

# Improvement of Prediction Accuracy

## - Text Difficulty Classification -

**Master of Applied Data Science, University of Michigan School of Information**

### **SIADS 696: Milestone II**

Gang Li, Lawrence Fung, Ki-hyun Baik

October 24, 2022

## 1. Executive Summary

By taking the challenge of classifying the difficulty of texts proposed by the faculty team, we examined traditional supervised learning models from Logistic Regression, Multinomial Naive Bayes and various BERT models as well as other unsupervised algorithms including Kmeans, PCA, UMAP, HDBSCAN, and tried to improve the accuracy of classification.

As a result, it is found out:

- Various combinations of traditional supervised learning models from Logistic Regression to Multinomial Naive Bayes were tested via TF-IDF text representation. There were some performances worthy of attention from different tuning approaches and combinations, especially by adjusting the pre-processing stage via LIME analysis, but it was hard to conclude to be successful given the limitation of the TF-IDF text representation stage.
- BERT is a language model pre-trained on an enormous amount of data, which performs very well on natural language processing tasks. When using this model, the majority of the time is spent on fine-tuning the model through its hyperparameters and different BERT models can be chosen depending on the task and resources available.
- UMAP + HDBSCAN makes more sense to reduce the high dimensional data, and visually present the data for intuitive exploration. However, PCA, Kmeans did not show promising results by clustering data.

More importantly, through error analysis and experiments, it is shown that:

- The traditional text processing including removal of stopwords, lemmatization, spelling correction actually impair the performance of traditional supervised learning models and also BERT models.
- Numbers of similar texts from the training datasets and the built-in classification scheme from the pretrained model could be one of reasons to impact the accuracy via sentence-BERT embedding and cosine similarity checking.

Note: given the prediction error analysis from the supervised learning, especially BERT, deserves further analysis, as agreed with the professor, we turn out our focus on the prediction error analysis, and the zero/few shot prediction - one of topics of this project is removed.

## 2. Introduction

The text difficulty problem is one of binary-labeling text classification problems. Normally it can be attributed to the domain of supervised learning.

To approach this problem as well as deepen our understanding on both supervised learning and unsupervised learning, we take the following steps:

1. Pick up the main-stream supervised learning models in terms of both traditional models and deep-learning based models, and identify the best one by comparing the accuracy based on the hyper-parameters recommended by model cards.
2. Fine tuning hyper-parameters of both types, and perform the sensitivity analysis by trying different combinations of hyper-parameters.
3. Perform the error analysis based on the best result from sensitivity analysis, and try to find out the causes and solutions to improve the accuracy.
4. Utilize the unsupervised learning algorithms, especially clustering to try to see whether the clustering could help to perform error analysis or support classification.

The following sections are used to carry on those steps.

## 3. Data Source

The dataset we used is a text difficulty dataset from Wikipedia, which can be found in the following link (<https://www.kaggle.com/t/5a1872e494574cc7bbf433fa8f4687d9>). The datasets came in the form of CSV files, in which the variables used for our project were the **original text** and **label**. We mainly used the “WikiLarge\_Train.csv” file for training purposes, where we used all records for our traditional supervised learning method. Due to time and cost we used 10% of the dataset for training and evaluating our supervised deep learning method using BERT models. And for unsupervised learning we used all records.


The target labels in the dataset can be defined as the following:

- 0 - text does NOT need to be simplified
- 1 - text needs simplification

## 4. Part A. Supervised learning

### 4.1 Traditional Supervised Learning

#### 4.1.1 Motivation

The general motivation of reviewing the traditional supervised learning method is to provide insight on whether it's necessary to use complex models over simple, but yet to be effective enough with comparative advantages by remaining 'simple', models and to maneuver on ways to improve. Under such awareness, three classification methods were tested in combination via TF-IDF text representation; Logistic Regression, Multinomial Naive Bayes, and Linear SVC. TF-IDF was selected especially above simple Bag-of-Words as it could compensate for the rarity  oblem, with slight advancement.

## 4.1.2. Dataset Analysis and Model Performance

### 4.1.2.1 Training dataset analysis and tuning

The analysis and tuning pipeline has been conducted as the following:

1. Basic analysis on the unique characteristics of the dataset  
The most notable characteristic that was identified through a rough EDA was that the original text had many duplicates with different labels. As this could significantly hamper the performance on classification, duplicates had been totally removed.
2. Text cleaning, lemmatization, stemming  
As the target of classification was to distinguish between whether an original text needs to be simplified or not, text cleaning, especially stopwords removal, was tested as one of the most important tuning features to see whether it could be a major contributing factor in the classification task.
3. Performance evaluation  
Through the vectorized TF-IDF text representation dataset, classifiers (Logistic Regression, Multinomial Naive Bayes, Linear SVC) were tuned, trained, and tested in various combinations of parameters. The result was evaluated via mean accuracy across 5 cross-validation folds, precision, recall, and f1-score.
4. LIME analysis  
Based on the classified results, LIME analysis has been performed to identify how individual text data and its words have contributed to make correct or incorrect classifications. Based on the analysis, the dataset was refined and re-fitted through multiple iterations aiming for improvement.

### 4.1.2.2 Performance comparison on text classification

1. Classification performance on various combinations of classification models  
For each classifier, the optimal-expected parameter values were found through GridSearchCV, with 5 cross-validation folds, and have been compared with the default settings. Considering the computational cost, only a few parameters were chosen to be searched. However, with higher tolerance for computational cost and time, more parameters led to higher performance in most cases.

[Table 4.1.2-1. Summary of Classification Report : Default vs GridSearchCV]

Classification Method		Accuracy	Precision	Recall	F1-Score
LogisticRegression	Default	0.6629	0.6633	0.6628	0.6626
	Optimized	0.6733	0.6743	0.6732	0.6728
MultinomialNB	Default	0.5406	0.5421	0.5404	0.5357
	Optimized	0.6044	0.6579	0.6039	0.5672
LinearSVC	Default	0.6031	0.6032	0.6031	0.6030
	Optimized	0.6642	0.6646	0.6641	0.6639

- a. Logistic regression : The best performing parameter was `LogisticRegression(C=0.1, multi_class='auto', penalty='l2')`. The higher the C value gets, the decision boundary tended to rely more on the train dataset and decreased in performance.
- b. MultinomialNB : The best performing parameter was `MultinomialNB(fit_prior=True, alpha=100)`. Lower alpha value gets, the more under-fitted the model turned due to the increase in complexity of outliers.
- c. LinearSVC : The best performing parameter was `LinearSVC(C=0.1, multi_class='ovr', penalty='l2')`. Showed a similar tendency as logistic regression classifier.

