Name: Akarshan kumar

Email address: akarshan.1711@gmail.com

Contact number: 7903299199

Anydesk address: **114 999 665**

Years of Work Experience: **0**

Date: **10th APR 2021**

Self Case Study -2: Understanding How BERT Learns to Identify Edits

Overview

In the recent past, there have been attempts by conventional ML models to perform operations on sequence classification tasks(including but not limited to NLP). The conventional models achieve a decent performance score on the task, being highly interpretable at the same time. But after the emergence of BERT and TRANSFORMERS as a new epitome of models to handle sequence classification tasks, the previous performance scores are outperformed as well. Given all that, these new emergent models have a caveat. The prevalent issue with the new emergent models like BERT is their interpretability. Unlike previous models(like CNN, LSTM, SVM), Transformers-based models have hundreds of millions of parameters within them, making it infeasible to manually check for interpretability. The paper finds that "classification performance scores do not always correlate with interpretability". As Transformers-based models are Ubiquitous within NLP tasks, it is eminent now to search for ways to make them interpretable.

Given all the above context, BERT's self-attention mechanism has been tried to interpret it, but some bodies of work suggest it being insufficient on single-sequence classification tasks while some suggest in favor of attention. This shall be put to test in this paper. Paper trains three Bert variants (BERT, RoBERT, SciBERT) with fine-tuning approach, on existing sentence classification task, and compare the results with previous models as a baseline, and then uses methods present in Vashishth et al. (2019) and DeYoung et al. (2020) paper to evaluate BERT's interpretability for single-sequence classification tasks.

The key contributions of this paper are

1. Applying pre-trained Transformer models to a sequence classification task and demonstrating notable improvements.
2. Quantifying BERT's interpretability on said sequence classification task using attention.
3. Comparing the impacts of pre-training methods on interpretability.

Methodology

AESW task(Daudaravicius et al. (2016)) dataset is used to fine-tune the following three models. BERT is the first pre-trained transformer model. RoBERTa has the same architecture as BERT but is pre-trained with more data (13GB vs 160GB) and demonstrates improvements over BERT on most benchmarks. SciBERT also has the same architecture as BERT but is pre-trained on 1.14M papers from semanticscholar.org(academic papers).

This results in the following findings

1. Fine-tuning pre-trained transformer models teach them previously unknown patterns.

2. BERT's final attention layer is more interpretable than SciBERT's or RoBERTa's.

Some novel patterns are also found as BERT and RoBERT attend to contractions in sentences as 'need edit' without any preexisting inclination to attend to them(this might be the cause of fine-tuning).

Quantifying the interpretability

Paper finds that BERT's attention in the last layer more accurately identifies words relevant to predicting "need edit", despite lower F1 scores on the classification task. We hypothesize that SciBERT's lower interpretability in the final layer is due in part to its scientific writing pre-training. It could be encoding the representation of an incorrect sentence in earlier layers,

