Tweet Sentiment Analysis

Rahul Rade[1], Juan Diaz Sada[2], Radwa Sherif Abdelbar[2], Lam Nguyen Thiet[2]
[1]D-ITET, ETH Zurich, Switzerland, [2]D-INFK, ETH Zurich, Switzerland

***Abstract*—This aim of this paper is to employ deep learning techniques to perform sentiment analysis on tweets. We explore different RNN/CNN-based architectures as well as transformer-based models with our proposed set of fine-tuning techniques. Our best model achieves a classification accuracy of 91.88% on the Kaggle Public LB[1] while ensemble ranks 92.04%. Finally, we provide an in-depth discussion of existing research questions on the inner workings of RoBERTa.**

I. Introduction

In the past decade, *Twitter* has risen to enormous popularity world wide, as has its influence in areas like business, politics, and society. Tweet sentiment analysis is therefore an area of large interest in multiple domains. Examples of the applications of tweet sentiment analysis are varied - market analysts might be interested in public reception of specific products, presidential campaigns would want to analyse public sentiment for a candidate, among many others. A specific challenge posed by tweet sentiment analysis is due to the nature of tweets. Tweets are limited to 280 characters and therefore people frequently use abbreviations in order to meet this limit, users also tend to use hashtags and informal slang. This makes the task of sentiment classification increasingly complex. Thus, it becomes very critical to form a suitable featurized representation of the text for capturing the intent, sarcasm and polarity of the tweet. This has become somewhat possible with the advent of modern deep learning algorithms.

In this paper, we explore previous sentiment analysis research and apply it to build deep learning models to classify tweets into positive or negative categories. Firstly, we explore simple RNN-CNN based models, investigating the effects of pre-trained word embedding and architecture choices on the performance of the model. Secondly, we consider various pre-trained transformer-based models and propose several techniques which can be used to effectively fine-tune such models to the target task while tackling the problems of generalization, overfitting, catastrophic forgetting, noisy true labels and computation complexity. Finally, we perform a series of experiments to investigate the factors which contribute to the performance of transformers, in particular RoBERTa[2] for NLP tasks. We also conduct ablation studies to discern the effect of individual attention heads on the overall model performance and the patterns captured by attention heads. We reproduce certain experiments from [3] to understand this. Our key contributions[1] summarized below:

- Exploratory analysis of RNN/CNN-based models with pretrained embeddings.
- We propose a set of training strategies to fine-tune the pre-trained transformers for any target task.
- Our ensemble achieves a classification accuracy of around 92.04% on Kaggle Public LB.
- We perform an in-depth analysis of behaviour of different attention heads and layers in RoBERTa transformer.

II. Background

In this section, we quickly review the existing models which we have used as an integral component of our experiments.

A. Long Short-Term Memory

LSTM is an RNN architecture, introduced in [4] as a solution to the vanishing gradient problem, which hinders the ability of RNNs to learn long-term dependencies in sequential data. In addition to the hidden state h_t, the LSTM has a cell state c_t which is responsible for propagating long-term information through the model. The model uses "gates" to make decisions about preserving, discarding or updating information from the cell and hidden states. Further details can be found in Appendix A.

B. Transformer Models

We quickly review the transformers which we have used as a baseline for evaluating our proposed fine-tuning strategies and experiments. We omit an exhaustive description of the models as they are quite similar to the ubiquitous de-facto architecture proposed by [5].

1) BERT: BERT[6] is a simple but effective model architecture which employs a deep multi-layer bidirectional transformer to learn deep contextual representations from text. The transformer architecture uses L layers with each layer consisting of a multi-head attention mechanism with A heads and a small feed-forward neural network with hidden dimensions H. The main highlight of the pre-training scheme used for BERT is the masked language modelling (MLM) objective which allows large-scale bidirectional training of transformers. The model is first pre-trained on a large unlabeled text corpus using the MLM and next sentence prediction (NSP) objectives. Finally, we fine-tune the architecture on the downstream sentiment classification task.