## 2. Pre-processing methods performance comparison

Text cleaning was tested on various grounds through the above optimized classifiers. Combinations of different approaches through RegEx and basic stopwords removal, lower casing words, expanding contractions, and stemming and lemmatization. While stemming and lemmatization didn't have either positive or negative impact on the overall performance, stopwords removal actually led to a notable decrease in accuracy, which is a rare case in many other text classification tasks. This could have happened due to the unique nature of this specific task : to classify whether a text needs to be 'simplified' or not. Stopwords per se could be one of the factors that prolongs a text, thus removing it could bring more confusion to the remaining words in the decision making process of a classifier.

[Table 4.1.2-2. Summary of accuracy : Pre-processing methods]

Classification Method	Stopwords	Stemming	Lemmatization	All
<b>LogisticRegression</b>	0.6424(-0.030)	0.6738	0.6732	0.6494
<b>MultinomialNB</b>	0.6098(-0.005)	0.6024	0.6058	0.6100
<b>LinearSVC</b>	0.6396(-0.025)	0.6677	0.6658	0.6397

## 3. LIME analysis and stopwords


LIME analysis was used to identify the contributing factor of correct and incorrect classifications. By selecting a random sample of 100~500 misclassified predictions, four types of texts were sorted from each sample; true-positive, true-negative, false-positive, and false-negative. Top 20~30 false-positive and negative words with prediction probability lower than  $< +/-0.2 \sim 0.4$  (depending on which classification category they fall into : 1 being +, 0 being -) were added to the list of stopwords to see whether performance can be increased by removing these words leading to false labeling. Among the False-factor-word list, overlapping words from the top 20~30 true-positives and negatives have been discarded since they are overlappers in both categories. The result showed a slight increase in performance by the increase in sample size, by 0.01~0.02 for accuracy in the case of 500 misclassified prediction samples, but failed to increase when more iterations were done on the dataset cleaned based on the previous result. This could be due to the fact that as more

stopwords are added through LIME analysis, the probability of remaining words being 'forced' to be mis-labeled could increase at some point. Rather greatly increasing the number of basic misclassified prediction samples from the beginning and conducting LIME analysis a single time looked more promising as more mislabel-leading words can be captured while not harming positive predicting words that can be lost when the analysis is iterated many times.

[Table 4.1.2-3. Summary of Classification Report : Post-LIME Analysis]

Classification Method	Accuracy	Precision	Recall	F1-Score
LogisticRegression	0.6956(+0.022)	0.6949	0.6941	0.6940
MultinomialNB	0.6041(-0.0003)	0.6565	0.6029	0.5669
LinearSVC	0.6769(+0.013)	0.6778	0.6751	0.6750

#### 4.1.2.3. Error analysis and improvement approaches

Traditional supervised learning methods showed some performance considering its simplicity but also proved to have limitations. Although there were some aspects worthy of attention for further development, like in the case of LIME analysis and data cleaning, it generally failed to show significant potential for improvement of performance. Rather, TF-IDF weighting in the data pre-processing stage seemed to be a powerful forger in the overall decision making process. Considering the basics of TF-IDF, as it doesn't take into account the order of the words and belongs therefore to the family of BoW models, there are limitations to conduct classification on long-sentenced texts consisting of ordered words. Relatedly, making no use of semantic similarities could be contributing as a negative factor as well. The importance of 'information' being fed to supervised learning methods seemed to outweigh the shape of the methods per se. To overcome such challenges, various types of pre-processing methods can be tried in more depth and experimenting with LIME analysis with a much greater sized sample could be a promising option to consider, which was limited for this project considering the cost and time. 

## 4.2 BERT-based Supervised Learning

### 4.2.1 Why BERT

As representative of the transformer-based pre-trained language model (PLM), BERT and its various developed models have achieved outstanding performances in various NLP tasks including text classification. Given its success, tremendous wrappers have been developed to facilitate the studies, research as well as business applications. That motivates us to take BERT to:

- Compare its performance with traditional ones to understand its technical advantages.
- Examine the development of BERT family and understand the evolving of its technologies
- Gain the sense of its disadvantages and pave the way to contribute its development as well as applications in the future.

### 4.2.2 Performance comparison of popular BERT models

In this project, we explored three different BERT models, BERT base model (uncased)<sup>1</sup>, BERT large model (uncased)<sup>2</sup>, and DistilBERT<sup>3</sup>.

BERT base model (uncased)<sup>1</sup> is a BERT model that has 12 layers of transformers block, a hidden size of 768, 12 attention heads and 110M trainable parameters. BERT large model (uncased)<sup>1</sup> has 24 layers of transformers block, a hidden size of 1024, 16 attention heads and 340M trainable parameters. DistilBERT<sup>2</sup> is a faster and cheaper model based on BERT base (uncased) that uses knowledge distillation in the pre-training phase. Knowledge distillation is the process of transferring knowledge from a larger model (BERT base) to a smaller model (DistilBERT).

In our case, we treated the BERT base model as the benchmark BERT model while we compared the performance of the BERT large model versus DistilBERT. As seen in the table below, we found that BERT large had the highest prediction accuracy with DistilBERT being slightly less accurate and BERT base performing the worst.

Model Name	Precision	Recall	Accuracy
BERT-base-uncased	0.7262	0.7208	0.7196
BERT-large-uncased	0.7355	0.7347	0.7346
DistilBERT	0.7283	0.7275	0.7277

However, we discovered several advantages and disadvantages to using BERT large and DistilBERT. Due to the number of trainable parameters that BERT large has, it will have the highest prediction accuracy. However, due to its size, training and using the model for inference requires significantly more time and resources. DistilBERT is a smaller version of BERT base so it takes significantly shorter time to train and perform inferences.

For example, for our dataset, we trained our model using a machine with 1 CPU, 1 GPU, and 32GB of memory. Training a BERT large model on the whole dataset took 4 to 5 hours while training a Distilbert model took about 1 hour. Additionally, BERT large had a much higher demand for memory. Thus, there is a tradeoff between speed and cost versus accuracy when choosing to use BERT large or Distilbert.

### 4.2.3 Improvements of BERT on text classification

#### 4.2.3.1 Training data analysis

Through exploratory data analysis we found that the maximum number of “words” in one record of text was **80**. Overall, there were **169,658 unique “words”**, with **44,695 “words”** occurring once within the whole training text. What was interesting was that **16.65%** of the records were duplicates.

Figure 4.2.3 - 1 in the appendix, we show a right-skewed distribution of sentence lengths with the average length being around 21 characters and a maximum length of 80 characters. And about 90% of the sentences in the training set have less than 40 characters.

