Sentiment Analysis on Shopee Reviews via Transfer Learning

Yasin Mohammed
Kulliyyah of Information Technology
IIUM Gombak
Selangor
Matric No. 1814111

Mohammed Raashid Salih Batha
Kulliyyah of Information Technology
IIUM Gombak
Selangor
Matric No. 1817209

Ahmad Nasali Bin Abd Manaf
Kulliyyah of Information Technology
IIUM Gombak
Selangor
Matric No. 1815091

Mahanad Chamie Majed Eddin
Kulliyyah of Information Technology
IIUM Gombak
Selangor
Matric No. 1625331

ABSTRACT

Online shopping provides us with a lot of benefits and advantages, especially during this pandemic. Reviewing products on Shopee can potentially become troublesome for customers and retailers, especially for new customers that are lurking around on Shopee to see whether a product is good or bad. This is because product reviews on Shopee can be inconsistent with the rating. In this project, we use transfer learning to train a deep learning model that can do sentiment analysis on Shopee reviews.

Keywords

Transfer learning; BERT; Shopee reviews

1. INTRODUCTION

1.1 Problem Statement

Product reviews can be inconsistent with their rating, thereby providing a skewed representation of the actual quality of the product. These can go under the radar of the seller, or the buyers, which can lead to poorer sales and less satisfaction overall.

1.2 Motivation

Some products get low ratings because the delivery time is later than expected (since more people use the platform and sometimes supply could not handle the high demand) and it is not fair for the product if it is actually really good. Using sentiment analysis to detect these inconsistencies and with transfer learning which technique of deep learning is a good approach to solve the problems.

2. RELATED WORK (MINIMAL)

2.1 Fang, X., Zhan, J. Sentiment analysis using product review data [1]

This paper proposes a general process with detailed steps on how to tackle one of the fundamental problems with regards to sentiment analysis: categorization of sentiment polarity. Sentiment polarity categorization concerns classifying a source text to one sentiment polarity, positive or negative. The middle of the region represents the neutral class. The source dataset for this experiment was Amazon product reviews, and sentiment analysis was conducted on a review level, and also on a sentence level and shows some promising results.

2.2 F. Zhuang et al., "A Comprehensive Survey on Transfer Learning" [2]

This paper provides a comprehensive review of the great variety of transfer learning mechanisms that have been developed over the years due to its popularity. It seeks to summarize and articulate more than 40 different transfer learning approaches, out of which more than 20 are used on an experimental basis to depict their performance. The datasets employed for this purpose include Amazon Reviews, Reuters-21578, and Office-31, primarily to demonstrate which transfer learning approach should be implemented for which problem domain.

2.3 X. Dong and G. De Melo, "A helping hand: Transfer learning for deep sentiment analysis" [3]

This paper demonstrates the viability of the transfer learning approach in general for the purposes of sentiment analysis. It recognizes the limitations of contemporary neural network techniques for solving sentiment analysis, which include the need for a substantial amount of data and also the lack of generality of the model obtained. The experiment was carried out on a variety of datasets spanning seven different languages, to great effect..

3. TECHNICAL BACKGROUND

3.1 Sentiment Analysis

Sentiment analysis (or opinion mining) is a natural language processing technique used to determine whether data is positive, negative or neutral. It's often used by businesses to detect sentiment in social data, gauge brand reputation, and understand customers. People around the world share their opinions on online social media, product review pages, booking websites, blogs, etc.

3.2 Transfer Learning

Humans often use their gained knowledge through experience to be applied to new tasks that they have never done before. One of the most powerful ideas of deep learning is that sometimes we can take knowledge from the neural network that has learned from one task and apply that to a separate task. While conventional machine learning technology has achieved great success and has been applied successfully in many realistic applications, some real-world scenarios still have a few limitations.

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. According to Zhuang, "Transfer learning aims at improving the performance of target learners on target domains by transferring the knowledge contained in different but related source domains." In other words, if the first task of the model is about image recognition between a dog and a cat, the model can be reused to new related tasks such as x-ray recognition for diagnosis treatment and it can be more specific.

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. According to Zhuang, "Transfer learning aims at improving the performance of target learners on target domains by transferring the knowledge contained in different but related source domains." In other words, if the first task of the model is about image recognition between a dog and a cat, the model can be reused to new related tasks such as x-ray recognition for diagnosis treatment and it can be more specific.