For sentiment classification task, the input to BERT consists of a sequence of tokens, $x_1, \ldots, x_N$, where the first token is a special classification token [CLS]. The final hidden representation corresponding to [CLS] token is fed to a subsequent fully connected network for classification tasks.

2) Other Models: We also conduct experiments with RoBERTa[2], ELECTRA[7], XLNet[8] and BERTweet[9]. We refer curious readers to the original papers for details.

[1]Our code is available here. Kaggle team: **cil_student**

III. Models and Methods

In this section, we give a brief overview of our RNN-CNN based models. We also describe the proposed set of fine-tuning strategies for transformers which we will investigate experimentally in the following sections.

A. Baseline Models

The following models were used as baselines:

1) GloVe + Random Forest Classifier: This baseline is extended from the CIL Lecture at ETH Zurich. We compute the GloVe embedding of each word. A sentence of length L is represented by a vector $s = \frac{1}{L}\sum_{i=1}^{L} v_i$ which is fed into a Random Forest Classifier.

2) Stanford GloVe + Random Forest Classifier: We improve the previous baseline by using pre-trained GloVe embeddings from Stanford University. These embeddings were computed on a data set containing 2 billion tweets.

3) Stanford GloVe + Simple RNN: Our third baseline is a classic RNN model, which is particularly suitable as a baseline for our RNN-based models in section III-B. Our previous baseline models do not keep track of word order, therefore all contextual information is lost. In this model, a sentence is represented by an $(L \times D)$ matrix where L is the sentence length and D is the size of the word vectors. This matrix is fed into an RNN, then a linear layer to output the classification.

B. RNN/CNN Models

In this section, we describe the part of our experiment which employs RNN-based or CNN-based models. We will further compare the results of these models to the transformer models described in the following section.

1) Stanford GloVe + CNN : Similar to the baseline models, a sentence is represented by an $(L \times D)$ matrix, where $D = 100$. Each CNN layer has N kernels, each with size $(K_i \times D)$. This can be interpreted as follows: the i-th layer applies its kernels to a window of size K_i sentences at a time. The output of convolution has dimensions $(N, L - K_i + 1)$. It is then passed through a ReLU function and a max-pooling layer of window size $(1 \times L - K_i + 1)$, which finally yields an N-dimensional vector for each convolution layer in the model. The N-dimensional vectors are concatenated together and fed into a fully-connected layer which outputs the prediction of the model.

2) Character Embeddings + CNN: For an alphabet of size V, each character is represented by a V-dimensonal one-hot encoded vector. Each sentence of character-length C is represented by a $V \times C$ matrix. This model, which is adapted form [10], is different from the model in section 1 in that it uses 1-dimensional CNN. We stack 6 CNN layers on top of one another, each with N filters. The i-th CNN layer applies a 1-dimensional filter to K_i characters at a time. The first two layers and the last layer apply max-pooling to their outputs. We slightly adapt the model to feed the resulting N-dimensional vector into one fully connected layer, instead of three as described in the paper. This yields a more robust model.

This models is perhaps the least interpretable in our models since it captures regional features related to the character composition of the tweet, without being aware of the complex semantics of sentences.

3) Stanford GloVe 100d + 2 Layer BiDirectional LSTM: The sentence representation is similar to the word CNN model. The $(L \times D)$ are passed through 2 stacked bidirectional LSTM layers with hidden dimension of size 256. The final forward and backward hidden vectors are concatenated together and then fed into a fully connected layer which outputs the prediction. We also experimented with adding a max-pool layer to the output of a fully connected layer with its inputs being a concatenation of the bi-directional hidden layers(which represent contextual embeddings in both directions) and the original GloVe embeddings, similar to the model proposed in [11]. However, we found that this only achieved a negligible improvement over the simpler model. We further improved the model by adding an attention mechanism, this achieved an increase to the model's accuracy and yielded our best performing RNN-based model.

C. Fine-tuning Strategies for Transformer Models

We provide a set of fine-tuning strategies for adapting the pre-trained transformers for sentiment classification task. We specifically experiment with pre-trained RoBERTa$_{\text{BASE}}$[2].

