INT3405E 41 HHM's Report

Hoàng Đức Mạnh
ID: 20020080
K65-CA-CLC2

Document Writer
Resnet and VGG coder

Nguyễn Đăng Huỳnh
ID: 20020295
K65-CA-CLC1

Document Writer
Focal loss function coder

Phạm Nguyễn Tuấn Hoàng
ID: 20020015
K65-CA-CLC1

PhoBERT baseline coder
README and requirement writer

***Abstract*— In the food industry, it is crucial to consider customer reviews. However, reading every comment to determine whether those are positive or negative is a tedious process. To make it easier and faster, machine learning is used to do the hard work for humans. The dataset that we are using is from Foody, a software that let people search, rate, and comment on eating places: restaurants, coffee shops, bars, karaoke, bakery, … in Viet Name. The data consists of 9071 examples, each telling us about customer reviews, images about the place, and customer rankings. We use 3 models to solve this problem: PhoBERT, Resnet-18, and VGG-11. Our approach gives us the ROC-AUC score of 0.94, 0.89, and 0.903 respectively for each model. Our code is available at: https://github.com/11TnM06/SentimentAnalysis**

***Index Terms*— INT3405E 41, sentiment analysis, top 7 in class, top 20 in general**

I. Introduction

In this technology age, many people use software to analyze basically everything such as shopping, tourists, products, and food, … This is done by left rating prior people who have experienced those things. If something has a high score means that people like it, but if the score is low then probably that place will run out of business.

In this paper, we will focus on the food industry. In terms of the dataset, it is collected from a software called Foody, which let people criticize the place that they are experiencing. It can be a restaurant, bakery, karaoke, coffee shop, etc. Foody can be found on both website and application platforms.

Our purpose is to classify whether people like the place or not by analyzing the comments and pictures they left. This problem is known as a Sentiment Analysis Problem. The machine needs to be trained to analyze and understand emotions in customer comments to classify whether that is a compliment or a criticism. The dataset used in this problem is from Foody, which contains 5 columns:

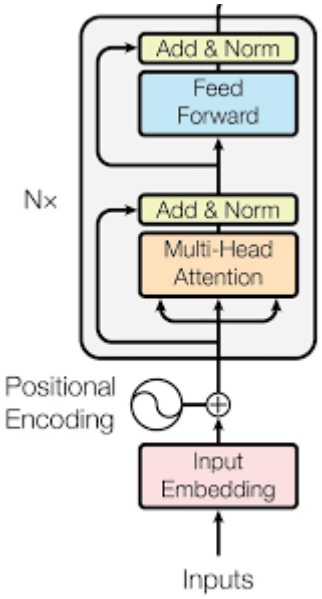
- RevId: Review ID.
- UserId: User ID.
- Comment: User's comment on the review.
- image_urls: The URL address contains images related to the review.
- Rating: The rating of the review.

We are going to classify the dataset into 2 classes (positive/negative), so a threshold of 6 is used to determine which one is in which class. This means if the Rating is greater or equal to 6 then we put it in class 1 and 0 otherwise.

II. Definitions

A. BERT

BERT stands for Bidirectional Encoder Representations from Transformers. This model is basically many Transformer's encoders stacking onto one another; hence, its name. BERT has 2 version: $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$.



The difference between the 2 versions is $\text{BERT}_{\text{base}}$ has 12 encoder layers and $\text{BERT}_{\text{large}}$ has 24 instead.

B. RoBERTa

RoBERTa stands for Robustly Optimized BERT Pre-training Approach. Its purpose is to optimize the training of BERT architecture in order to take lesser time during pre-training. RoBERTa's architecture is similar to its ancestor BERT, but some changes have been applied to improve the results. These changes are:

- Removing the Next Sentence Prediction (NSP) objective
- Training with bigger batch sizes & longer sequences
- Dynamically changing the masking pattern

C. PhoBERT

Just like BERT, PhoBERT also has 2 versions "base" and "large". They are the first public large-scale monolingual language models pre-trained for Vietnamese. PhoBERTbase and PhoBERTlarge use the same architectures as BERTbase and BERTlarge, respectively; however, PhoBERT pre-training approach is based on RoBERTa.