Figure 4.2.3 - 2 in the appendix, we show two distributions of sentence lengths for each target label. They appear to be similar, but sentences that tend to be longer are more likely to be classified as "text that needs to be simplified".

#### 4.2.3.2 Choices of hyperparameters

We found the most important step when using any BERT model that affects accuracy was fine-tuning through the hyperparameters. Despite the BERT large model (uncased) having the best accuracy, the biggest downside as mentioned earlier is that the model is costly and takes a significant amount of time to train. Hence, we used DistilBERT to perform a sensitivity analysis of three hyperparameters, batch size, training epochs, and learning rate.

Firstly, there were a few hyperparameters that we set and did not change. The max sequence length, which is the number of tokens the BERT model can take as input, was set to 128. This number provided a balance between accuracy and required resources. This parameter directly correlates with the memory usage, hence if we increased the value it would require more memory and take more time to train the model. One other parameter we did not change was the warmup steps, set at 600. The idea of this parameter is to allow the optimizer to compute correct statistics of the gradients.

According to Google's research paper<sup>4</sup> of BERT, the following hyperparameter values were found to work for most tasks:

- Batch size: 16, 32
- Learning Rates: 5e-5, 3e-5, 2e-5, (additionally 1e-5)
- Number of epochs: 2, 3, 4

Through the various tests, we can see the sensitivity of hyperparameters in the Figure 4.2.3 - 2 in the appendix.

As we were using the AdamW optimizer, a higher learning rate would have a larger negative impact on the accuracy. The AdamW optimizer is a variant of the Adam optimizer with the implementation of weight decay, which is used for regularization to prevent overfitting. Additionally, implementing the AdamW optimizer sped up training time. Thus, we observed an up to 2% decrease in accuracy between learning rates of 1e-5 and 5e-5.

Batch size had a much smaller impact on accuracy than learning rate, in which we observed an up to 1% decrease in accuracy. Finally, the number of training epochs affected accuracy the least of the three hyperparameters, where we observed an up to 0.8% decrease in accuracy. Thus, we can conclude that in fine-tuning the BERT model, the most important hyperparameter was learning rate.

#### 4.2.3.3 Error Analysis Using LIME

While evaluating the model, we found that some of the predictions made were incorrect. Using LIME, we attempted to analyze a few of these predictions. In Figure 4.2.3 - 4 below, we can see that the word "whole" had the highest importance in making this prediction. However, by moving the word to another section of the sentence, such as after "the"



produces the correct prediction with 62% probability. Removing the word increases this probability to 81%. For this case, we may assume that grammar and spelling may affect the accuracy. From this, we can identify how these prediction errors occur and think about how to remedy this issue. Further steps can include extra preprocessing steps involving spelling and grammar check.

Visualization For Score				
Legend: <span style="color: red;">■</span> Negative <span style="color: gray;">■</span> Neutral <span style="color: green;">■</span> Positive				
True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
1	0 (0.68)	Whole the area is uninhabited , surrounded by rocky mountains , having an elevation of 16000 to 17000 feet .	1.41	[CLS] whole the area is uninhabited , surrounded by rocky mountains , having an elevation of 1600 ##0 to 1700 ##0 feet . [SEP]

Figure 4.2.3 - 4

In Figure 4.2.3 - 5 below, we see that the word “calculated” had the highest importance in the prediction. By changing the word to “found” we were able to match the true prediction. We can infer that word choice may play a role in text difficulty classification. From this, we may consider the next steps to be using a larger set of data.

Visualization For Score				
Legend: <span style="color: red;">■</span> Negative <span style="color: gray;">■</span> Neutral <span style="color: green;">■</span> Positive				
True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
0	1 (0.41)	In 1939 , Robert Oppenheimer and H. Snyder calculated that a star would have to be at least three times as massive as the sun to form a black hole .	0.64	[CLS] in 1939 , robert op ##pen ##heimer and h . snyder calculated that a star would have to be at least three times as massive as the sun to form a black hole . [SEP]

Figure 4.2.3 - 5

In Figure 4.2.3 - 6 below, the word “deuterocanonical” was highlighted as a word with high importance to the prediction result. The word “deuterocanonical” and the following word in the sentence “apocryphal” are the same where the latter is the protestant term. By removing “deuterocanonical”, we were able to obtain the correct label. This error may occur due to a low occurrence of a rare and complicated word. As mentioned in the previous example, we may increase the size of the training dataset and build a larger vocabulary for the model.

Visualization For Score				
Legend: <span style="color: red;">■</span> Negative <span style="color: gray;">■</span> Neutral <span style="color: green;">■</span> Positive				
True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
0	1 (0.36)	The Book of Baruch , sometimes called 1 Baruch , is a deuterocanonical or apocryphal book of the Bible .	-0.33	[CLS] the book of bar ##uch , sometimes called 1 bar ##uch , is a de ##uter ##oca ##non ##ical or ap ##oc ##ry ##pha ##l book of the bible . [SEP]

Figure 4.2.3 - 6

#### 4.2.3.4 Additional Studies - Impact of cleaning text

In the traditional text classification algorithm, preprocessing text is a very crucial step, which could heavily impact the accuracy of classification results. BERT as the new text embedding way as well as PLM, it is contextual based language model, the traditional preprocessing could negatively or hardly impact the accuracy of classification\*.

We picked up several traditional key preprocessing process:

- Removal of stopwords.
- Removal of punctuation.



- Spelling correction
- Lemmatization

Without any exception, the result does reflect the research of the aforementioned papers.

Furthermore, we also detected the records who have the 2 different labels, apparently, removing those could impact the accuracy. However, to our surprise, removal of duplicate records with the same label also shows a negative impact. The reason could be, the duplicate records may increase the weight of correctly-labeling sentences as well as chances of correct classification.

The comparison of accuracy is as below:

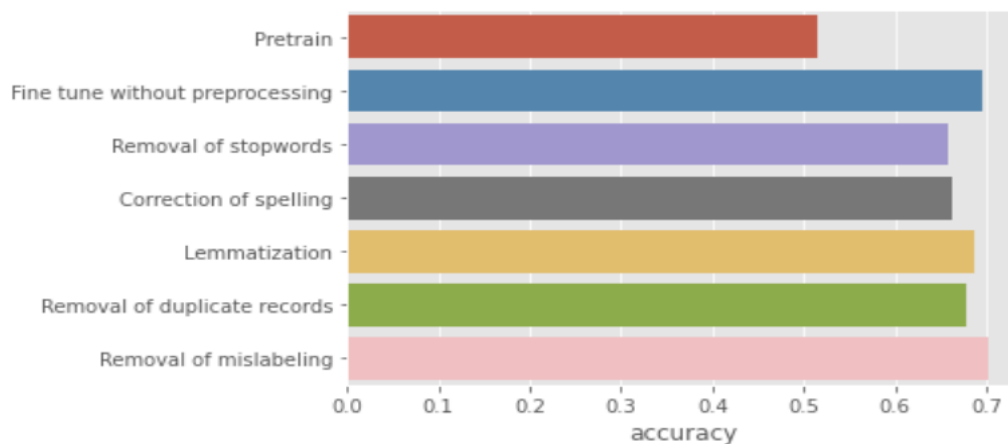


Fig 4.2 - 1. The comparison of accuracy based on the preprocessing steps.