1) Simple Finetuning (FT): In order to adapt transformers to the target sentiment classification task, we simply fine-tune the network for 3 epochs with certain heuristics. While performing inductive transfer across tasks, a general intuition is that lower layers extract more general information while higher layers extract task-specific information. So, we use a decaying learning rate for different parameters groups. In particular, we use a high learning rate for final fully connected classification layers, a moderate one for transformer layers and a very low learning rate for the embedding layer. This process is similar to discriminative fine-tuning as proposed in [12]. This helps to prevent the problem of catastrophic forgetting and improves generalization. We use a batch size of 64 and learning rates $\in$ {5e-6, 1e-5, 2e-5, 5e-5}, with Adam optimizer [13][14] with a linear warmup[12] for the first $K\%$ (K =6, 8, or 10) of steps followed by a linear decay to 0.

2) Multi-task Finetuning (MT): Multi-task Learning[15] is a methodology where a single network is training on multiple tasks simultaneously with the motivation that the knowledge obtained from one task would constructively benefit the other. In multitask fine-tuning, we add an auxiliary language modelling (LM) task which provides a robust supervision to the target classification task. We weigh the LM cross entropy loss by a parameter λ as presented below:

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda * \mathcal{L}_{lm} \qquad (1)$$

All the parameters of the base transformer are shared across both tasks. We only use separate fully connected heads for LM and classification tasks.

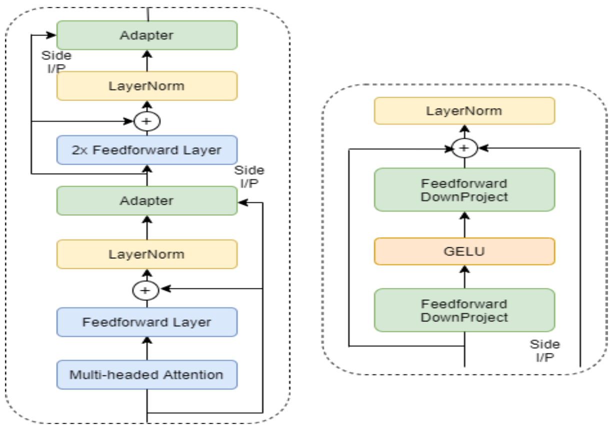


Fig. 1: *Left*: RoBERTa + Adapters, *Right*: Adapter Module

3) Training with Teacher Soft Labels (STL): Label smoothing is often used as a technique for enhancing the performance of the model. It prevents the model from becoming overconfident about labels and improves generalization[16]. In practice, labels are usually smoothed by an appropriate factor. However, instead of smoothing all the labels by the same factor, we use labels obtained from a fine-tuned transformer to compute the smooth labels. This helps to alleviate the effect of noisy or wrong labels in the training set. The resulting training objective is given by:

$$\mathcal{L} = \sum_i p_i * \log(\hat{p}_i) \tag{2}$$

where, $p_i = (t_i + y_i)$, t_i is probability estimated by teacher model, y_i is the ground truth label and $\hat{p}_i$ is probability estimated by model which is being trained. This objective provides a rich signal for training the network by fully leveraging the full teacher distribution.

4) Parameter Efficient Fine-tuning with Adapters: Simple fine-tuning is highly parameter inefficient when there are many downstream tasks with the process of fine-tuning leading to an entirely different model for each task. This increases the amount of parameters to be stored and shared. Instead, we explore an efficient fine-tuning approach based on adapter modules proposed in [17]. Adapters are feed-forward networks inserted between layers of the base transformer network. The adapters themselves have a very small number of parameters. This results in a large reduction in the number of parameters which need to be adapted for each downstream task, without significantly compromising the performance on the tasks.

We use the bottleneck adapter modules proposed in [17] with a slight modification in their placement in the base architecture as shown in Figure 1. Also, we incorporate an additional layer normalization with skip connection before the adapters. This improves performance and stability of the training process. The adapter modules are initialized with identity at the start of training and are the only trainable parameters in the network. These modules learn small residuals, thus serving the purpose of making modifications to the activations of the respective layer with the goal of improving the performance.