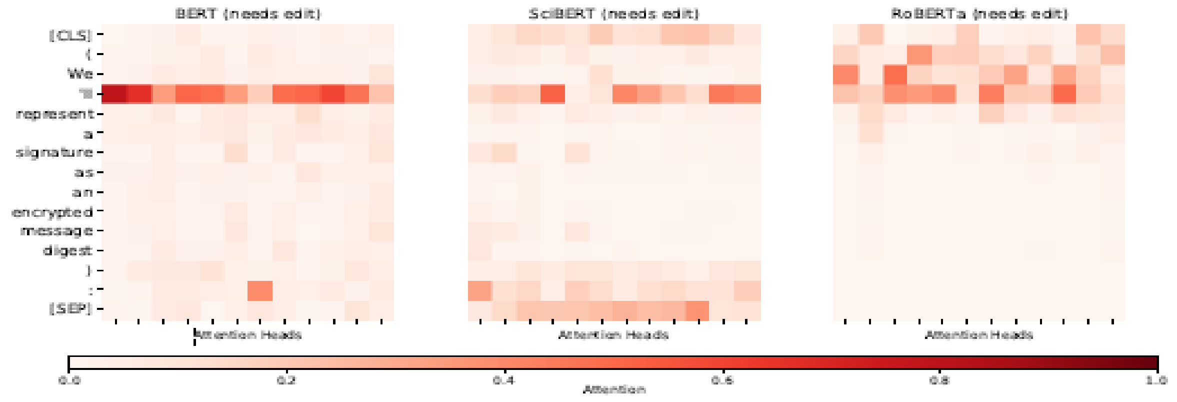


Figure 2: [CLS] attention maps for "(We~~'ll~~ **will** represent a signature as an encrypted message digest):" in the last layer. Notice that all three models attend heavily to 'll.

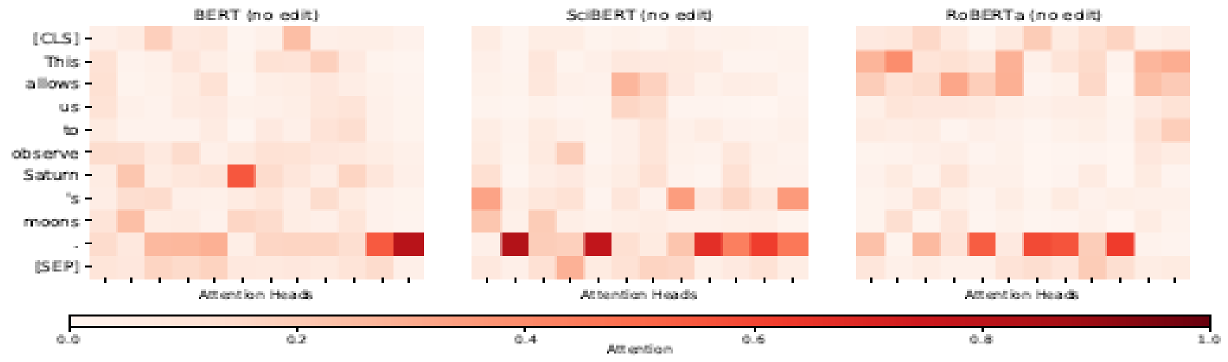


Figure 3: [CLS] attention maps for "This allows us to observe Saturn's moons" in the last layer. Notice that all three models predict "no edit" and do not attend to 's.