Note: Pretrain means, without any training, and evaluating results on test dataset.

## 5. Part B. Unsupervised Learning


### 5.1 Motivation

As one of typical categories of unsupervised learning, clustering (“2.3. Clustering — scikit-learn 1.1.2 documentation”) algorithms could be an effective way to give the overall sense of data categories under the given features. This motivates us to utilize the clustering algorithms to analyze the error data set from supervised learning results, more specifically from BERT classification results, and try to:

- Differentiate the error data set from the correct ones to identify the characteristics of error data, then:
- Clustering the different error types from the error data to try to identify the causes.

K-means is a widely used clustering algorithm given it

- Relatively simple to implement.
- Scales to large data sets.
- Guarantees convergence.

However, as the number of dimensions increases, it could not make the appropriate choice of an appropriate size of locality for density estimation, consequently, it could not resolve clustering (r Molc  ov #). Given the classification result is the product of deep learning,

and normally, as one of key components of this product, the embedding of text produced by BERT is used as the features of clustering. In our work, sentence-BERT (Reimers and Gurevych) is introduced for this purpose. Since the text embeddings is high dimensional vectors, in order to visualize the clustering results as well as avoid the curse of dimensionality (r Molchanov #), PCA is introduced to reduce the high dimensionality for Kmeans.

## 5.2 Data Source

The dataset we used is the subset of the random sampling for the classification of trained BERT models based on the data set mentioned by section 3. There are 2 dataframes, in which the usage and the variables used for our project were:

df\_train - training dataset for BERT model:

- Original text: original text
- Label: correct label for the text.
- Err: all are 0. It is only used to combine the data frames.

df\_test - test dataset for BERT model:

- Original text: original text
- Label: correct label for the text.
- Err: this record is classified correctly or not. 1 means WRONG, 0 means Correct.

## 5.3 Unsupervised learning methods

As shown in the right figure, basically, the process of unsupervised learning is:

1. Combine both train and test set as a whole dataset with identifying those records with wrong prediction results (variable: err), and create the text embedding by sentence-BERT
2. Reduce the dimensionality of embedding to 2-dimensions by PCA.
3. By the elbow method ("Elbow method (clustering)") to find out the optimal no. of clusters ( K clusters), then run Kmeans to cluster the PCA results

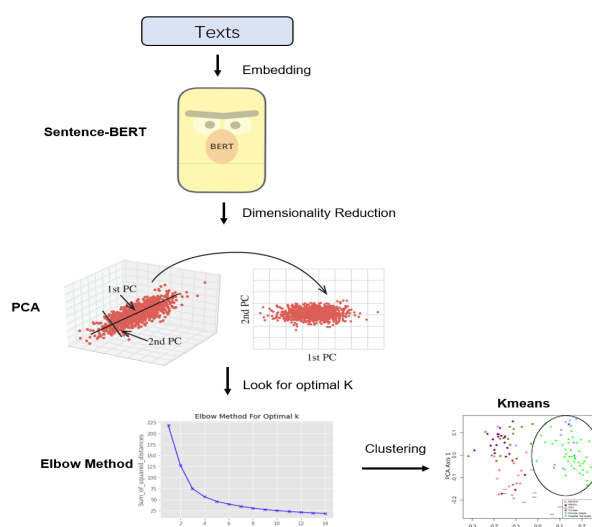



Fig 5.3 - 1 The process to unsupervised learning

## 5.4 Unsupervised evaluation

As the result of the elbow method, it looks like 2 is the optimal K ( no. of clusters), and the outcome of K-means could not help to give the hint on the pattern of error data from the supervised learning ( the section 4.  ). Furthermore, as the sample checks, the points closed to each other look like they are not relevant. Please see the following figures.

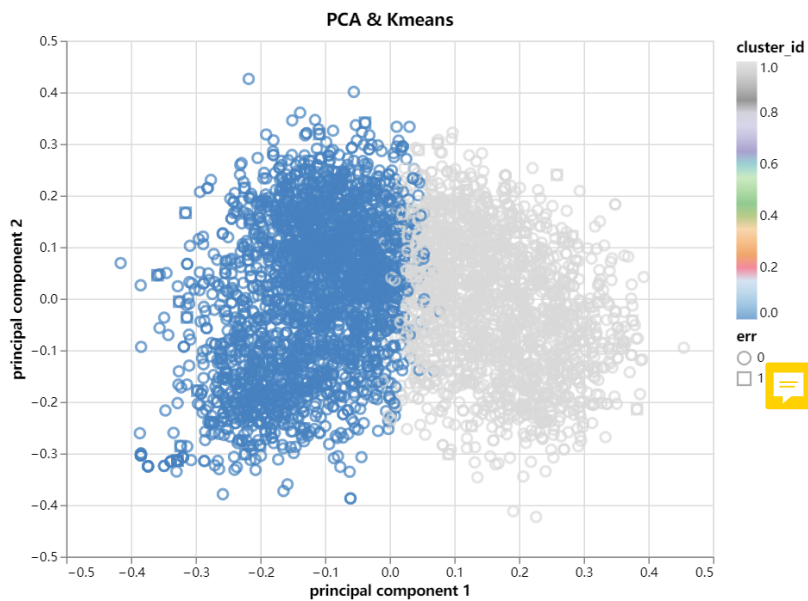


Fig 5.4 - 1: The plot of PCA & Kmeans result. “Blue” for cluster 0, and “gray” for cluster 1. Err:0” means correct prediction from supervised learning, and “Err:1” means wrong prediction.

