training process. In all the methods, the models weights are modified at one layer and one attention component at a time. The three methods are described in the following subsections, and the experiment results are in Section 5.

3.2.1 Convex Combination

Observing each submodule in a model demonstrate different robustness characteristics to parameter perturbation, Zhang et al [5] reported the $module\ criticality$ phenomenon. In another paper, Chatterji et al. [6] formalized the $criticality$ score metric and proved the metric is effective in revealing the network generalization. For each submodule *att*, they proposed to rewind submodule weights back to a convex combination, which is defined as $\theta^{\alpha}_{att} = (1-\alpha)\theta^{0}_{att} + \alpha \cdot \theta^{f}_{att}, \alpha \in [0,1]$, where θ^{0}_{att} denotes the initial weights and θ^{f}_{att} denotes the final weights.

The criticality of a submodule component *att* is definded as:

$$\text{Criti}_{att} = \min \alpha_{att} \text{ s.t. Loss} \left(\text{Model with } \theta^{f}_{att} \right) - \text{Loss} \left(\text{Model with } \theta^{rew}_{att} \right) < \epsilon \tag{1}$$

where θ^{f}_{att} denotes the final model and θ^{rew}_{att} denotes the weights after rewinding, and ϵ is a threshold.

The criticality score describes the minimum α that the weights can get to initialization while the performance drop is smaller than the threshold ϵ. More intuitively, a submodule can be said critical if the model performance drops significantly after being modified: a high criticality score indicates that the model rely heavily on the specific submodule.

3.2.2 Rewinding to Mid-stage Weights

The previous method inspire us to rewind the submodule weights of the final model back to any model weights during the training. Suppose the model is trained to convergence after n epochs, then we rewind $\theta^{f}_{att} = \theta^{i}_{att}, i \in [0, n]$ where θ^{i}_{att} is the model weight at the training epoch i.

Though we cannot calculate criticality score, we can observe the submodule importance during the training.

3.3 Freezing submodules during the training

Different from previous methods that part of model weights (submodules) are modified at evaluation, another method is to freeze the submodule weights during the training process by setting gradients to zeros.

In the method, the frozen model training curves are compared with the baseline model trained normally.

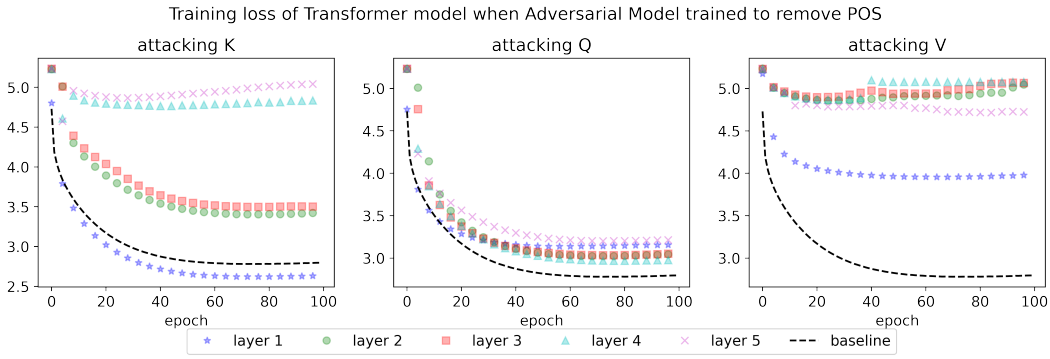

Figure 3: Training loss of the transformer model when using the adversarial model to attack POS information in K,Q,V representations at different layers