Model	**Val. Acc.**	**Test Acc.**	
		Public	**Private**
CIL GloVe + RF[a]	65.89	59.28	NA
Stanford GloVe + RF[a]	75.04	72.24	NA
Stanford GloVe + Simple RNN[a]	83.37	83.340	NA
CharCNN[a]	79.0	78.18	NA
WordCNN[a]	86.56	86.58	NA
LSTM[a]	88.12	87.4	NA
BERT$_{\text{BASE}}$	90.35	89.30	NA
RoBERTa$_{\text{BASE}}$	90.91	90.54	NA
XLM-RoBERTa$_{\text{BASE}}$	90.39	89.52	NA
XLNet$_{\text{BASE}}$	90.72	89.88	NA
ELECTRA$_{\text{BASE}}$	91.01	90.01	NA
BERTweet	92.06	91.16	NA
RoBERTa$_{\text{LARGE}}$	91.31	90.96	NA
XLNet$_{\text{LARGE}}$	91.06	89.66	NA
RoBERTa$_{\text{BASE}}$+MT	91.16	90.30	NA
RoBERTa$_{\text{BASE}}$+STL$_{\text{BERTweet}}$	91.11	90.94	NA
RoBERTa$_{\text{BASE}}$+Adapter	89.19	88.76	NA
ELECTRA$_{\text{LARGE}}$+STL$_{\text{BERTweet}}$	91.58	91.24	NA
BERTweet+STL$_{\text{BERTweet}}$	92.06	91.38	NA
BERTweet+STL$_{\text{BERTweet}}$+SWA[18]	**92.19**	91.26	NA
BERTweet+STL$_{\text{BERTweet}}$[b]	92.00	91.62	NA
BERTweet+STL$_{\text{ELETCTRA}_{\text{L}}\text{+STL}}$[b]	**92.12**	**91.88**	NA
Transformer Ensemble	-	**92.04**	NA

TABLE I: Performance scores on validation and test sets. [a]A different validation set is used than that for transformers. [b]Hyperparameters are tuned compared to previous models.

IV. Results

Table I presents our obtained scores on the evaluation and test sets. Our LSTM model outperformed the baseline RNN model by approximately 6%. We observe that all the pre-trained transformer models consistently outperform RNN-CNN architectures. Among the models trained via simple fine-tuning strategy, BERTweet significantly outperforms even the large version of other models. BERTweet benefits from being pre-trained on a huge corpus of 850M tweets[9]. Our experiments comparing different fine-tuning strategies suggest that there is no significant benefit obtained from multi-task training. Nonetheless, there is a large boost obtained by training with teacher soft labels with an improvement of upto 0.5%. RoBERTa$_{\text{BASE}}$ when fine-tuned with soft labels obtains a score comparable to vanilla RoBERTa$_{\text{LARGE}}$. An interesting observation is that this approach also improves the performance of BERTweet itself. Moreover, it helps in model generalization to the test set. The last fine-tuning scheme based on adapters performs marginally worse than simple fine-tuning on RoBERTa. We try different bottleneck layer sizes $\in \{32, 64, 128\}$ & select the best performing one. This carries the advantages of being highly parameter and computation efficient with much less parameters to be updated during fine-tuning. Finally, we compute a linear ensemble of the predictions made by transformers to achieve the best performance of 92.04%. A linear ensemble leverages the advantage of diversity of models to achieve a more robust score.

V. Discussion

We try to answer the following open-ended questions regarding the inner workings of BERT as in [3]. We conduct all the experiments using RoBERTa$_{\text{BASE}}$[2] with naive fine-tuning.