The reason why this model exists is that the success

of pre-trained BERT and its variants has largely been limited to the English language. Moreover, existing pre-trained multilingual BERT-based models have 2 main concerns:

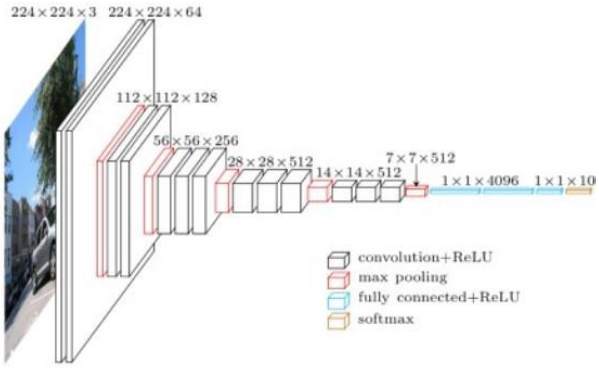
- They are trained using The Vietnamese Wikipedia corpus, which has only 1GB of data. It is relatively small.
- All publicly released monolingual and multilingual BERT-based language models are not aware of the difference between Vietnamese syllables and word tokens.

To handle the first concern, PhoBERT is pre-trained using a 20GB dataset of uncompressed texts. Furthermore, the second concern is solved with the help of RDRSegmenter from VnCoreNLP to perform word and sentence segmentation on the pre-training dataset. For optimization, the RoBERTa implementation in fairseq is employed.

D. VGG

VGG, also known as VGGNet, stands for Visual Geometry Group. It is a classic deep Convolution Neural Network (CNN) architecture with multiple layers. The "deep" here refers to a number of layers in a VGG network. For instance, VGG-16 has 16 convolutional layers and VGG-19 has 19 layers.

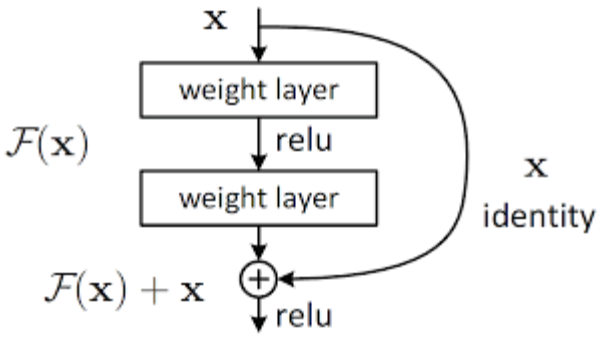
For this Foody problem, we apply VGG-11.