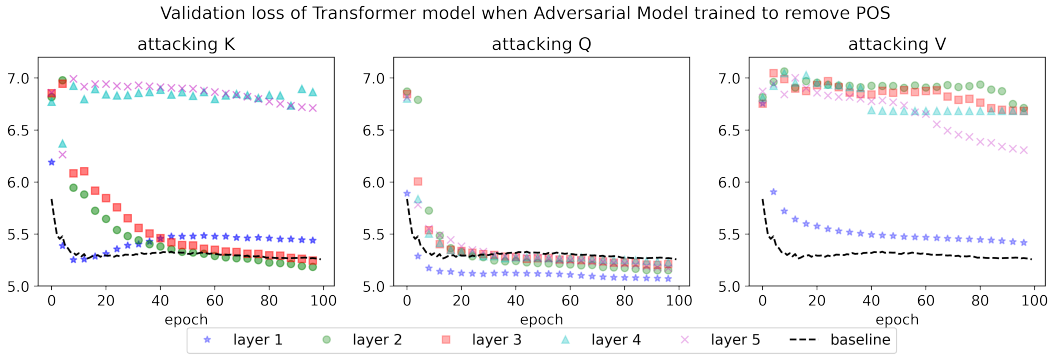

Figure 4: Validation loss of the transformer model when using the adversarial model to attack POS information in K,Q,V representations at different layers

4 Results and discussion of Adversarial Attacking Transformer submodules

4.1 Results

Figure 3 is the training loss of the transformer when an adversarial network is used to attack the POS information in K, Q, V representations at layer 1 through layer 5, and a baseline transformer loss when no adversarial network is used. We find that when POS in K representations in the earliest layer is being attacked, the train loss of the transformer model is improved over the baseline model. However, if we look at validation loss as in figure 4, the validation loss of such attack increases over the baseline model. Therefore attacking POS in K from the earliest layer causes the model to overfit. Attacking POS in K from middle layers increases the transformer model train loss but result in a lower than baseline validation loss, making the model more robust. Attacking POS of K in late layers makes both the train and validation loss close to initialization.

When POS in Q representations is attacked, the train loss of attacking all five layers cluster closely above the baseline training loss. The better than baseline validation loss further shows that such attacks make the model more robust

When POS in V are attacked in middle and late layers, we observe close to initialization train and validation loss. Attacking POS in V from the earliest layer leads to a moderately worse train loss but very close to baseline validation loss.

In figure 5 is the training loss of the transformer model when DEP information is being attacked. When attacking DEP in K, the train loss is higher at all layers. For middle to late layers, the train loss is close to initialization, but for early layers, the train loss is still reasonably close to the baseline.

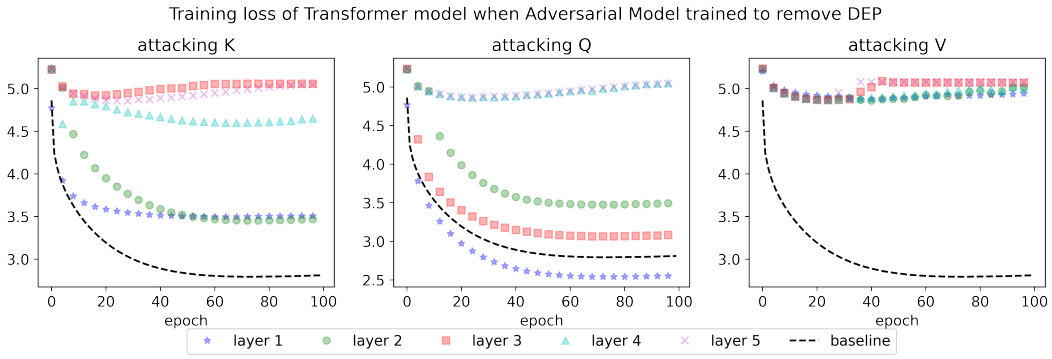

Figure 5: Training loss of the transformer model when using the adversarial model to attack DEP information in K,Q,V representations at different layers