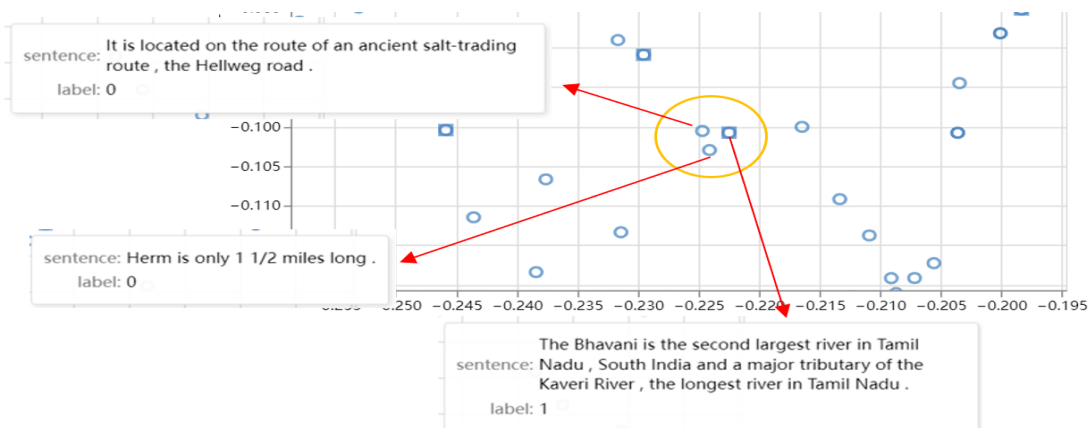


Fig 5.4 - 2: The sample check result looks like the closed points are not similar.

Based on the research, UMAP + HDBSCAN may help to improve the clustering by focusing more on the global structure of the dataset. (Portals).

As the result of the experiment, even UMAP + HDBSCAN could not help to distinguish the error records from others, however, it could provide fine-grained clusters, in which the points close to each other look like the similarity of some levels. It can be shown in the following figures.

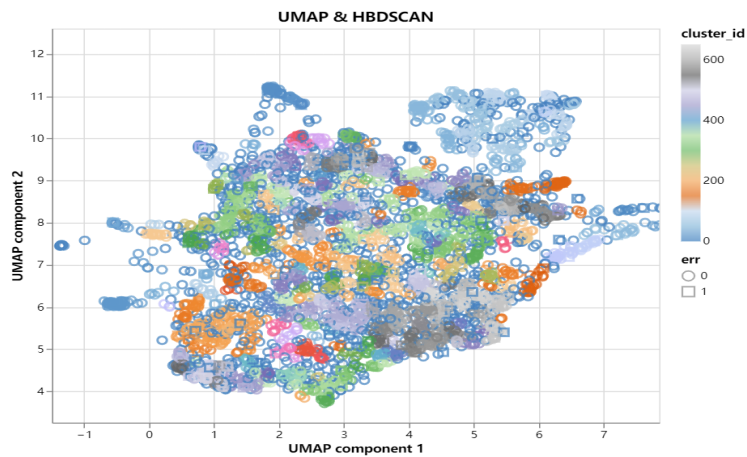


Fig 5.4 - 3: The plot of UMAP & HDBSCAN

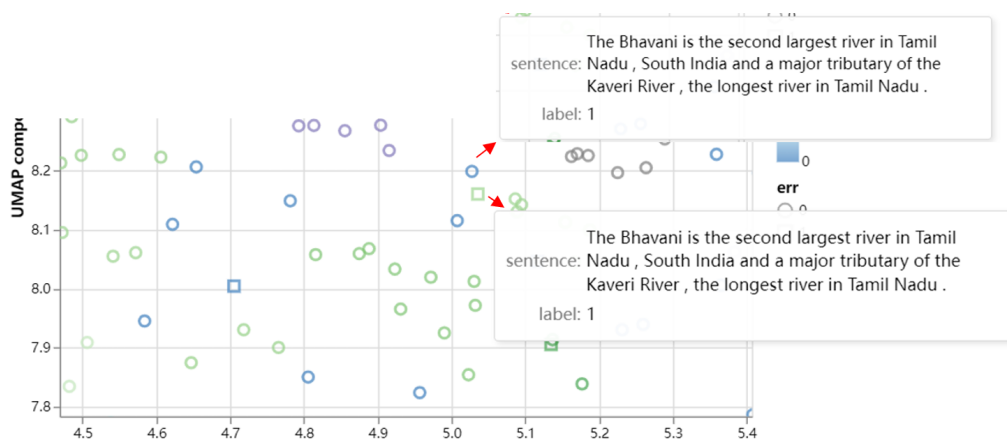


Fig 5.4 - 4: The same sample, but the close points looks like those are same, but with different label.

Through the above sample check from UMAP & HDBSCAN, that reveals one direction - why the above same sentences have the different prediction result. That could be something that needs to be investigated in the BERT model. The further detail will be discussed in section 6.1.

## 6. Discussion

### 6.1 Key learnings & next steps from supervised learning

Traditional supervised learning methods are still a robust method considering their simplicity and less computational complexity. However, despite the difference in classification models, data preprocessing turned out to be the single most contributing factor in increasing or decreasing performance. Although the limitations of TF-IDF was able to be mitigated to a certain extent through further data cleaning and LIME analysis, testing alternative text representation techniques, and/or developing LIME analysis in more depth, would be conducive in identifying the best machine learning pipeline for this given task.

Using BERT for supervised learning is very effective as an improvement to the traditional supervised learning methods. As with many deep learning models, BERT requires much more resources, be it time, memory, or computational power. As mentioned in the previous paragraph, data preprocessing can widely affect performance. However, with deep learning models another important factor to be aware of are the hyperparameters. After the optimal hyperparameters are defined, further improvement to the performance will mainly rely on improving the preprocessing steps. These steps can be added through a LIME analysis of correct and incorrect predictions. Further steps to improve performance may include exploring more evolved BERT models, such as RoBERTa and DeBERTa. Additionally, with the amount of resources required to train BERT models, as mentioned before, we could only use a fraction of the whole dataset. So if there were more resources available, we could also train the various models with the whole dataset.

Based on the evaluation result of DistilBERT on test dataset, through picking up several samples as well as, here are several key findings:

1. The classification results of similar sentences ( be concluded from cosine similarity on the sentence-BERT embedding) can directly impact the classification of this sentence. For example, if the classification results of 3 similar sentences are 1, another similar sentence's result is 0, the 4th sentence's result will be 1 instead of 0. That could be one of potential mislabeling issues.