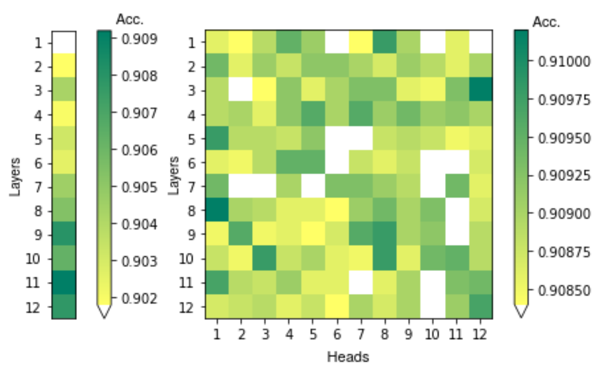


Fig. 2: *Left*: Effect of Masking Layers and *Right*: Effect of Pruning Heads

- How does pruning attention heads and disabling entire layers of RoBERTa affect its performance?
- What kind of information - syntactic, semantic, contextual or some arbitrary patterns, do the heads focus on?
- Effect of fine-tuning? What kind of examples lead to failure of the model to comprehend the implicit sentiment?

Firstly, we evaluate the contribution of each attention head and layer to the overall performance. We run multiple experiments by disabling a particular layer of the model. We disable a particular layer by simply replacing it with an identity layer. No significant decrease in performance is observed as presented in Figure 2. Instead some configurations like disabling layer 11 surprisingly benefit the performance while masking earlier layers detrimentally affects performance due the fact that crucial bi, tri-gram interactions and word-level contexts are critically captured by these layers. To probe the influence of individual heads, we perform ablation by pruning each head and observe its effect. Disabling certain performance critical heads result in a drop in accuracy (see Figure 2). However, as in the case of layers, pruning some heads leads to a performance gain of upto 0.1-0.2%. This suggests that RoBERTa is massively over-parameterized and there is a large redundancy in the patterns captured. The use of dropout in the original architecture may be a cause for this.

We manually probe the attention maps for different layers of RoBERTa. We find that there is a general trend of heads attending to different tokens. Specifically, we observe a few patterns such as: vertical, near-diagonal and hybrid as illustrated in Figure 3. Vertical patterns (marked red) correspond to attention heads focusing on a specific token. We see that a significant number of heads focus on the special tokens <s>and </s>. As noted in [19], these patterns indicate a sort of "no-op" wherein the attention layer hardly affects the output. The near-diagonal attentions (black) focus on capturing local contextual information for each word and disabling such critical heads drops the performance as seen in Figure 2. Such patterns are more evident in the lower layers of the network suggesting a hierarchical aggregation of features. The final category of attention maps is composed of complex patterns such as a combination of previous two types, very broad attentions and

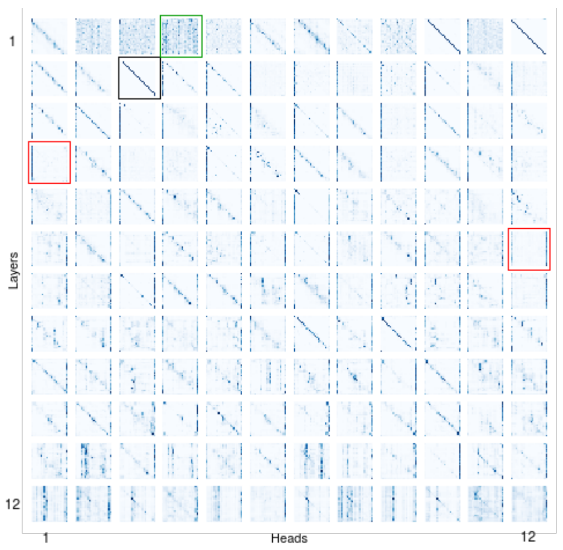
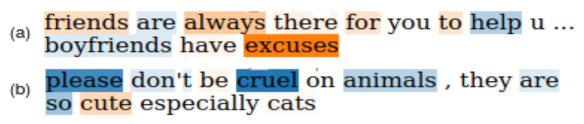


Fig. 3: Attention maps for all layers of RoBERTa

those which specifically relate to some syntactic grammar rules. The latter ones capture certain essential linguistic properties such as synonyms, coreferent-antecedent relations, noun modifiers and so on. These observations are consistent with the results obtained in [19] and we refer readers to [19] for details.