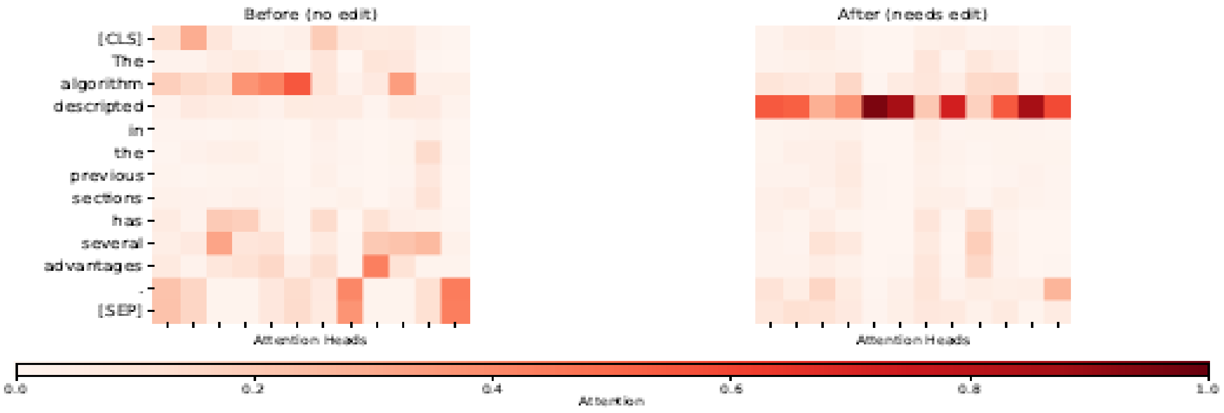
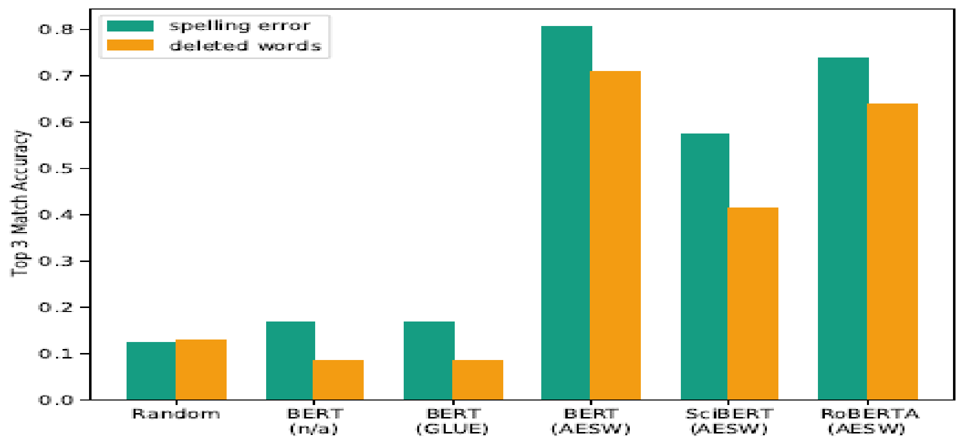


Figure 4: Comparison of BERT's [CLS] attention map for "The algorithm ~~descripted~~**described** in the previous sections has several advantages" before and after fine-tuning on the AESW task.

Conclusion

Furthermore, experiments are done to come up with the below graph for interpretability



Top 3 match accuracy being the top 3 relevant words having the most attention for editing prediction by BERT vs actual labels.

Research-Papers

1. http://jalammar.github.io/illustrated-transformer/

   https://arxiv.org/abs/1706.03762

   "The Illustrated Transformers", An explanation of " Attention is all you need"

   - In this blog, Jay alammar talks about the "Attention is all you need" paper where Transformers were first presented.
   - He breaks down the Transformer in all of its components and gives a wide idea about the architecture.
   - First, it is shown that transformers are nothing but a stacked set of encoders and decoders.
   - Each encoder is further broken down into a Self-attention module followed by a Feedforward network. A decoder also has those 2 layers but an Attention layer between them to focus on relevant parts of the input sentence.
   - Self-attention is explained in detail in the following section as a matrix multiplication of multiple vectors and softmax on top of that. Also, the mechanism of many self-attention heads working together is exposited in a fairly detailed manner.

- Furthermore positional encoding, residual connections, decoder side, the final layer, loss functions, and training method is discussed.

2. https://www.aclweb.org/anthology/W16-0506

   A Report on the Automatic Evaluation of Scientific Writing Shared Task

   - The AESW(Automatic Evaluation of Scientific Writing) is a work to identify sentences that need editing, to keep their scientific value intact in the formal English language.
   - The data set is a collection of text extracts from 9,919 published journal articles
   - This effort is made keeping in mind the automated support that can be extended to non-English speaking researchers that publish papers in English.
   - The data is acquired by the company VTeX, a professional editing company. Data has two aligned versions of the same text - before and after editing.
   - The main goals of the task were to

     – identify sentence-level features that are unique to scientific writing;

     – provide a common ground for development and comparison of sentence-level automated writing evaluation systems for scientific writing;

     – establish the state-of-the-art performance in the field.

   - The Task also defines some baseline models to test the viability of data while setting the reference for state-of-the-art automatic evaluations. The data is shared with multiple participants and with the help of classical machine learning, it is classified for need edits or not. The Task is binary classification and two types of predictions are evaluated: Binary prediction and Probabilistic estimation.