Situation 1: Correct classification on Pretrain BERT

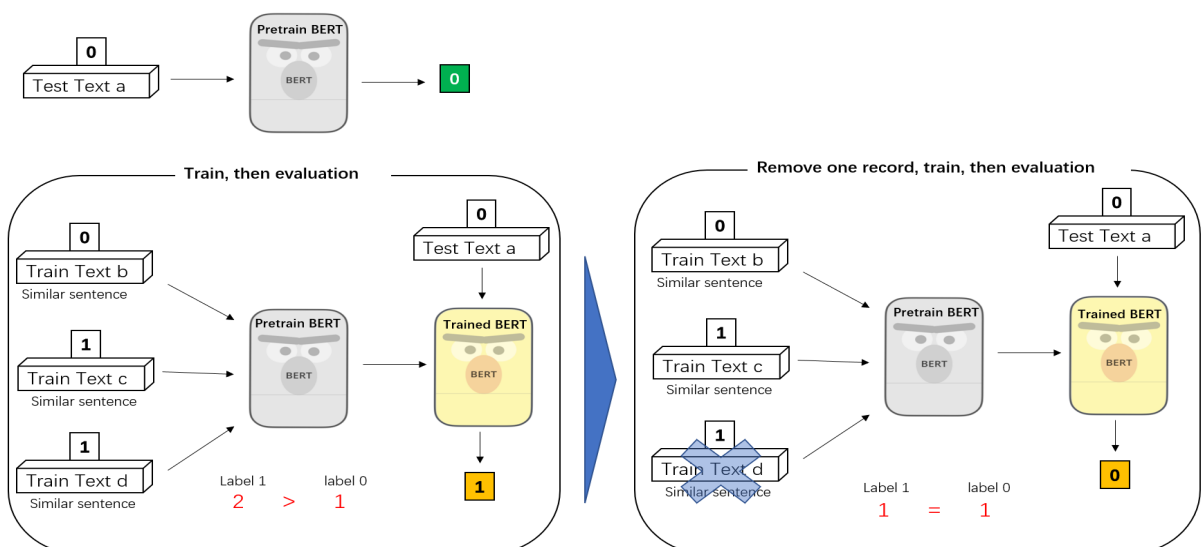


Fig 6.1 - 1 Potential labeling impact

The detailed sample can be found in the appendix.

2. The built-in classification scheme in the pretrain model ( without training) can directly impact the classification of this sentence as well.

Situation 2: Wrong classification on Pretrain BERT

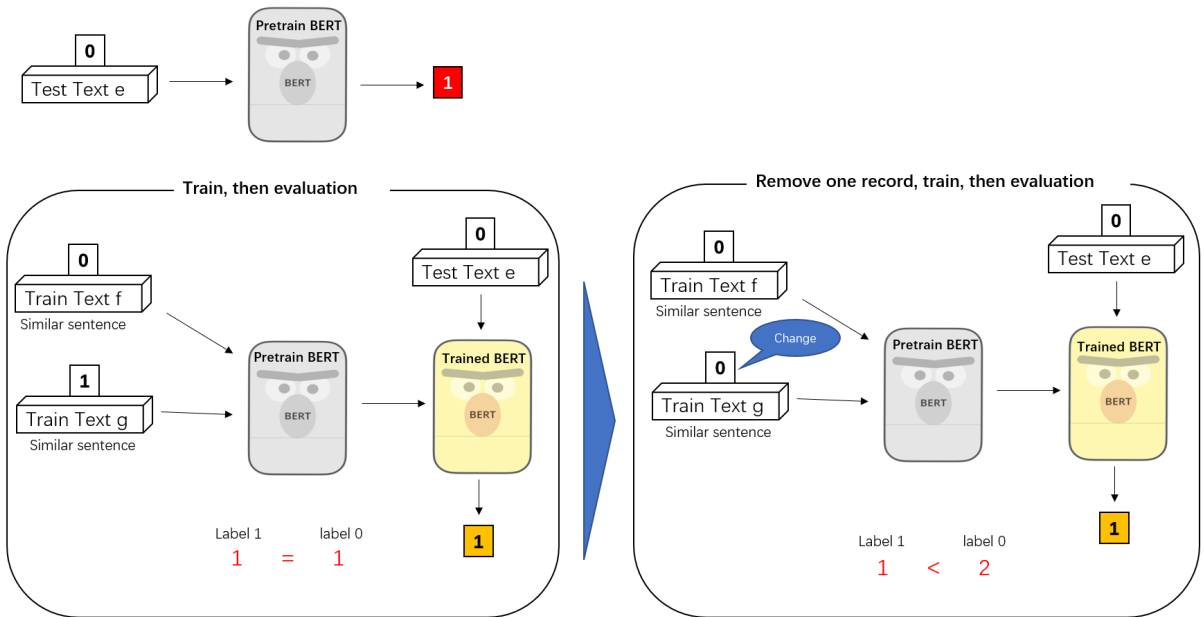


Fig 6.1 - 2 Impact from pretrain model

The detailed sample can be found in the appendix.

Those findings inspire us to dig into the further understanding on the process of BERT classification, and move toward the following possible approach to address the problems:

1. Identify errors with 2 types: one is pretrain result is correct, another is not.
2. Make sentence-BERT based embedding, but set 0.5 or above as similarity threshold(cosine similarity's measure result is 0 to 1: 0 means totally different, 1 means almost same), programmably cluster sentences whose similarity is above threshold.
3. Separate the results into 2 types mentioned in 1.

Basically, this approach could give the sense on how the error could be raised from the given training data and the pretrain model. For the related coding, please see the appendix.

Furthermore, the above approach is purely based on the BERT-based embedding of text, in fact, this base is only one of the fundamental layers of the classification scheme in BERT. In order to measure the impact from the label as well as the pretrain model, it should look further into the scheme of BERT. There are some papers which are looking for the ways to identify the similar issue, such as "Identifying Mislabelled Data using the Area Under the Margin Ranking" (Pleiss) is one of papers recommended by professor, which measures the logits before softmax layer during training, and proposes the approach to identify the mislabelling.

Going forward, if we could work out one tool to perform the early check on the labeling issues as well as implication of the pretrain model, that may help the practitioners to look for more ways to improve the accuracy of prediction. That could be one of further studies from our team in the future.

## 6.2 Key learnings & next steps from unsupervised learning

### UMAP + HDBSCAN on classification

UMAP + HDBSCAN does provide the hints to move toward the similarity check between error predictions and training set. However, how to choose the key parameters such as `n_neighbors` from UMAP, `min_cluster_size` from HDBSCAN, is still not clear for us. Given there is possibility to utilize this approach to identify the characteristics of error records in the overall level, consequently, it is worth studying further.

### Performance metrics of clustering algorithms

Although the elbow method does not work in our application, it is important to know the performance metrics on unsupervised learning given there is lack of ground truth labeling in most cases. That could be greatly helpful to make the best decision to choose the right algorithms to assist error analysis from the classification problem, especially in overall level.

Silhouette score (Henrique), or another quantitative discriminant method (Shi) are proposed to measure the performance of clustering algorithms. That should be worth studying further to look for the appropriate algorithms in the future.

