CSCI 6040 Final Project Reproducibility Report

Amy Wightman
dunphya20@students.ecu.edu

James Rosebaugh-Nordan
rosebaughnordanj20@students.ecu.edu

Connor Bullard
bullardc21@students.ecu.edu

I. Introduction

The paper we selected studied a more topic-agnostic approach to same sentiment identification in the hopes of providing a method for better generalization towards various corpora. To accomplish their goal, the authors collected datasets from Yelp and Amazon and made use of the star ratings provided by users to split the data into positive versus negative sentiments, followed by training the BERT language model to accurately classify pairs of reviews as positive or negative each.

II. Scope of Reproducibility

The goal of this paper that we set out to reproduce is to determine if two separate texts have the same sentiment of either positive or negative. Primary data used for the study consisted of reviews of various businesses downloaded from Yelp. The authors sought to train a model that would be able to make that determination regardless of text or polarity–with regard to pairings, such as positive/negative, positive/positive, etc.–and fine tune it to be as accurate or more accurate than previous related works. A caveat for the model described in the paper is that the two texts being compared must be from the same context–in this case, the two Yelp reviews being compared at one time must both be about the same business. We strove to test the following claims from the original paper:

- BERT can be tuned to provide accurate classifications of the sentiments of two separate corpora, and
- It can do this regardless of either corpus' sentiment (i.e., one could be positive and the other could be negative)

III. Methodology

The authors of the paper have made their source code publicly available on GitHub. We were able to obtain the code used by the authors, which consisted primarily of Jupyter notebooks and Python files, and used these as our starting point for our project. We began our reproduction of the paper by attempting to recreate locally the findings and results in the authors' code. We made use of Google Colab for its collaborative functions, and as the semester progressed we switched briefly to using an ECU server to host a JupyterLab environment in the hopes that greater processing power would boost our potential for recreating the results of the paper. We had some success running the notebooks on our personal machines. The authors' code from Jupyter notebooks and Python files was a big help but we also further developed our final project by taking inspiration from various sources that have attempted to do similar work as this paper. Much time was spent organizing and importing the datasets and libraries we needed as the notebooks in the repo seemed to be missing some "!pip" statements. Additionally, an issue was opened on the authors' repository to make them aware of a version incompatibility with one of the dependencies [1]. Using the same Yelp reviews and that the authors of the paper used may have been our downfall because there was a massive corpus of data which turned out to be more than the resources we had available could handle. When we went to train the model, we got about 7 hours into the training before the system failed and we were at nearly 85% done, even with a smaller dataset. Using the BERT model seemed to need a lot of processing time which was constrained by the VPN timeout. We still wanted a proof of concept, so we derived smaller subsets of data (comprising 100, 1,000, and 10,000 reviews) and trained using these samples to determine if we could achieve accurate sentiment analysis. We also decidedd to switch our model to DistilBERT [2] to speed up processing time due to its lightweight footprint and faster processing speeds. We were able to train a model and evaluate reviews as positive or negative, but we were ultimately unable to reproduce the experiment as outlined in the paper, primarily due to lack of computing power.

A. Model

For their experiment, the authors made use of the BERT language model with an additional classification layer. This layer contained a dropout of 0.1, as well as a dense layer with sigmoid activation. Specifically, the authors used a pre-trained BERT model called BERT-base-uncased, which they finely tuned using pairs of reviews of either matching or mismatching sentiments. The data were split into training, validation, and test sets on a 80:10:10 ratio, respectively. In the test setup, the best model trained for 15 epochs, but only added increased accuracy by one percent over its alternatives.

B. Datasets

The datasets used by the authors in the paper were from Yelp and Amazon. We were able to obtain data from the yelp datasets using Kaggle. Since the files were large, we utilized

[1]The issue identified can be located at: https://github.com/webis-de/emnlp21-same-sentiment/issues/

[2]More information about DistilBERT is located at: https://huggingface.co/docs/transformers/model_doc/distilbert

Kaggle's API and were able to download the files directly to the ECU server. In order to speed up our processing, we tried multiple sizes of datasets. Initially using the authors code, we tried using smaller datasets to see if that would allow us to keep a connection on the server. Unfortunately the VPN timed out after 7 hours of training the model even on a dataset of only 1000 records. For our implementation attempt, we used the same yelp dataset with a variety of sizes. They were scaled down for the sake of ensuring we were able to reproduce the results at all, and to allow for quicker runs between changing parameters. We started with 100 records, then 1,000, and finally 10,000.

C. Hyperparameters

We were able to adjust the sample size fairly easily by changing some of the code but even using 20% of the data turned out to take so long to process and train that we were timed out of our server. We did see that increasing the sequence length from 128 to 512 and reducing the batch size to 6 from 32 made some difference so we had hoped to adjust those to see if there was a better combination but we did not get the chance. In our final model, we retained the maximum sequence length of 512, and instructed the DistilBERT model to train on the data in one epoch. We found that increasing the dropout slightly to 0.2 had a positive effect on the overall accuracy of our model.

