Reproducibility report instructions

Tyler Rada, Quinton Velazquez, Jacob Rowell

{radat19, rowellj17, velazquezq19}@ecu.edu

**Reproducibility Summary**

**Scope of Reproducibility**

Ivan Montero, Nikolaos Pappas, and Noah A. Smith proposed an approach that converts a pre-trained transformer language model into a sentence-level autoencoder. This autoencoder is able to reconstruct its pretraining data. The resulting model improves the performance of the pretrained model on sentence-level tasks while maintaining its performance on multi sentence tasks.

**Methodology**

Using a variety of hardware we were unable to get the models to train and thus we were unable to evaluate our results

The authors provided us with a link to their GitHub that had all of their code.

We also had access to the majority of the Datasets that the team used to train the Model.

We were trying to use our personal laptops(16GB of RAM, Intel Core i7) and Desktops(12GB of RAM. Intel Core i5 10400), We also attempted to use an Ubuntu VM with 60GB of memory allocated to it.

Training the Autobot was unfortunately a fail for our whole team.

**Results**

In conclusion, Yes, creating the autoencoder bottleneck system was successful for the original team. Unfortunately for our team we were not able to reproduce any of their tests. Following the original team's documentation they were able to show that Autobot was able to provide a 0.7% increase in accuracy over roBERTa.

**What was easy**

The initial installation was fairly simple. All we needed to do was copy the gitHub onto our computers and run a setup line. It took a few minutes for everything to install but it eventually worked.

First Installation took about 1 hour but it worked.

Second Installation using Ubuntu VM only took a few minutes.

**What was difficult**

Attempting to train the model itself was difficult. We kept running into errors while running the Datasets due to our personal machines not being strong enough.

Another issue we had run into is that if we were to replicate the larger experiments, our computers were not able to handle the information.

Training started off with around 15 hours but then started to throw errors and crashed.

We even tried to use the Ubuntu VM and allocated 60GB of Memory…Still failed.

**Communication with original authors**

Unfortunately we were unable to communicate with the original authors due to time constraints.

**1 Introduction**

Representation learning for text via pre-training a language model on a large scale has become a standard starting point for developing NLP systems. This approach stands in contrast to autoencoders, also trained on raw text, but with the objective of learning to encode each input as a vector that allows full reconstruction.

The authors explored the construction of a sentence-level autoencoder from a pretrained, frozen transformer language model. They adapted the masked language modeling objective as a generative, denoising one, while only training a sentence bottleneck and a single-layer modified transformer decoder.

They demonstrated that the sentence representations discovered their model achieve better quality than previous methods. The previous methods extracted representations from pretrained transformers on text similarity tasks, an example of controlled generation, and single-sentence classification tasks in the GLUE benchmark. All while using fewer parameters than large pretrained models.

**2 Scope of reproducibility**

Clearly itemize the claims you are testing:

• Claim 1: The fine tuning Autobot has Higher Accuracy with smaller test models over the pre-trained models.

• Claim 2: The larger the text, the Accuracy will start to drop significantly.

• Claim 3: Once the Autobot is trained using Yelp, it will far surpass the accuracy of the pre-trained models that are also trained with Yelp.

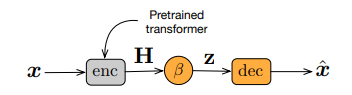
**3.1 Model descriptions**

The authors created AUTOBOT, a new autoencoder model for learning sentence “bottleneck” (i.e., fixed-size) representations from pretrained transformers that is useful for similarity, generation, and classification

The model has two unique components:

(i) a transformation that uses dot product attention to dynamically pool semantic information from the pretrained model’s hidden states into a sentence bottleneck representation

(ii) a shallow transformer decoder that is modified to operate based on the bottleneck representation



**Encoder:** The authors chose to keep the original encoder fixed and train a transformation β that will learn to compress H into a single representation z = β(H; θ), with θ being an additional set of parameters to be learned during finetuning.

**Decoder:** The Transformer decoder architecture expects hidden representations for every token input from the encoder in order for each output candidate to attend to each input token.

**Training considerations:** To avoid training the model from scratch, the creators fine tuned it for 100K optimization steps on a pretraining dataset using the base RoBERTa model.The model is trained using an input reconstruction loss by minimizing the negative log-likelihood computed over the reconstructed inputs.

**3.2 Datasets**

Since RoBERTa’s Datasets are not available publicly, the team had to use the Datasets that were used to train BERT.

GLUE (General Language Understanding Evaluation)- A collection of nine natural language understanding tasks, including single-sentence tasks CoLA and SST-2, similarity and paraphrasing tasks MRPC, STS-B and QQP, and natural language inference tasks MNLI, QNLI, RTE and WNLI.

[glue · Datasets at Hugging Face](https://huggingface.co/datasets/glue)

CHANGE-IT - dataset contains approximately 152,000 article-headline pairs, collected from two Italian newspapers situated at opposite ends of the political spectrum, namely la Repubblica and Il Giornale.