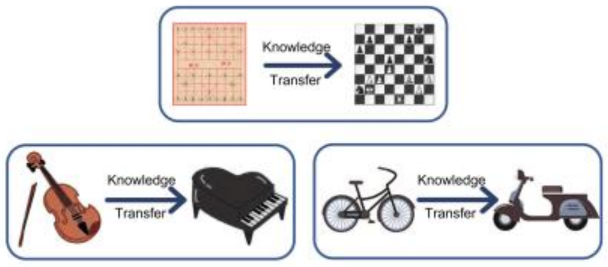


Figure 1. Intuitive diagram of transfer learning

Figure 1 depicts an intuitive picture of transfer learning inspired by human beings' capabilities to transfer knowledge across domains. Transfer learning aims to leverage knowledge from a related domain (called source domain) to improve the learning performance or minimize the number of label examples required in a target domain. Transfer learning comes in handy when creating a new model from scratch since the latter requires a lot of resources. So, instead of just text categorization which the existing model already does, we could train the existing model to do sentiment analysis specifically for product reviews on Shopee.

The ideal scenario of machine learning is that there are abundant label training instances, which have the same distribution as the test data. However, collecting sufficient training data is often expensive, time-consuming, or even unrealistic in many scenarios. For instance, data for Shopee product reviews is enormously large and raw to be trained. There are many products advertised on the platform and most of them are written in different languages even some of them have internet slang and emojis. Transfer learning can partially solve this problem by relaxing the need of mass unlabelled data from the product reviews and predict it using the pre-existing model that is similar with regards to the problem at hand.

3.3 Transfer Learning and Sentiment Analysis

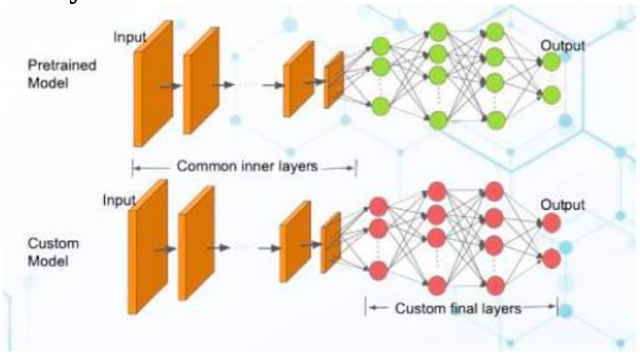


Figure 2. Transfer learning within deep neural networks

Figure 2 shows how transfer learning can be adapted for new related tasks at a deep learning level. BERT algorithm is a natural language processing algorithm by Google for text classification and to find the meaning of search query on its search engine. Since, as mentioned earlier, that training a model from scratch requires a lot of resources, the appealing strategy is to tune the existing model to fit into another task such as ours. This is because it has similarity toward text classification in general and is suitable to do advance sentiment analysis with big data sets. According to Dong, in 2018 deep learning in neural networks is widely used in sentiment polarity classification but suffers from their dependence on very large annotated training corpora [3]. This is because the nature of online discussion on the internet is rapidly changing as it is a difficult to train the dataset on a large amount of data, as it has already been trained. In simpler words, the dataset is only implemented into the algorithm with a hugging method and tuned to get the expected output so that it makes sense towards the problems.

The relevance of using transfer learning here is that our problem is now easier since we only need to prepare the dataset by processing and standardizing such as removing the not so important attributes such that it fits toward the algorithm.

3.4 Transformers

The Transformer was introduced in 2017 as a deep learning model, mainly used in the natural language processing (NLP) domain. Transformers are designed, like recurrent neural networks (RNNs), for tasks such as translation and text summarization, to handle sequential data, such as natural language. Transformers, however, do not require that the sequential data be processed in sequence, unlike RNNs. For instance, the Transformer does not need to process the beginning of it before the end of the input data if the data is a natural language sentence.

The Transformer facilitates much more parallelization than RNNs because of this feature and thus reduces training times. Since the Transformer model enables more training parallelization, it has allowed for training on larger datasets than was possible before it was implemented. This has led to the development of pre-trained systems like BERT (Bidirectional Encoder Representations from Transformers) as mentioned earlier.