D. Implementation

We used the code that had been written by the authors which was extensive and well documented, with a few pieces missing [1]. It was done completely in Python and we used the Jupyter notebooks to run the experiments. The Python dependencies include:

- Transformers
- Scikit-Learn
- Seaborn
- MLFlow
- Tensorboard

We used our own code to import the datasets from Kaggle and we also wrote code to fix some of the errors that we encountered, while following the implementation instructions provided which were mainly related to packages not being available because they had not been imported but were being used [2]. The author mentioned in his correspondence to us that they had gone through many iterations so we think that they may have had more packages installed than they realized so they probably left some of them out of the implementation instructions. In our attempt to recreate the claim of the paper, we used a DistilBERT model in the hopes that the more lightweight form of BERT would allow for faster processing and testing. Our model was inspired by several data science articles that we were able to glean knowledge from in developing our implementation [3] [4]. We also drew inspiration from other research papers on similar topics, specifically from a study on multi-modal embedding and its applications with sentiment analysis [5]. Similarly to the paper, we used the DistilBERT-base-uncased model, whereas the paper used the BERT-base-uncased model. Replicating the claim of comparing two texts simultaneously to determine the sentiment direction and whether they matched or not proved to be a difficult endeavor outside of the existing code, and if we had more time on the project perhaps we would have been able to achieve this implementation.

Hardware	Specification
Architecture:	x86_64
CPU op-mode(s):	32-bit, 64-bit
Byte Order:	Little Endian
Address sizes:	45 bits physical, 48 bits virtual
CPU(s):	6
On-line CPU(s) list:	0-5
Thread(s) per core:	1
Core(s) per socket:	3
Socket(s):	2
NUMA node(s):	1
Vendor ID:	AuthenticAMD
CPU family:	23
Model:	49
Model name:	AMD EPYC 7282 16-Core Processor
Stepping:	0
CPU MHz:	2799.999
BogoMIPS:	5599.99
Hypervisor vendor:	VMware
Virtualization type:	full
L1d cache:	192 KiB
L1i cache:	192 KiB
L2 cache:	3 MiB
L3 cache:	32 MiB

TABLE I
ECU SERVER HARDWARE

E. Experimental Setup

We used JupyterHub (provided by Dr. Herndon) to run the notebooks. We had to run JupyterLab and MLFlow on the server at the same time we were running the notebooks for everything to run properly as explained in the readme for the paper. The code can be found here: https://github.com/webis-de/emnlp21-same-sentiment. The resources available on the JupyterHub to which we had access can be seen in Table I. For the Jupyter notebook with the code we used in our attempt to recreate the claim of the paper, it was executed on a PC equipped with a Ryzen 5 3600 processor and an RX 580 graphics card with 32gb of RAM.

F. Computational Requirements

For the purposes of our scaled-back reproduction of the original study, the computational resources required would likely be met with any modern computer. We estimate that a CPU of at least a Ryzen 3 or Intel i5 caliber without a discrete graphics card would likely be sufficient for our work. Although the machines we used for our final attempts were more extravagant than these recommendations, they should suffice for a relatively speedy processing of our code against the smaller datasets. The computational resources needed to execute the code provided by the authors against the unaltered datasets appear to be far greater, at least from our experience. As previously mentioned, even seven hours of runtime on

the server provided to us by ECU was not sufficient for the entire Yelp dataset to be processed through a key Jupyter notebook. Advanced hardware with a discrete graphics card (ideally a newer and higher-end NVIDIA card in which a Jupyter notebook could make use of the CUDA codes) would be recommended.

IV. Results

In our experiment, although we were unable to recreate the mechanism of comparing two texts together and determining if they shared the same sentiment, we were successful in building a DistilBERT model to test against a smaller set of Yelp reviews extracted from the main dataset. Through our understanding of the original study and our own recreation, we determined that the experiment was definitely possible to be reproduced (at least, on a smaller scale) with better equipment and that the model could be trained to accurately determine sentiment, and therefore could determine if two separate texts share the same sentiment—that is, the same sentiment classification problem. As we worked through trying to reproduce the experiment using our own code, we understood that the data needed to be put through the preprocessing which was done as part of the original experiment. The authors had a significant amount of code written to process the data prior to training their models which we now more completely understand to have had a significant impact on the outcome of the predictions. Because we were unable to implement a test on two texts simultaneously, we focused our efforts on developing a language model to determine sentimentality of the Yelp reviews in our dataset. Our scaled back model and dataset took approximately 10-15 minutes to train on average, and the best result brought in an 88% accuracy rate for determining if the review was positive or negative. This accuracy was achieved after implementing a preprocessing step which included using regex filtering and lemmatization of words using the TextBlob library[3]