## 6.3 Ethical Consideration

Judging the difficulty of texts has the subject nature. That means, there is no “ground truth” on the difficulty of the text. Also the factors contributing to text difficulty vary depending on the first language and proficiency level of a reader. Thus, labeling texts as simple or not, should address what “simple” means as well as the detailed criteria to label data.

Given the possible usage of the classification on text difficulty could be educational, it requires to specify the target audiences precisely in order to avoid the potential harm on the inapplicable audiences.

Furthermore, since the dataset is most likely used to train the models, and it could have some wrong labeling, the owners of dataset should keep the transparency on how the quality of training data is ensured, and what problems there would be, and what solutions could be taken to fix those problems.

Also there is no perfect model, it should make clear the limitations and drawbacks of the models to be utilized, and communicate the impact with the target audiences.



## References

1. "Bert-Base-Uncased · Hugging Face." *Bert-Base-Uncased · Hugging Face*, <https://huggingface.co/bert-base-uncased?text=Paris%2Bis%2Bthe%2B%5BMASK%5D%2Bof%2BFrance>.
2. "Bert-Large-Uncased · Hugging Face." *Bert-Large-Uncased · Hugging Face*, <https://huggingface.co/bert-large-uncased?text=Paris%2Bis%2Bthe%2B%5BMASK%5D%2Bof%2BFrance>.
3. "Distilbert." *DistilBERT*, [https://huggingface.co/docs/transformers/model\\_doc/distilbert](https://huggingface.co/docs/transformers/model_doc/distilbert).
4. Google-Research. "Google-Research/Bert: Tensorflow Code and Pre-Trained Models for Bert." *GitHub*, <https://github.com/google-research/bert>.
5. Nlptown. "Nlptown/NLP-Notebooks: A Collection of Notebooks for Natural Language Processing from NLP Town." *GitHub*, <https://github.com/nlptown/nlp-notebooks>.
6. "Captum · Model Interpretability for Pytorch." *Captum*, [https://captum.ai/tutorials/Bert\\_SQUAD\\_Interpret](https://captum.ai/tutorials/Bert_SQUAD_Interpret).
7. "Elbow method (clustering)." *Wikipedia*, [https://en.wikipedia.org/wiki/Elbow\\_method\\_\(clustering\)](https://en.wikipedia.org/wiki/Elbow_method_(clustering)). Accessed 12 September 2022.
8. Reimers, Nils, and Iryna Gurevych. "[1908.10084] Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." *arXiv*, 27 August 2019, <https://arxiv.org/abs/1908.10084>. Accessed 23 October 2022.
9. r Molchanov, Vladimi. "Overcoming the Curse of Dimensionality When Clustering Multivariate Volume Data." *SciTePress Digital Library*, vol. 1, no. 1, 2018, p. 11.
10. "2.3. Clustering — scikit-learn 1.1.2 documentation." *Scikit-learn*, <https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation>. Accessed 12 September 2022.
11. Pleiss, Geoff. "Identifying Mislabeled Data using the Area Under the Margin Ranking." *Identifying Mislabeled Data using the Area Under the Margin Ranking*,