Transformer uses the attention-mechanism. It uses with the aid of two pieces, Transformer is an architecture for turning one sequence into another (Encoder and Decoder). On the left is the Encoder and on the right is the Decoder. Both the Encoder and

Decoder are composed of modules that can be stacked multiple times on top of each other, defined in the figure by Nx. We see that the modules predominantly consist of layers of Multi-Head Focus and Feed Forward. As we cannot use strings directly, the inputs and outputs (target sentences) are first embedded in an n-dimensional space.

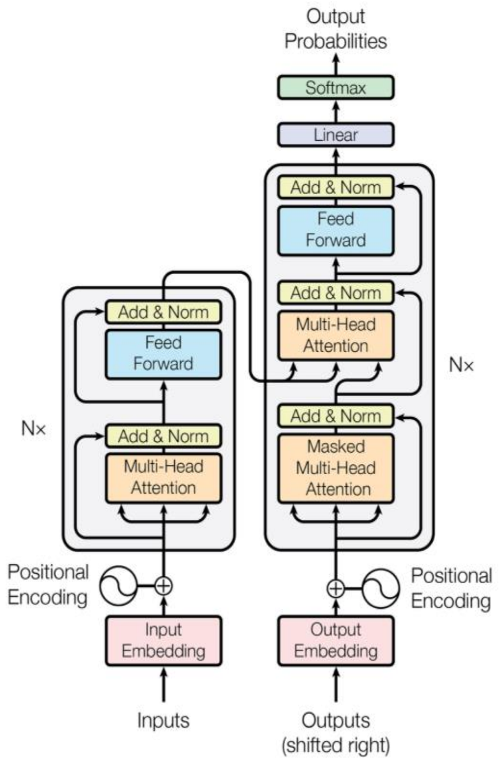


Figure 3. The Transformer - model architecture

The three kinds of Attention possible in a model:

1. Encoder-Decoder Attention: Attention between the input sequence and the output sequence.
2. Self attention in the input sequence: Attends to all the words in the input sequence.
3. Self attention in the output sequence: One thing we should be wary of here is that the scope of self attention is limited to the words that occur before a given word. This prevents any information leaks during the training of the model. This is done by masking the words that occur after it for each step. So for step 1, only the first word of the output sequence is NOT masked, for step 2, the first two words are NOT masked and so on.

4. FORMAL WELL-MOTIVATED DESCRIPTION OF YOUR APPROACH

The approach for generating our model may be broken down into the following sections:

4.1 Data Collection

Before we could delve into the actual training process, we needed a dataset to train our model. As already discussed in the introduction, we intended to train a deep learning model that could be used to do sentiment analysis on Shopee product reviews. Fortunately, we did not have to search extensively for a dataset that could achieve what we intended. In the middle of last year, the Shopee career team organized a 2-month coding challenge covering various topics under data analytics and data science, and among them was a challenge to create a sentiment analysis model for Shopee reviews. The dataset for the latter was uploaded on Kaggle by Shopee and contains records of 146,811 user reviews consisting, each record consisting of the comment along with the rating on the scale of 1 to 5 [8]. We used this specific dataset for training our model.

4.2 Data Preparation

The dataset that was downloaded has 146,811 records. As it was scraped from the Shopee website by the Shopee team, it is likely that several of the reviews are duplicates from users having the same comment and rating. Having these duplicates in our training set would not be useful and would only increase the time it takes to train the model. So, it was better that we dropped these from our dataframe. To make sure same reviews with different cases are also dropped, the cases were also lowered for all of the comments in the dataset, before the duplicates were dropped using the drop_duplicates() method. After dropping these records, the total number of records remaining were 116,697, which around 30k less rows than the original dataset.

The dataset that we have contains ratings on the scale of 1 to 5, as already mentioned. But it lacks sentiment labels for the reviews. This label is necessary to train our model for sentiment analysis. To work around this, a sentiment column was augmented into our dataframe based on the rating in each record. Ratings ranging from 1 to 2 were assigned the 'negative' label, 3 assigned the 'neutral' label and lastly, the ranges 4 to 5 were assigned the 'positive' label.

With the above, the data was more or less ready for training. Further pre-processing steps such as tokenizing, cleaning, etc. did not need to be carried out at this stage, the reason for which is explained later.