Lastly, we try to understand the effect of fine-tuning. We measure the similarity between the model layer weights before and after fine-tuning in terms of normalized Frobenius norm. We see that the last few layers encode highly task-specific features and are changed the most. However, even after fine-tuning, RoBERTa occasionally fails to comprehend implicit sarcasm and informal slang present in some of the tweets. Instead, the attention mechanism sometimes focuses on words expressing strong positive or negative sentiment and bases its decision on those words. Figure 4 illustrates two such samples.


(a) friends are always there for you to help u ... boyfriends have excuses

(b) please don't be cruel on animals , they are so cute especially cats

Fig. 4: Examples on which RoBERTa fails. Explanations are obtained using LIME[20]. Orange indicates +ve sentiment and blue is -ve. True label for (a) is -ve and for (b) is +ve.

VI. Summary

We presented various approaches to the task of tweet sentiment analysis. Firstly, an analysis of RNN/CNN based models and their performance improvements over simpler baselines. Then, we performed an exhaustive comparison of existing pre-trained transformers and provided a set of strategies for fine-tuning transformers to the target task. We find that BERTweet significantly outperforms all other models. Further, fine-tuning with teacher soft labels results in a significant performance boost. Finally, we looked into some ways to interpret the inner workings of RoBERTa.

References

[1] ETHZ CIL Text Classification 2020 (Project 2). [Online]. Available: https://www.kaggle.com/c/cil-text-classification-2020/leaderboard

[2] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[3] O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky, "Revealing the dark secrets of BERT," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4365–4374. [Online]. Available: https://www.aclweb.org/anthology/D19-1445

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[7] K. Clark, M. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: pre-training text encoders as discriminators rather than generators," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=r1xMH1BtvB

[8] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," 2019.

[9] T. V. Dat Quoc Nguyen and A. T. Nguyen, "BERTweet: A pre-trained language model for English Tweets," *arXiv preprint*, vol. arXiv:2005.10200, 2020.

[10] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[11] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *AAAI*, 2015.

[12] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2018.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[14] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[15] S. Ruder, "An overview of multi-task learning in deep neural networks," 2017.

[16] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 4694–4703. [Online]. Available: http://papers.nips.cc/paper/8717-when-does-label-smoothing-help.pdf

[17] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[18] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," 2018, cite arxiv:1803.05407Comment: Appears at the Conference on Uncertainty in Artificial Intelligence (UAI), 2018. [Online]. Available: http://arxiv.org/abs/1803.05407

[19] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does bert look at? an analysis of bert's attention," in *BlackBoxNLP@ACL*, 2019.

[20] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016, pp. 1135–1144.

Appendix

A. LSTM Details

At time step t, the cell and hidden states are updated as follows ($\star$ denotes point-wise multiplication):

- The forget gate: $f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$.
- The input gate: $i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$.
- A candidate update for the cell state: $\tilde{c}_t = tanh(W_c h_{t-1} + U_c x_t + b_c)$.
- The cell state update: $c_t = f_t \star c_{t-1} + i_t \star \tilde{c}_t$. f_t masks out the values of the previous hidden state that should be discarded, while i_t decides the values that should be updated.
- The output gate $o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$.
- The hidden state update: $h_t = o_t \star tanh(c_t)$: the hidden state is equal to the cell state masked by the output gate o_t.

B. Additional Plots

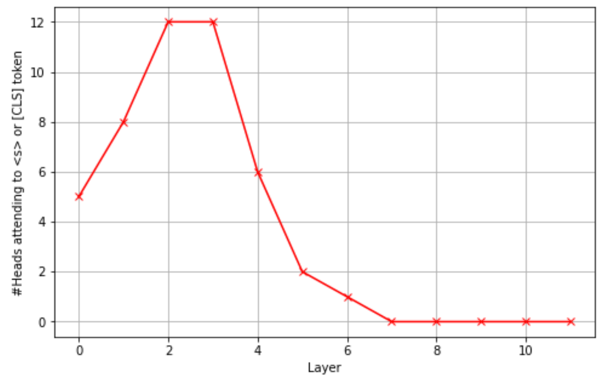


Fig. 5: Number of heads per layer attending to <s>token. We consider a head to be attending to <s> if more than 75% of its tokens attend to <s>.