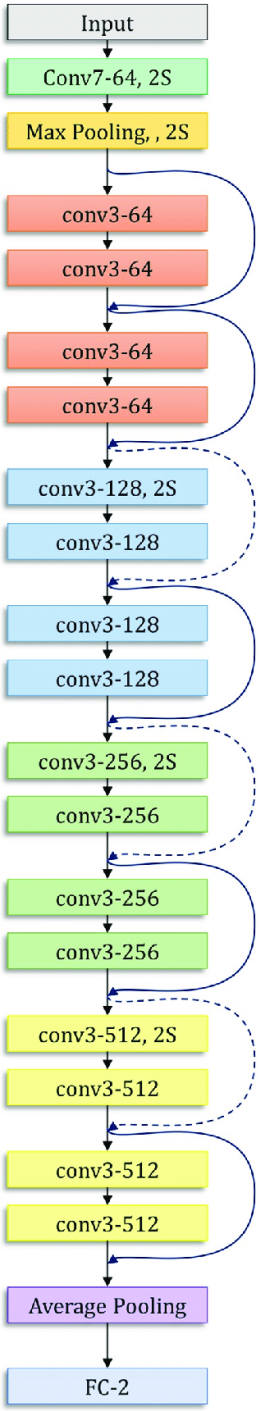
E. Resnet

Resnet is a *CNN* model to work with hundreds of layers. The problem when building CNN model with many convolution layers is *Vanishing Gradient* will occur. Vanishing Gradient or Exploding Gradient usually happens to deep neural networks. Neural networks do gradient descent by finding the derivatives of the loss function to each parameter from the output back to the input. In reality, gradients usually get smaller and smaller after getting to lower layers. This leads to unconverging weights since they only get a slight update.

The solution that Resnet offers is to use a uniform "skip" connection to pass through one or more layers. Such block is called Residual Block.



For our Sentiment Analysis Problem, we use Resnet-18 to train the Foody dataset. The number "18" here refers to the number of layers in the network



F. Focal Loss

1) Cross Entropy Loss

Before getting into *Focal Loss*, let's learn about some fundamentals first. Firstly, let's get to know about *Cross Entropy Loss:*

$$CE = -\sum_{i=1}^{n} y_i \log(p_i) \quad (1)$$

Where y is the true label and p is the predicted probability.

The drawback of the loss function (1) is that it doesn't fine the machine harshly when it predicts wrong. Consequently, a different loss function is needed for an imbalanced dataset.

2) Balanced Cross Entropy

The most natural solution for the above issue is to apply weights which is the inverse of the label frequencies. The new loss function is called *Balanced Cross Entropy*

$$BCE = -\sum_{i=1}^{n} \alpha_i \log(p_i) \quad (2)$$

where $\alpha_i = \frac{1}{f_i + \epsilon}$, f_i is the frequency of class i, ϵ is a small positive number to avoid zero-division. Alternatively, α can also be a hyperparameter which is in [0, 1].

Balanced Cross Entropy is a loss function which can balance out the data distribution. Nonetheless, it doesn't really change the gradient descent of the loss function.

3) Focal Loss

Unlike its ancestor, *Focal Loss* is superior. It can deal with imbalanced data and has a strong effect on gradient descent so that the model doesn't only learn on the labels which have a high frequency. *Focal Loss* is first introduced in RetinaNet:

$$FL = -\alpha_i (1 - p_i)^{\gamma} \log(p_i) \quad (3)$$

γ is usually chosen in [0; 5]

As can be seen from (3) that *Focal Loss* is just the product of *Balance Cross Entropy* with $(1 - p_i)^{\gamma}$, but this has a strong impact on the loss function and gradient descent on the same time. Let's take a look at 2 cases:

- Easy prediction: It is obvious that a model trained on imbalanced data can predict more precisely on high-frequency labels. This is also known as "easy prediction". The probability, in this case, tends to be high; therefore $(1 - p_i)^{\gamma}$ tends to be small and has slight effect on the loss function.
- Hard prediction: In this case, p_i will be smaller; hence, $(1 - p_i)^{\gamma}$ will be closer to 1. This affects more compared with the "easy prediction" case.

Now let's consider the gradient:

$$\frac{\partial FL}{\partial p_i} = \alpha_i (1 - p_i)^{\gamma} \left[\frac{\gamma \log(p_i)}{1 - p_i} - \frac{1}{p_i} \right] \quad (4)$$

- Easy prediction: $(1 - p_i)^{\gamma}$ is small, so it doesn't have any effect on gradient descent
- Hard prediction: $(1 - p_i)^{\gamma}$ is big, which have more impact on gradient descent