4.3 Pre-Training Configuration

Instead of training a completely new model from scratch, we mentioned that we intended to use transfer learning on an already existing pre-trained model for sequence classification. We defined 'bert-base-uncased' as the base model that will be fine-tuned for our dataset. It is a popular transformers model released by Google and was trained on a very large corpus to carry out tasks such as sequence classification, token classification and question-answering. Hence, it was suited for our particular task here, which comes under sequence classification. Once the model is defined, the Blurr library does the job of downloading the model from the Hugging Face repository and also getting the associated architecture name and tokenizer for the model. These are returned as objects by the Blurr.get_hf_objects() method and stored in separate variables in the code. After this step, we define a Datablock object. In FastAI library, a Datablock is an API that abstracts the task of preparing and transforming the data for training. It allows us to define the block type, which describes the type of the data being used, the transformations to be applied on our data, and also the features and labels of our data. The Hugging Face objects that we retrieved in the last step are passed as arguments to HF_TextBlock method of the Blurr library, which creates a wrapper object that can then be passed as an argument in the Datablock method. The Datablock doesn't carry out any transformations in and of itself at this stage. It just creates a blueprint describing how the data should be prepared and transformed for training. Actual operation on data is initiated by calling the dataloaders method of the Datablock object. We also pass the batch size at this stage. At this stage, the data is collected, transformed and compiled into batches and returned as a

DataLoader object. This object contains the actual pre-processed data and is ready to be used for training.

4.4 Training

To initiate the training, we need to create a new object of a FastAI class called Learner. This object is created by passing the DataLoader, the model, an optimization function, a loss function, a list of metrics, a list of callbacks and a splitter function. An optimization function handles the learning rate of the parameters during neural network training. It adjusts the learning rate according to the algorithm defined by the particular optimizer function. Learning rate defines how quickly the neural network adapts the model parameters to come up with a solution. By default, the Adam optimizer is used in FastAI to create the learner object. The loss function is closely connected to the optimizer function as the calculation of learning rate is based on it. It is usually a measure of how far the actual output of the model is from the intended result. Next, we have the metrics, which defines what model evaluation metrics are used during the training phase. We also have callbacks which lists the functions that would be called at various stages of the training. And lastly, we have the splitter function that defines how the model parameters are split for training.

The above, like the datablock object, just creates a structure for the learner. We call the create_opt() method to create the optimizer function specific to the learner structure that we defined. Then we freeze all of the layers of the model except the last one by calling the freeze function. This is an important step of transfer learning. We initially only want to adjust the last layer, and do not wish to modify the weights in the other layers of the model, as they're already well trained. Hence, those layers are frozen. We then find the an appropriate learning rate for the training using a custom function taken from the FastAI forum (Chang, 2019) [7]. The the model is then fit for one epoch, so that the last layer is adjusted to our data. Once we have a somewhat trained last layer, we can then unfreeze the remaining layers and carry out few more epochs of training so that all the layers are now being adjusted to fit our data. The pre-trained would not be greatly affected now as the last layer has already taken much of burden of getting the right output during the first round of training. And with this, we finally have a model fine-tuned for our use case.

4.5 Testing

There are multiple ways of testing our model. The first is by calling the show_results method, which carries out the sentiment on a portion of the validation set. The number of results to be displayed can be modified by changing the max_n parameter. In addition to that, we can also use a special method provided by the Blurr library to do sentiment analysis on a string directly. The string is passed as an argument and the prediction is then returned as an output. The final way, that we decided, to test our model was by scraping the reviews of a product from the Shopee website into a dataframe and then using the test_dl method to turn it into a compatible DataLoader set. This can then be passed in the aforementioned show_results method and will output the results of the sentiment analysis in the same manner as it did with the validation set.

5. EXPERIMENTAL SETUP & IMPLEMENTATION EXAMPLE

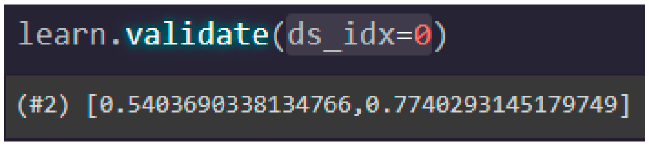
We used Google Colab as the working environment. Training a deep learning neural network model on a CPU only environment is heavily time consuming and inefficient. Fortunately, Google Colab provides a GPU based runtime with decent enough memory allocation to train our model for free. It also comes pre-installed with some of the important libraries that are required for data pre-processing and training, including Pandas, NumPy, FastAI, Scikit-learn and PyTorch. In our case, we only had to manually install the ohmeow-blurr library to get ourselves started.