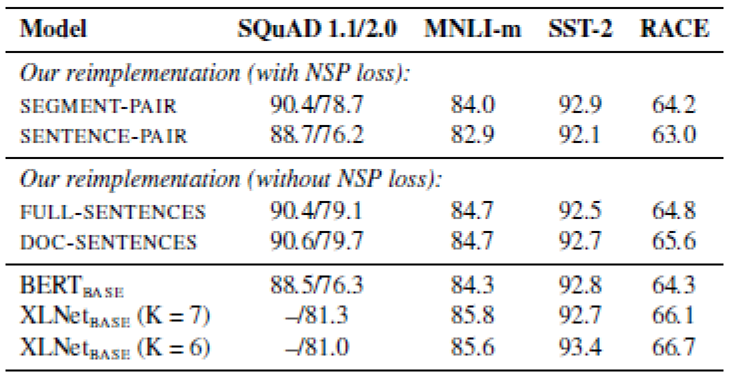
3. https://arxiv.org/pdf/1907.11692.pdf

   RoBERTa: A Robustly Optimized BERT Pretraining Approach

- The paper suggests that language pretraining has led to significant performance gains. BERT when published was heavily undertrained. With more finetuning, it could match or exceed the performance of every model that came after it.
- Paper modifies the training method in simple ways,

  (1)training the model longer, with bigger batches, over more data; (2) removing the next sentence prediction objective; (3) training on longer sequences; and (4) dynamically changing the masking pattern applied to the training data.

- This paper shows the importance of design choices and training strategies that lead to better downstream tasks. A new and novel dataset is used as it was confirmed using more training data enhances the BERT's performance, making it competitive with the following language models.
- Bert was trained mainly for MLM and NSP tasks with adam optimizers and a triangular learning rate callback was used giving the following results on the dataset.



Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ ($K = 7$)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ ($K = 6$)	–/81.0	85.6	93.4	66.7

Table 2: Development set results for base models pretrained over BOOKCORPUS and WIKIPEDIA. All models are trained for 1M steps with a batch size of 256 sequences. We report F1 for SQuAD and accuracy for MNLI-m, SST-2 and RACE. Reported results are medians over five random initializations (seeds). Results for $\text{BERT}_{\text{BASE}}$ and $\text{XLNet}_{\text{BASE}}$ are from Yang et al. (2019).

4. https://www.aclweb.org/anthology/D19-1371/

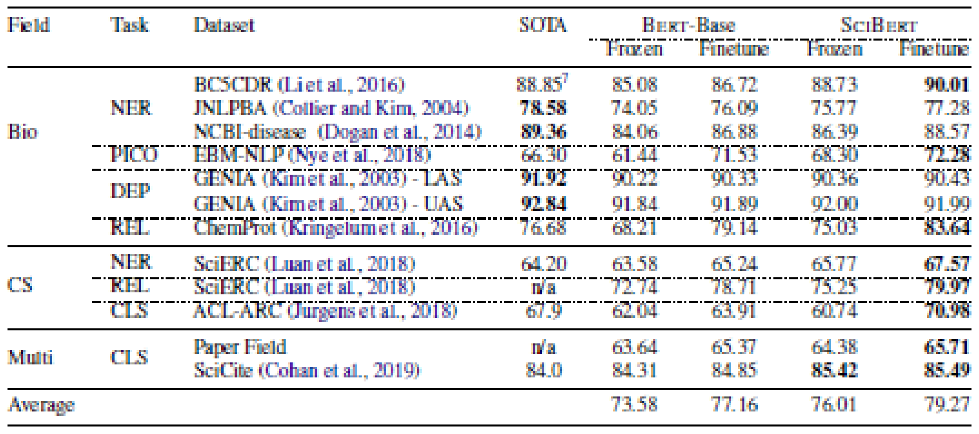
   SCIBERT: A Pre Trained Language Model for Scientific Text

   - SciBert was introduced to address the lack of high-quality, large-scale labeled scientific data. A voluminous increase in the publishing of scientific research papers in this decade has made NLP an essential tool for large-scale knowledge extraction.