G. K-Fold Cross Validation

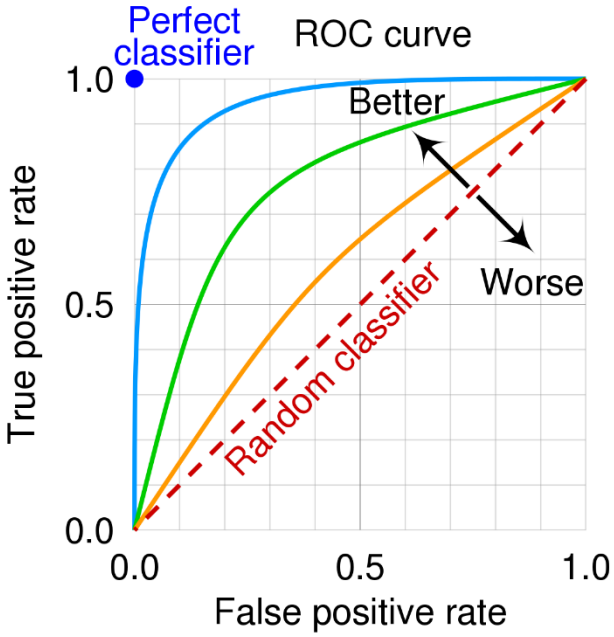
In order to evaluate our model, Cross Validation is commonly used. Its basic principle is dividing the dataset into 2: a training set and test set. Machine will use the training set to learn and the test set to analyze its performance.

The drawback of regular Cross Validation is that we need to figure out the splitting ratio to maximize both training set and test set so that the model can learn and be examined fairly. Hence, *K-Fold* is used to solve the problem.

Instead of dividing the dataset into 2, it is split into k equal portions. Then for each part, the current part is for evaluation and the remaining is for training. After getting all analyses, their average is the result.

H. ROC-AUC

ROC-AUC or AUROC, the area under the receiver operating characteristic, is a performance metric that you can use to evaluate classification models. AUROC tells you whether your model is able to correctly rank examples. AUROC is thus a performance metric for "discrimination": it tells you about the model's ability to discriminate between positive examples and negative examples.

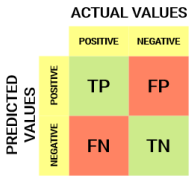


The figure above shows some example ROC curves. The AUROC for a given curve is simply the area beneath it.

The worst AUROC is 0.5, and the best AUROC is 1.0.

- An AUROC of 0.5 (area under the red dashed line in the figure above) corresponds to a coin flip, i.e. a useless model.
- An AUROC less than 0.7 is sub-optimal performance
- An AUROC of 0.70 – 0.80 is good performance
- An AUROC greater than 0.8 is excellent performance
- An AUROC of 1.0 (area under the blue line in the figure above) corresponds to a perfect classifier

The AUROC is calculated as the area under the ROC curve. A ROC curve shows the trade-off between a true positive rate (TPR) and a false positive rate (FPR):



$$TPR = \frac{TP}{TP+FN} \quad (5)$$

$$FPR = \frac{FP}{FP+TN} \quad (6)$$

III. Our Approach

Now, all the required knowledge has been given, let's apply them for classifying food reviews using Sentiment Analysis techniques.

A. Train with PhoBERT

One thing that needs to be concerned when training the model with PhoBERT with a Vietnamese dataset is to segment words first. For example, this sentence "Tôi là sinh viên trường Đại học Công nghệ" forms 6 words after segmentation "Tôi là sinh_viên trường Đại_học Công_nghệ". Luckily, this can be done with the help of RDRSegmenter from VnCoreNLP.

1) Preprocessing

Before applying the dataset to train the model, it is a good approach to preprocess the data first.

Firstly, we only need to classify customers' ratings, so converting them to 0/1 is necessary. As said before, we use a threshold of 6 to determine which rating gets into which class.