We trained the model at first with the dataset we downloaded without dropping any duplicates from our dataset for 3 epochs. This took 12 minutes for each epoch. Then we trained the model with the dataset having the duplicates dropped. In this case, the training time halved to 6 minutes per epoch. Lastly, we trained it as described in our approach, first with all the layers frozen except the last for one epoch, which took 10 minutes, and then with all the layers unfrozen for 7 epochs, where each took 17 minutes.

6. RESULTS

6.1 First model

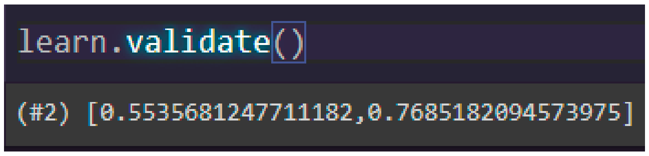
In our first training with the dataset rows not dropped, we achieved an accuracy of 77.4% on the training set:



```
learn.validate(ds_idx=0)

(#2) [0.5403690338134766,0.7740293145179749]
```

Figure 4. Training set accuracy of first model

And an accuracy of 76.8% on the validation set:

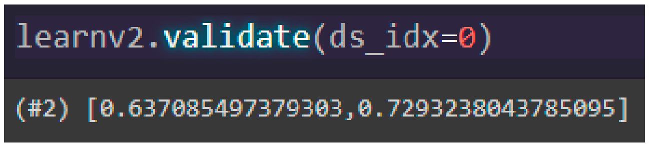


```
learn.validate()

(#2) [0.5535681247711182,0.7685182094573975]
```

Figure 5. Validation set accuracy of first model

6.2 Second model

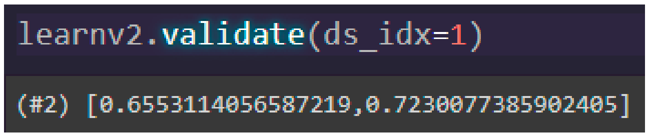
In the second training, we achieved an accuracy of 72.9% on the training set:



```
learnv2.validate(ds_idx=0)

(#2) [0.637085497379303,0.7293238043785095]
```

Figure 6. Training set accuracy of second model

And an accuracy of 72.3% on the validation set:

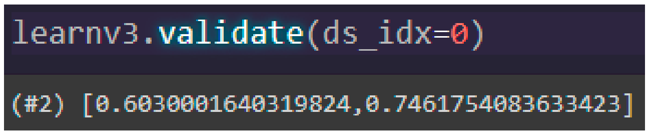


```
learnv2.validate(ds_idx=1)

(#2) [0.6553114056587219,0.7230077385902405]
```

Figure 7. Validation set accuracy of second model

6.3 Third model

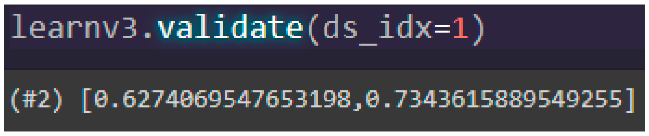
And lastly, in the third model, we got an accuracy of 74.6% on the training set:



```
learnv3.validate(ds_idx=0)

(#2) [0.6030001640319824,0.7461754083633423]
```

Figure 8. Training set accuracy of third model

And an accuracy of 73.4% on the validation set:



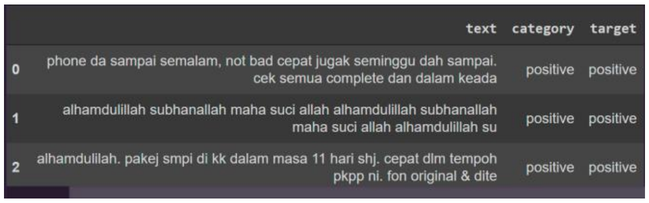
```
learnv3.validate(ds_idx=1)

(#2) [0.6274069547653198,0.7343615889549255]
```

Figure 9. Validation set accuracy of third model

In all of the above images, the first number represents the loss and the second number represents the accuracy.