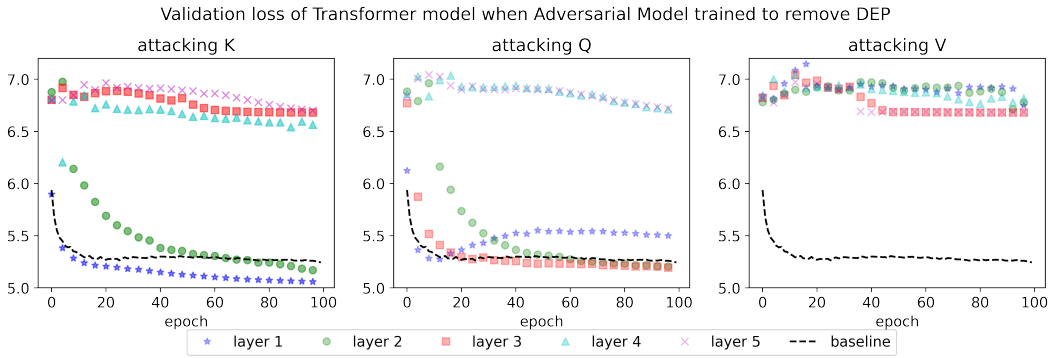

Figure 6: Validation loss of the transformer model when using the adversarial model to attack DEP information in K,Q,V representations at different layers

Looking at the validation loss, the attacking DEP in K of early layers produces better than baseline validation loss. When attacking DEP information in Q, we see a very similar train loss pattern compared to attacking POS in K. Attacking DEP in V at all layers makes the model not learn anything, because both the train loss and validation loss are just initialization values.

4.2 Discussion

We can try to explain the findings using certain interpretations of the structure of the transformer model in terms of the positioning of the layers and the structure that produce the K, Q, V representations. Since the model is taking part of sentences to predict a distribution over possible next words, we can think of earlier layers functions to transfer words to representations, and later layers function to transfer representations to probability distributions. Then, for K, Q, V, there is some subtlety to how they are used. As shown at the beginning of this paper, to calculate the attention score, K and Q are first multiplied, and then V is multiplied to that product.

Attacking the POS or DEP information in K representations of late layers can keep the train loss up. This means that when the transformer is trying to generate a probabilistic prediction for the next words, it is very important for it to have POS/DEP information encoded in K representations. Second, attacking early layers can cause the model to overfit. This means that encoding linguistic information can help the transformer model not to overfit. On the contrary, attacking K and Q representations from middle layers can make the transformer model more robust. This shows that linguistic information could be a burden for the transformer at the middle layers.

Another perspective we want to gain from the experiments is the flow of information within the transformer model. We care about this because the information is generally transferable, the model could recover certain missing information in part of the model from other parts of the model. Looking at the training plots, we see attacking V representations in almost all layers can make the transformer model not learn on the next word prediction task. This might be because that when calculating attention scores, the V representations are the last values to be included in the calculation. So if V representations are being attacked, the model can not recover the calculation of proper attention scores that allows the transformer model to learn. When attacking Q or K, we can see from train loss that most of the time the model is able to recover from the attack to some extend. We can also see that there are some similarities between the K loss curves and Q loss curves for removing both linguistic information. This should be addressed by the form of the scaled dot product attention, where K and Q are not really distinguishable in the calculation of the attention score 1.

5 Results of Rewinding and Freezing Method

In the rewinding experiments, we will first describe the training and validation results of the original task and then investigate three methods: 1) rewinding with convex combination; 2) rewinding on mid-stage models; 3) freezing at the training stage. In each of the three methods, we modify the weights of one submodule at a layer at a time.

5.1 Observing Results

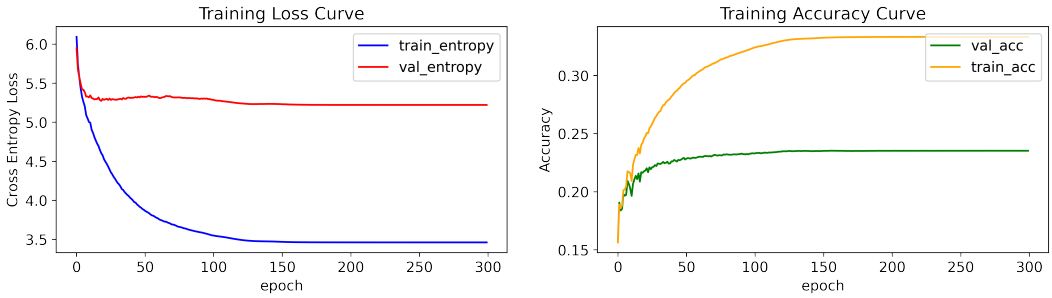

Figure 7: Training and Validation Curve of the Original Task

We trained a model to convergence and the training curve is shown in Figure 7. The training loss and accuracy are 3.462 and 33.32% respectively at convergence, and the validation loss and accuracy are 5.222 and 23.53% at convergence. In the following experiments we set the model at epoch 200 as our final model and its weight is denoted as θ^f_{att} and the initial untrained model as θ^0_{att}. This model is considered the baseline model.