Secondly, VnCoreNLP's tokenizer is used to segment each comment. However, this tokenizer returns a list of segmented words, we need to merge them into a sentence

2) Training

To evaluate our model, we ask KFold for help. Here, the dataset is split into 10 sections:

	Number of positive labels in train set	Number of positive labels in test set
Partition 1	6423/8163	724/908
Partition 2	6433/8164	714/907
Partition 3	6450/8164	697/907
Partition 4	6430/8164	717/907
Partition 5	6453/8164	694/907
Partition 6	6407/8164	740/907
Partition 7	6429/8164	718/907
Partition 8	6424/8164	723/907
Partition 9	6441/8164	706/907
Partition 10	6433/8164	714/907

From this table, the dataset is imbalanced. This can be solved by using Focal Loss. We use hyperparameters $\alpha = 0.25$ and $\gamma = 2$ to train our model. Finally, for scoring, ROC-AUC score is applied.

After training, we get this result:

	Training		Validating
	Loss	***AUROC***	***AUROC***
Partition 1	0.0191	0.9706	0.9866
Partition 2	0.0117	0.9882	0.9922
Partition 3	0.0078	0.9949	0.9989
Partition 4	0.0047	0.9976	0.9999
Partition 5	0.0031	0.9991	0.9999
Partition 6	0.0020	0.9996	0.9996
Partition 7	0.0014	0.9997	0.9999
Partition 8	0.0012	0.9997	1.0000
Partition 9	0.0013	0.9998	0.9998
Partition 10	0.0014	0.9999	1.0000

B. Training with Resnet and VGG

For Resnet and VGG, specifically Resnet-18 and VGG-11, we not just only use customer comments, but images are also applied to train.

Since the dataset provides us with links to images, we need to download all of them first. The preprocessing is pretty much the same as above, but for images, we use some transformers: resize, center crop, and normalize.

This is the result 2 models give us:

	Training		Validating
	Loss	***AUROC***	***AUROC***
Partition 1	0.0417	1.0000	0.9192
Partition 2	0.0313	1.0000	0.9456
Partition 3	0.0270	1.0000	0.9699
Partition 4	0.0207	1.0000	0.9915
Partition 5	0.0162	1.0000	0.9983
Partition 6	0.0114	1.0000	0.9994
Partition 7	0.0088	1.0000	0.9996
Partition 8	0.0078	1.0000	1.0000
Partition 9	0.0056	1.0000	0.9999
Partition 10	0.0043	1.0000	1.0000

Table 1: Train with Resnet-18

	Training		Validating
	Loss	***AUROC***	***AUROC***
Partition 1	0.0422	1.0000	0.8909
Partition 2	0.0324	1.0000	0.9296
Partition 3	0.0221	1.0000	0.9863
Partition 4	0.0082	1.0000	0.9973
Partition 5	0.0028	1.0000	0.9998
Partition 6	0.0016	1.0000	1.0000
Partition 7	0.0012	1.0000	1.0000

Table 2: Train with VGG-11

IV. Exprimental Results

With each partition, a csv file is created and submitted. From that we can analyze the score that Kaggle gives us. It is clear that, from all 3 approaches, PhoBERT performs best, and the other 2 have overfitted the data.

Model	***PhoBERT***	***Resnet-18***	***VGG-11***
Score	0.94	0.89	0.903

V. Conclusion

In this paper, we have presented our approach to Classify Food Reviews using Sentiment Analysis techniques. In the 3 methods that we use, it seems like PhoBERT has taken the W with the score of 0.94. This gives us the 20th ranking in the general leaderboard and top 7 in class INT3405E 41. Even though the score Kaggle gives us is high, there is still some room for improvement.

VI. References

Dat Quoc Nguyen, A. T. (2020, 10 5). PhoBERT: Pre-trained language models for Vietnamese. p. https://arxiv.org/abs/2003.00744.

Kaiming He, X. Z. (2015, 12 10). Deep Residual Learning for Image Recognition. p. https://arxiv.org/abs/1512.03385.

Karen Simonyan, A. Z. (2015, 4 10). Very Deep Convolutional Networks for Large-Scale Image Recognition.