- In this work, the paper makes the following contributions:

  (i)Release SCIBERT, a new resource demonstrated to improve performance on a range of NLP tasks in the scientific domain. SCIBERT is a pre-trained language model based on BERT but trained on a large corpus of scientific text. (ii) Performing extensive experimentation to investigate the performance of fine-tuning versus task-specific architectures atop frozen embeddings and the effect of having an in-domain vocabulary. (iii) We evaluate SCIBERT on a suite of tasks in the scientific domain and achieve new state-of-the-art (SOTA) results on many of these tasks.

- unsupervised pre-training of language models on large corpora significantly improves performance on many NLP tasks. Paper trains SCIBERT on a random sample of 1.14M papers from Semantic Scholar. All pre-trained BERT models are converted to be compatible with PyTorch using the PyTorch transformers library.
- Paper observes that SciBert outperformance Bert-Base on scientific tasks while achieving new SOTA results on many of these tasks using SciBert.



Field	Task	Dataset	SOTA	BERT-Base		SCIBERT	
				Frozen	Finetune	Frozen	Finetune
Bio	NER	BC5CDR (Li et al., 2016)	88.85[7]	85.08	86.72	88.73	**90.01**
		JNLPBA (Collier and Kim, 2004)	**78.58**	74.05	76.09	75.77	77.28
		NCBI-disease (Dogan et al., 2014)	**89.36**	84.06	86.88	86.39	88.57
	PICO	EBM-NLP (Nye et al., 2018)	66.30	61.44	71.53	68.30	**72.28**
	DEP	GENIA (Kim et al., 2003) - LAS	**91.92**	90.22	90.33	90.36	90.43
		GENIA (Kim et al., 2003) - UAS	**92.84**	91.84	91.89	92.00	91.99
	REL	ChemProt (Kringelum et al., 2016)	76.68	68.21	79.14	75.03	**83.64**
CS	NER	SciERC (Luan et al., 2018)	64.20	63.58	65.24	65.77	**67.57**
	REL	SciERC (Luan et al., 2018)	n/a	72.74	78.71	75.25	**79.97**
	CLS	ACL-ARC (Jurgens et al., 2018)	67.9	62.04	63.91	60.74	**70.98**
Multi	CLS	Paper Field	n/a	63.64	65.37	64.38	**65.71**
		SciCite (Cohan et al., 2019)	84.0	84.31	84.85	**85.42**	**85.49**
Average				73.58	77.16	76.01	79.27

Table 1: Test performances of all BERT variants on all tasks and datasets. **Bold** indicates the SOTA result (multiple results bolded if difference within 95% bootstrap confidence interval). Keeping with past work, we report macro F1 scores for NER (span-level), macro F1 scores for REL and CLS (sentence-level), and macro F1 for PICO (token-level), and micro F1 for ChemProt specifically. For DEP, we report labeled (LAS) and unlabeled (UAS) attachment scores (excluding punctuation) for the same model with hyperparameters tuned for LAS. All results are the average of multiple runs with different random seeds.

- Given the disjoint vocabularies (Section 2) and the magnitude of improvement over BERT-Base (Section 4), we suspect that while an in-domain vocabulary is helpful, SCIBERT benefits most from the scientific corpus pretraining.

5. https://arxiv.org/abs/1702.08608

Towards A Rigorous Science of Interpretable Machine Learning

- The paper states that measuring performance such as accuracy is rather simple than measuring interpretability as those performances can be quantified.
- A popular fallback is the criterion of interpretability can be: if the system can explain its reasoning, we then can verify whether that reasoning is sound with respect to these auxiliary criteria.
- there is little consensus on what interpretability in machine learning is, or how to evaluate it for benchmarking.
- Currently, interpretability evaluation typically falls into two categories. The first evaluates interpretability in the context of an application. The second evaluates interpretability via quantifiable proxy i.e first claims that some models are interpretable for a given task, then optimize within those models.