5.1.1 Rewinding method 1: Convex Combination

In the first rewinding method, we rewind the weights of a specific submodule of a Transformer layer back to the convex combination: $\theta^\alpha_{att} = (1-\alpha)\theta^0_{att} + \alpha \cdot \theta^f_{att}$, where $\alpha \in [0, 1]$, with granularity 0.01.

Figure 8 shows the evaluation performance of convex combination on the training set. The color map describes the loss change for different level of dependence on the initialization. The y-axis shows the α value in the convex combination and the smaller α stands for a greater weight on the initial state. The x-axis shows the submodule that we modified. We observe that when we rewind the weights of submodule v at layer 2,3,4,and 5, there are apparent drops while others only have slight dips in performance.

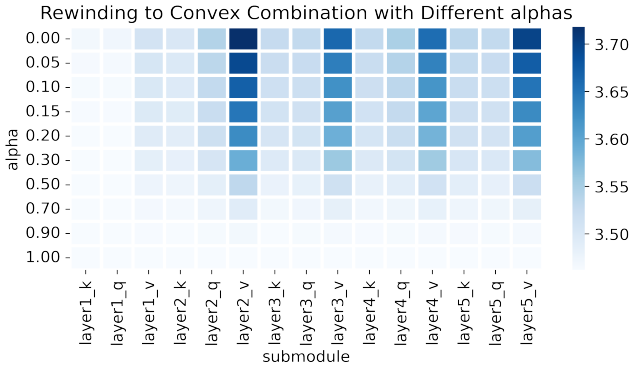

Figure 8: Rewinding Loss of convex combinations

Then, we use the metric *criticality* in Equation 1 to quantify the submodule importance. We set the threshold to be 1% of the final loss, which means we will test how much we may rewind the weight back to initial state while a performance tabulation is within $\epsilon = 0.0346$.

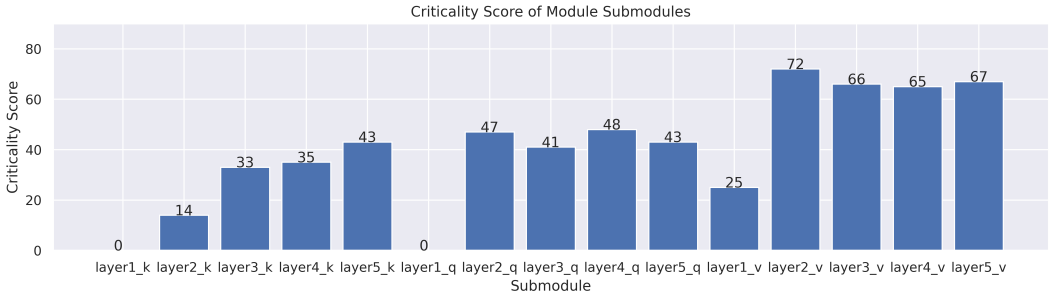

Figure 9: Criticality Score

Figure 9 shows the *criticality* score of each submodule. In the last two figures, we observe that

- Layer 1 has smaller criticality scores among all layers
- The models with v edited have a more significant performance drop while the models with k and q edited have almost the same amount of performance dip.
- Upper encoder-attention layers are more critical than lower encoder-attention layers