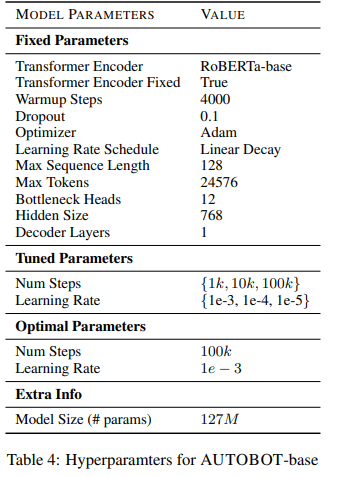
[gsarti/change\_it · Datasets at Hugging Face](https://huggingface.co/datasets/gsarti/change_it)

Both links provide the adequate source to run the Datasets. Additionally we also used the Dataset found with the TensorFlow website. TensorFlow provides the source code and examples.

[glue | TensorFlow Datasets](https://www.tensorflow.org/datasets/catalog/glue)

**3.3 Hyperparameters**

The Authors had provided the Hyperparameters that they had used.



**3.4 Implementation**

The authors provided us with a link to their GitHub that had all of their source code.

The languages used to create the Autobot model consisted of: Python (Most of it was done in this language), JavaScript, Shell, and C++.

[ivanmontero/autobot: Implementation of the paper 'Sentence Bottleneck Autoencoders from Transformer Language Models' (github.com)](https://github.com/ivanmontero/autobot)

**3.5 Experimental setup**

For all of the experiments involving base models, the authors used a computation cluster with 5 NVIDIA RTX 2080 TI GPU, 11GB GPU memory, and 128GB RAM.

For large models, we use a computation cluster with 4 NVIDIA TITAN RTX GPUs, 24GB GPU memory and 256GB RAM

The authors extracted the sentences from the BooksCorpus and English Wikipedia datasets to recreate the BERT dataset, and use RoBERTa-base’s pretrained tokenizer.

Unfortunately we did not have access to RoBERTa’s pre-training data in order to use their tokenizer.

<https://github.com/Jrowell1/autobot>

**3.6 Computational requirements**

For all of the experiments involving base models, the authors used a computation cluster with 5 NVIDIA RTX 2080 TI GPU, 11GB GPU memory, and 128GB RAM.

For large models, we use a computation cluster with 4 NVIDIA TITAN RTX GPUs, 24GB GPU memory and 256GB RAM

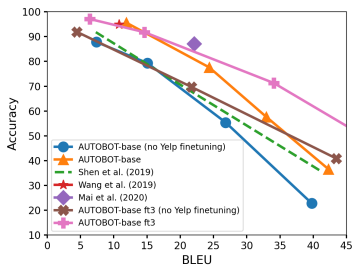
**4 Results**

For our Claims, yes the model results from the original team do prove our claims to be correct. They were able to show that when they tested smaller amounts of text, the Autobot was more accurate from the original models ranging from 0.2%-10.46%.

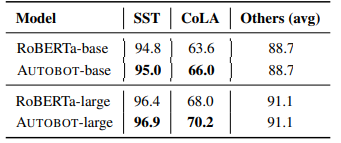
The second Claim was also proven to be true. When 100k lines were introduced the Autobot only had an accuracy of about 22% while the baseline of other models was around 40%.

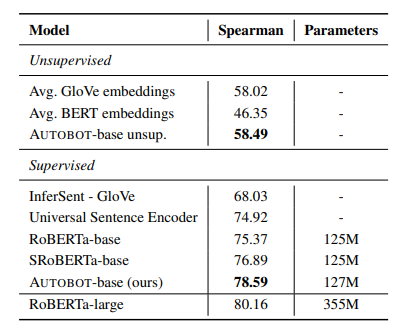
For our third Claim, while yes the Yelp trained Autobot had significantly better performance than its Base form at 100k lines, (it reached the 40% accuracy) other models had also jumped with accuracy, one even reaching about 60% accuracy.

**4.1 Result 1**

****

**4.2 Result 2**

****

****

**4.3 Additional results not present in the original paper**

With the complete failure of our experiments we can not give any additional results that were not in the original paper.

**5 Discussion**

**5.1 What was easy**

The readme for the project/research was pretty direct and simple to follow. For installation, we simply had to execute the bash file specified. Then we followed a link to attempt to preprocess the dataset from Wikipedia (WikiText-103) that was included in the repository. Unfortunately, that is where we ran into numerous issues, which we will discuss next.

**5.2 What was difficult**

Attempting to train the model itself was difficult. We kept running into errors such as, RequestDepencewarning (it wanted us to update to a version that was incompatible with the Datasets)

Another issue we had run into is that if we were to replicate the larger experiments, our computers were not able to handle the information.

Training started off with around 15 hours but then started to throw errors and crashed.

We experienced a few complete system crashes due to BERT

We even tried to use the Ubuntu VM and allocated 60GB of Memory…Still failed.

**5.3 Recommendations for reproducibility**

Unfortunately due to time constraints, we were not able to reach out to the original authors.

**References**

[Sentence Bottleneck Autoencoders from Transformer Language Models (aclanthology.org)](https://aclanthology.org/2021.emnlp-main.137.pdf)

[glue · Datasets at Hugging Face](https://huggingface.co/datasets/glue)

[gsarti/change\_it · Datasets at Hugging Face](https://huggingface.co/datasets/gsarti/change_it)

[glue | TensorFlow Datasets](https://www.tensorflow.org/datasets/catalog/glue)