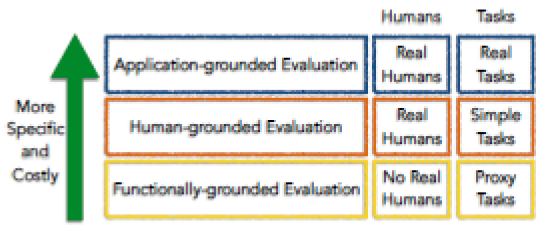


Figure 1: Taxonomy of evaluation approaches for interpretability

- The above chart shows the types of tests for interpretability discussed by this paper, that can be classified based on the level of human interaction and the standard of the tasks required for an approach.
- This paper states that Explanation is not necessary either because (1) there are no significant consequences for unacceptable results or (2) the problem is sufficiently well-studied and validated in real

applications that we trust the system’s decision, even if the system is not perfect. When the problem formalization is incomplete, creating a fundamental barrier to optimization and evaluation, then there is a need for interpretability.

- Then further the paper discusses ways in which interpretability can be quantified and finally states that A contribution that is focused on a particular application should be expected to be evaluated in the context of that application (application- grounded evaluation), or on a human experiment with a closely-related task (human-grounded evaluation). Also, We should categorize our applications and methods with a common taxonomy.

First Cut Approach

1. Starting from scratch first, the fundamentals from the first resource mentioned in the above section i.e. “The Illustrated Transformers”, and “ Attention is all you need” will be thoroughly read.

2. Following this, The 2 variants of the BERT i.e. RoBert and SciBert will be studied thoroughly from the respective research papers.

3. Seems that hugging face and pytorch libraries will have to be learned to get the respective programmatic implementation of the Bert and its variants.

4. After this, the Data has to be acquired from the resources mentioned in the respective paper. An NLP-based EDA will be done to see the nature of the lingual data.

5. The main goal will be to replicate the results from the paper “Understanding how Bert identifies edits”. If successful new scoring metrics will be tried, to furthermore increase the original scores if possible.

6. A debugging approach to the model using failure cases should be kept in mind.

7. Interpretability shall also be kept in mind, so tools like tensorboard, weight and biases shall be used to figure out the play of weights in different layers of bert.

8. More research shall be done. Also, youtube tutorials for hugging face and PyTorch shall be considered for learning purposes.

9. At last, an end to end editing web app can be a possibility, with Bert as its core.

Notes when you build your final notebook:

1. You should not train any model either it can be a ML model or DL model or Countvectorizer or even simple StandardScalar

2. You should not read train data files

3. The function1 takes only one argument “X” (a single data points i.e 1*d feature) and the inside the function you will preprocess data point similar to the process you did while you featurize your train data

    a. Ex: consider you are doing taxi demand prediction case study (problem definition: given a time and location predict the number of pickups that can happen)

    b. so in your final notebook, you need to pass only those two values

    c. def final(X):

preprocess data i.e data cleaning, filling missing values etc

compute features based on this X

use pre trained model

return predicted outputs

final([time, location])

    d. in the instructions, we have mentioned two functions one with original values and one without it

    e. final([time, location])   # in this function you need to return the predictions, no need to compute the metric

    f. final(set of [time, location] values, corresponding Y values) # when you pass the Y values, we can compute the error metric(Y, y_predict)

4. After you have preprocessed the data point you will featurize it, with the help of trained vectorizers or methods you have followed for your train data
5. Assume this function is like you are productionizing the best model you have built, you need to measure the time for predicting and report the time. Make sure you keep the time as low as possible
6. Check this live session: https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/4148/hands-on-live-session-deploy-an-ml-model-using-apis-on-aws/5/module-5-feature-engineering-productionization-and-deployment-of-ml-models