5.1.2 Rewinding method 2: Rewind the performance to the mid-stage

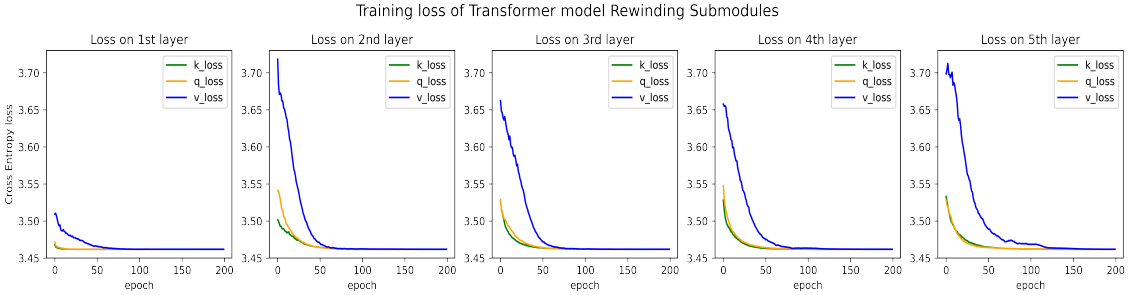

Figure 10: Rewinding Submodules

In this method, we edit the submodule weights by replacing the final model weights of a submodule with the mid-stage weights of that submodule: $\theta^f_{att} = \theta^i_{att}, i \in [1, 200], att \in \{k, q, v\}$ where i denotes the epoch and att denotes the attention component we edit on.

Figure 10 shows the loss curve if we rewind the model weights back to mid-stage weights. We observe that:

- Rewinding weights on V brings higher performance change while modifying K and Q has no huge difference
- Layer 1 is the most robust encoder layer against rewinding

5.1.3 Rewinding method 3: Freezing submodules during the training

In this method, we no longer edit the model weights during the evaluation but freeze some submodule weights throughout the training. Our goal is to see whether freezing submodules affacts the model training process.

We freeze the submodules by setting the gradients to be zero during the training and Figure 11 and Figure 12 shows the training curves and validation curves respectively.

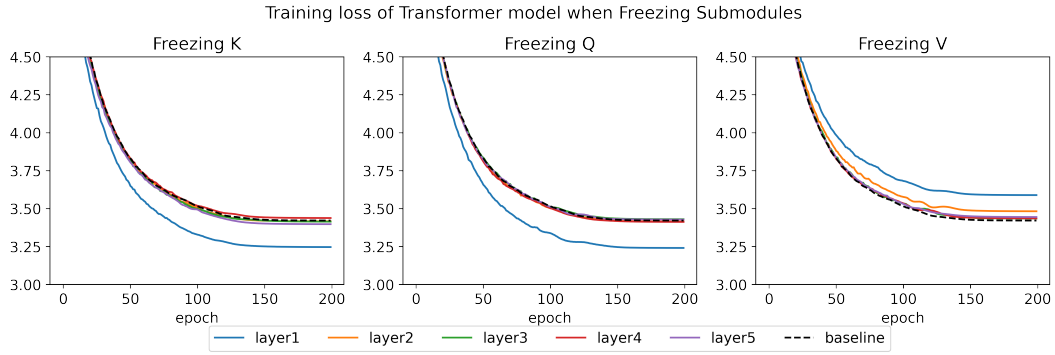

Figure 11: Performance of Freezing Submodules on Train Set

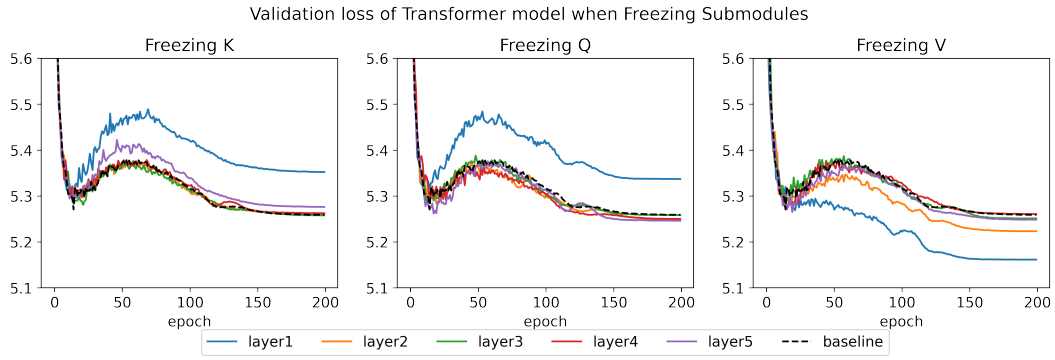

Figure 12: Performance of Freezing Submodules on Valid Set

We compare the training curve of the baseline model (in dashed black line) and the model with frozen submodules. There are some observations:

- In general, the models with first layer edited have more significant performance changes. Freezing K and Q submodules in layer 1 drops the training curve but increases the V validation loss.
- Freezing layer 1 causes over-fitting on K and Q but freezing V causes underfitting
- Freezing upper layer submodules has little impact on the results

5.2 Discussion

In this part we investigated the submodule importance in Transformer through three experiments and the results show that each submodules are not equally important in different settings.

Self attention components of K, Q, and V in layer 1 are important during the training since it causes overfitting if freezing K and Q while underftting if freezing V. On the contrary, they are the least important during the evaluation since they have very low criticality scores in the rewinding experiment. The overlapped training curves show that attention components in the upper layers are not important since information can be recovered through training, even they are frozen.

We also agree that V is more important in the attention dot product as for its high criticality scores, which accords with the adversarial experiment results.

6 Conclusion

In this work we investigated the importance and functionalities of individual submodule component in Transformer model. The experiment results reveal that some submodules in specific layers are more important in the next word prediction task. The V component is more vulnerable under adversarial attacking and has higher criticality scores. Also, attacking or freezing submodules in lower layers can cause performance perturbation and over-fitting in component K and Q but under-fitting in V. Both methods have shown that removing information from K and Q can make the model have less validation loss, therefore more robust. Lastly, the Transformer model shows that it is able to recover from information lost through attacking or freezing in most of the layers and submodules.

7 Future Work

To attain more generalizable results, the same experiment should be done on more different linguistic information. To have more convincing result, it is better to run the attack on each layer and attributes (K,Q,V) multiple times to generate a confidence interval. Because the convergence of the training might change as per observations during the experiment.

It would also be interesting to run the experiments on attacking or freezing combinations of K,Q,V or other parts of the transformer model (feed-forward part, decoder attentions, different attention heads, etc). If time commits, it would be also interesting to inhabit both K and Q simultaneously, or test the hypothesis if they are able to compensate for each other. Similarly, we may also inhabit a whole module or inhabit the attention component (one of K, Q, and V) in all the layers.

8 Acknowledgements and Lessons learned

First of all, we would like to sincerely thank Naomi Saphra for guidance, inspiration and help made in this project as an advisor. This is an interesting and fantastic project we have ever done since it's very close to NLP research. We also met some difficulties understanding previous research works in this area, determining the project direction, and understanding the Transformer details. Unlike applying NLP models to some specific tasks, a research-oriented project does not guarantee any meaningful results. Fortunately, by conducting literature review and many experiments, we finally discover some interesting and surprising properties of Transformer models.

9 Contribution

In this Capstone project Ziyi (ZX) contributes mostly to the adversarial experiment and Letian (LJ) contributes mostly to the rewinding&freezing experiment.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, & Illia Polosukhin. (2017). Attention Is All You Need.

[2] Logan, S. (2019). Using Knowledge Graphs for Fact-Aware Language Modeling. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics.

[3] Yaroslav Ganin, & Victor Lempitsky. (2015). Unsupervised Domain Adaptation by Backpropagation.

[4] Stephen Merity, Caiming Xiong, James Bradbury, & Richard Socher. (2016). Pointer Sentinel Mixture Models.

[5] Zhang, C., Bengio, S., & Singer, Y. (2019). Are all layers created equal?. arXiv preprint arXiv:1902.01996.

[6] Chatterji, N. S., Neyshabur, B., Sedghi, H. (2019). The intriguing role of module criticality in the generalization of deep networks. arXiv preprint arXiv:1912.00528.