V. Discussion

By no means does our inability to recreate the results of the experiment ourselves indicate to us that the initial experiment was flawed in some way. From every appearance after studying the code and the findings claimed in the paper, we believe they are accurate representations and should be reproducible in the right environment. Utilizing a trained BERT model to determine sentiment classification of two separate texts proved to be a challenging task. As mentioned earlier in this report, if we had more time to continue the work perhaps we would have found a way to develop our own tests of comparing the texts together for sentiment analysis. At almost every step we encountered issues that consumed a significant portion of the time we had to complete the work–from the code issues that required manual testing and adaptations to resolve locally, to the resource constraints that prevented us from using the full datasets and the full BERT model. Despite these setbacks, this project was a rewarding and meaningful way to get a deeper appreciation for natural language processing and the high degree of attention to detail required in training models and proposing viable solutions. We each feel that we are completing this report and project with greater competency in this area of computer science and are not dissuaded from continuing to learn even with the struggles we faced.

[3]TextBlob documentation and additional information is located at: https://textblob.readthedocs.io/en/dev/

A. What Was Simple

It was easy to find the authors' code and data used. Reading through the code was fairly easy as well because the authors did a good job of writing comments for each section, so when we added code to try to reproduce or make the training faster it was clear which sections we needed to add to. The code was all well written and the comments make it easy to take parts that you might need to try to reproduce something similar. I believe it could be used to reproduce this experiment if you had more computing power. We were able to find information on all of the packages that they used because they used well known packages that had good documentation. They did write some of their own code, but having documentation available to read over to help us understand what they were trying to do helped a lot.

B. What Was Difficult

The data was available, but there were some hoops that we had to jump through like signing up for Kaggle and getting and using an API token. There are step by step guides though that are easy enough to follow. The most difficult thing would be finding a machine that could process all of the available data to make the model they used in a reasonable amount of time without failing. It took way more time than we expected to train the model. Other than those issues, there were several "!pip" commands that were not included in the notebook, so we did have to spend a while running the cells in the notebooks, reading the errors and adding in the missing pieces. Though it was tedious, it was not too difficult. The actual execution of the authors' code was also challenging. As previously mentioned we encountered several issues:

- Although the documentation provided by the authors was thorough and useful, at times the code required small tweaks (for example, in the D2_samesentiment_yelp_create_pairs.ipynb notebook, an import statement wrongly used "collection" instead of "collections" as a library name)
- At times the code changes required to calibrate the notebooks on our machines/Google Colab/ECU's server were extensive enough that several hours were devoted to attempting to correct it.

Due to this latter point, we were unable to completely run all the original code locally. It became pertinent for the sake of time and progress to attempt another direction for our study, which is why we began to look at alternative sources for implementing a BERT-type model for determining if two texts share sentiments.

C. Recommendations for Reproducibility

After encountering the issues of resource constraints and code changes required to make the notebooks work in our attempts, we would recommend to others reproducing this study what we began to focus on ourselves—boil the code down into its simplest parts with a focus on the goal of classifying two texts of varying sentiments. We read several articles that dealt with this issue that provided insight into what we were attempting, and we have included them as references below. We would recommend readers to use these and similar resources when trying to recreate this experiment themselves. Lastly, feel free to contact us or the authors of the paper, or leave a note on their GitHub repository with any concerns as they are typically prompt and considerate in their responses.

D. Communication with Author(s)

Throughout the semester we were able to communicate with the authors on several occasions. We emailed Erik Körner whose email was listed on the paper and he responded within a few days, with encouragement and answers to all of our questions. In addition to that he pointed us to a similar experiment that he referenced and he updated the github repo within a day of responding to us with a few tweaks he thought would be helpful. Once we ran into our resource issues we tried to contact him again, but he had mentioned earlier that he was busy with another project, and he had not responded so we were unable to get more details about the hardware he used to run the final version of the experiment. Despite this, we are grateful that he took the time to respond at all, and as mentioned his advice was genuinely useful and well-meaning.

REFERENCES

[1] Erik Körner, Ahmad Dawar Hakimi, Gerhard Heyer, and Martin Potthast. Github repository for casting the same sentiment classification problem. https://github.com/webis-de/emnlp21-same-sentiment.

[2] Amy Wightman, James Rosebaugh-Nordan, and Connor Bullard. Csci 6040 final project github repository. https://github.com/ymadh/nlp.

[3] Sentiment analysis in 10 minutes with bert and tensorflow. https://towardsdatascience.com/sentiment-analysis-in-10-minutes-with-bert-and-hugging-face-294e8a04b671. Accessed: 2022-04-19.

[4] Building a sentiment analysis model using yelp reviews and ml ensemble methods. https://towardsdatascience.com/building-a-sentiment-analysis-model-using-yelp-reviews-and-ml-ensemble-methods-80e45db6d0c7. Accessed: 2022-04-19.

[5] Limeng Cui, Suhang Wang, and Dongwon Lee. Same: Sentiment-aware multi-modal embedding for detecting fake news. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '19, page 41–48, New York, NY, USA, 2019. Association for Computing Machinery.