6.4 Results on scraped data

We also have the following results from our scraped data. The category column represents the actual category and target represents the prediction.



	text	category	target
0	phone da sampai semalam, not bad cepat jugak seminggu dah sampai. cek semua complete dan dalam keada	positive	positive
1	alhamdulillah subhanallah maha suci allah alhamdulillah subhanallah maha suci allah alhamdulillah su	positive	positive
2	alhamdulilah. pakej smpi di kk dalam masa 11 hari shj. cepat dlm tempoh pkpp ni. fon original & dite	positive	positive

Figure 10. Sentiment analysis on scraped data

7. ERROR ANALYSIS

From the confusion matrix, two important inferences can be observed:

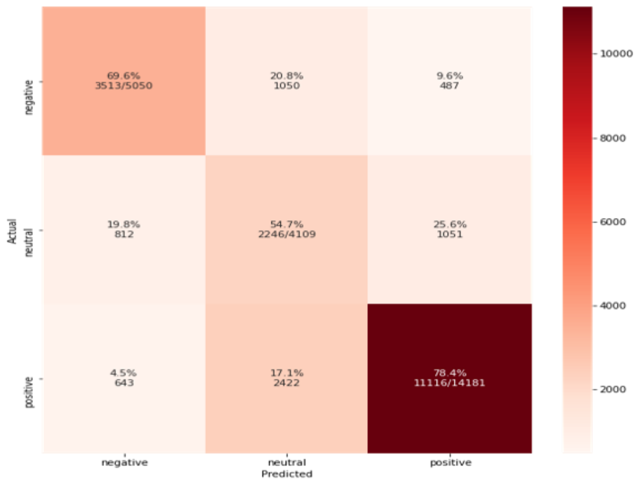


Figure 11. Confusion matrix

First, the derived confusion matrix is unlike the traditional confusion matrix with four cells, representing true positive, true negative, false positive, and false negative. This is due to the fact that our problem is not a binary prediction problem, but rather, a multiclass prediction problem. We have predictions for positive, negative, and the additional category of neutral, which results in a confusion matrix of nine cells.

Second, a very significant inference can be observed, and this is regarding the neutral class. All of the predictions involving the neutral class (other than the neutral-neutral pairing), has a much higher error rate as compared to the positive and negative classes. In other words, the positive-negative misclassification is much lower than the predictions involving the neutral class.

Predicting neutral classes, is inherently a tad unreliable. This is because neutral classes, by definition, are usually somewhere in between positive and negative classes. They don't have any unique characteristics of their own, generally speaking. This can lead to the higher misclassification rate, as observed from the confusion matrix.

The issue with neutral prediction is further exacerbated by the moderacy response bias. This is a phenomenon where a participant who is required to pick a rating on a scale usually gravitates towards the middle of the response scale (Bogner & Landrock, 2016) [6]. The way our preprocessing is defined, a score of 3 out of 5 qualifies in the neutral category, which happens to be the middle of the scale.

8. REFERENCES

[1] Fang, X., Zhan, J. Sentiment analysis using product review data. Journal of Big Data 2, 5 (2015). https://doi.org/10.1186/s40537-015-0015-2

[2] F. Zhuang et al., "A Comprehensive Survey on Transfer Learning," Proc. IEEE, vol. 109, no. 1, pp. 43–76, 2021, doi: 10.1109/JPROC.2020.3004555.\

[3] X. Dong and G. De Melo, "A helping hand: Transfer learning for deep sentiment analysis," ACL 2018 - 56th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf. (Long Pap., vol. 1, pp. 2524–2534, 2018, doi: 10.18653/v1/p18-1235.

[4] Kulshrestha, R. (2020, November 22). Transformers in NLP: A beginner friendly explanation | Towards Data Science. Medium. https://towardsdatascience.com/transformers-89034557de14

[5] M. (2020, September 22). What is a Transformer? - Inside Machine learning. Medium. https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04

[6] Bogner, K., & Landrock, U. (2016). Response Biases in Standardised Surveys. GESIS Survey Guidelines. Mannheim, Germany: GESIS – Leibniz Institute for the Social Sciences. doi: 10.15465/gesis-sg_en_016

[7] Chang, A. 2019. Automated Learning Rate Suggester. https://forums.fast.ai/t/automated-learning-rate-suggester/44199.

[8] Suitnatsnoc, L. (2020, August 03). Shopee Code League 2020 Data Science. Retrieved January 24, 2021, from https://www.kaggle.com/davydev/shopee-code-league-20