## Appendix

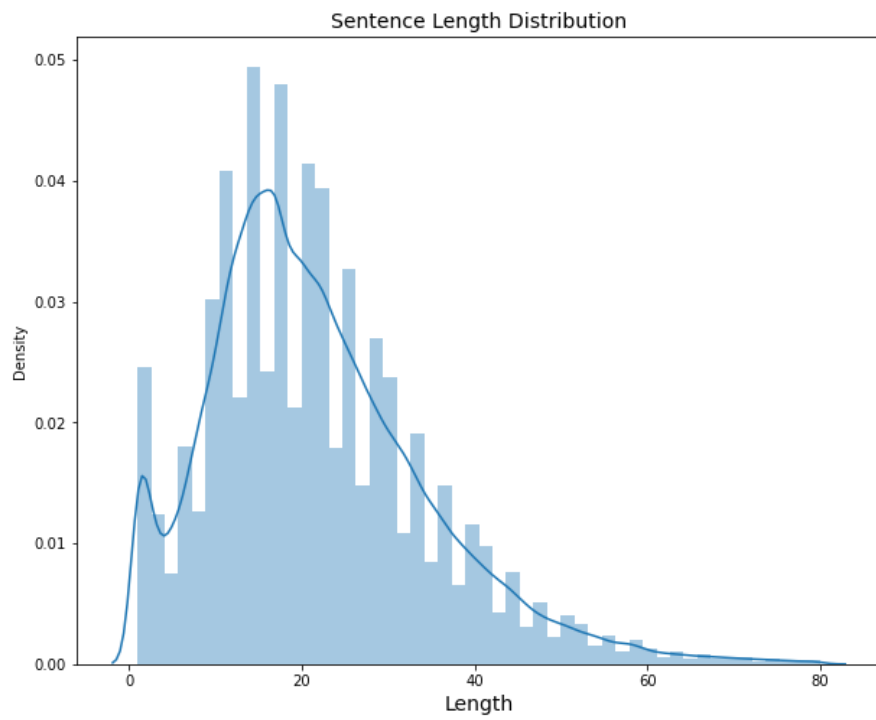


Figure 4.2.3 - 1

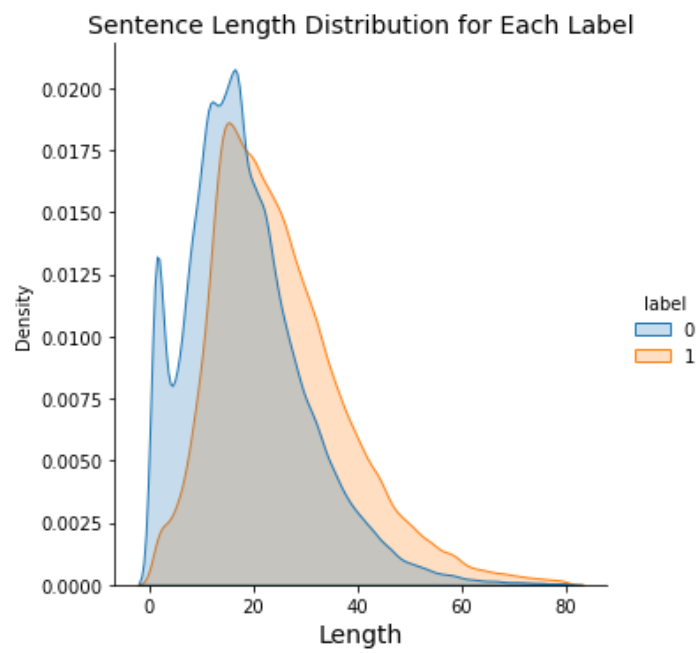


Figure 4.2.3 - 2

## Hyperparameters Effect on Accuracy

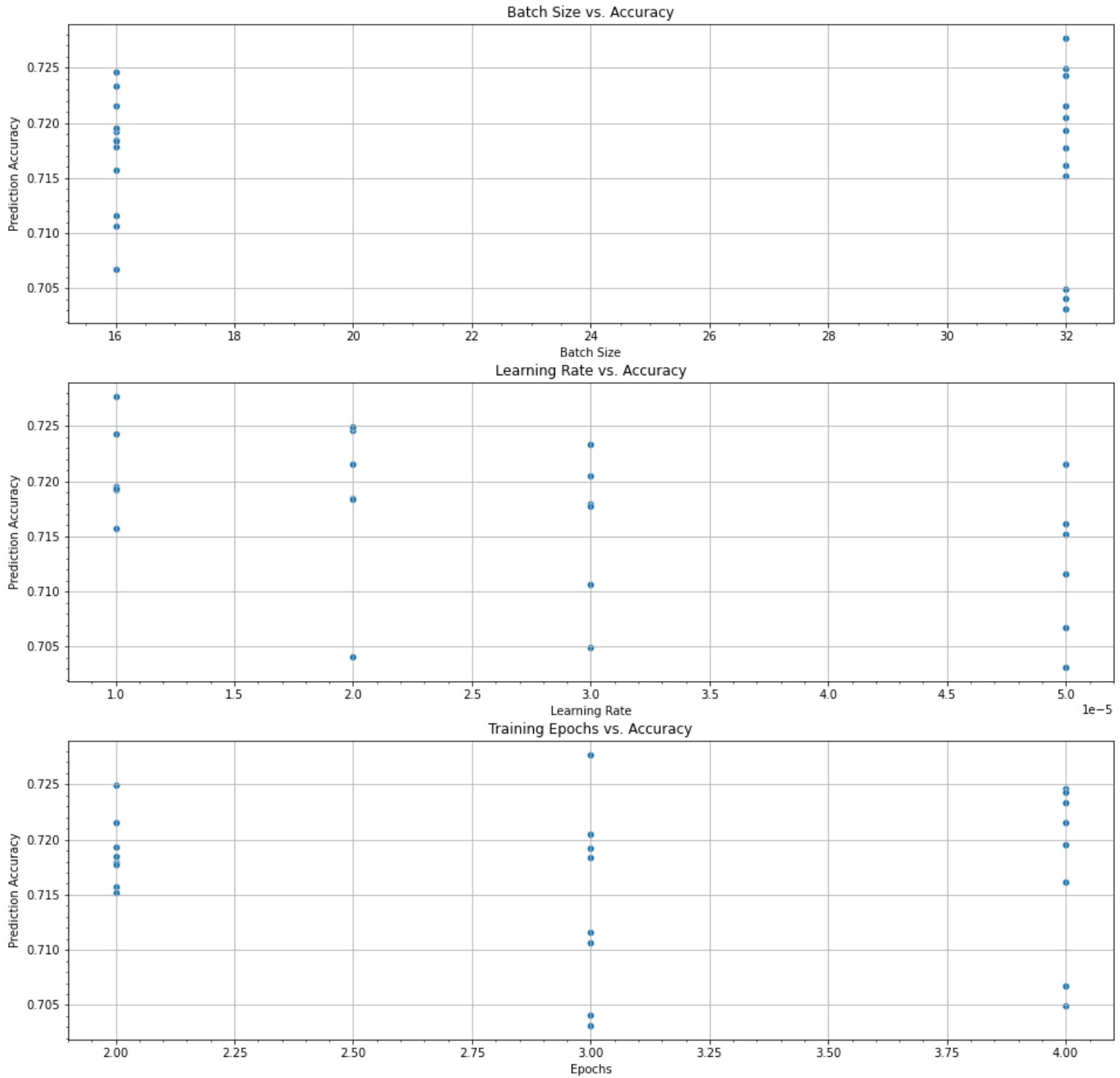


Figure 4.2.3 - 3

Samples of Chapter 6.1:

### Situation 1: Correct classification on Pretrain BERT

Test Sentence

original_text	label	err	init_pred	best_pred
For SV Ried he made his professional debut in ...	0	1	0	1

Prediction by Pretrain model

Ground Truth

Prediction by Fine tuning without preprocessing

Training Sentences

original_text	label	score
He plays Germany national team .	0	0.509770
A playmaker and striker who made his internati...	1	0.505181
On 26 April 2004 Carlsen became a Grandmaster ...	1	0.524805
He currently plays for Kyoto Sanga F.C. .	1	0.517599
He played goalkeeper , in particular with São ...	1	0.510566

Ground Truth

Similarity score in cosine.

Similar sentences

Label 1  
4

>

label 0  
1

### Situation 2: Wrong classification on Pretrain BERT

Test Sentence

original_text	label	err	init_pred	best_pred
16 313 to 399 Emperor Nintoku Oho Sazaki no Mi...	1	1	0	0

Both prediction are wrong.

Training Sentences

original_text	label	score
72 1073 to 1086 Emperor Shirakawa Sadahito Bro...	0	0.524942
1586 & ndash ; Emperor Go-Yozei becomes Empero...	1	0.613717

Label 1  
1

=

label 0  
1

## The snippet of coding in chapter 6.1

```
▼ ## Create the dataset for comparison,

# Create training set
df_comp=pd.DataFrame()
df_comp['original_text']=df_acc[df_acc['description']=='Fine tune without preprocessing']['texts'].values[0][0]
df_comp['label']=df_acc[df_acc['description']=='Fine tune without preprocessing']['labels'].values[0][0]
df_comp=df_comp.reset_index()

# Test dataset
df_test['id']=df_test.index
df_test['id']=df_test['id'].apply(lambda x: 'e'+str(x))
```

Batches: 0% | 0/106 [00:00<?, ?it/s]

```
▼ ## Create the embedding for comparison dataset
query_embeddings=embedder.encode(df_comp['original_text'])
```

```
▼ ## Work out the similarity clusters and identify the record of self
from sklearn.metrics.pairwise import cosine_similarity

# Set the similarity threshold
THRESHOLD=0.5

# Create dataset to cluster the similar sentence
err_cluster=pd.DataFrame(columns=list(df_comp.columns)+['score', 'cluster'])
df_err=df_test[df_test['err']==1]

▼ for i in range(len(df_err)):
    df_temp=pd.DataFrame(columns=err_cluster.columns)
    str_to_predict=df_test[df_test['err']==1]['original_text'].iloc[i]
    # Compare similarity
    sim=cosine_similarity([embedder.encode(str_to_predict)], query_embeddings)
    j=np.argmax(sim)
    sim_rows=list(np.where(sim[0]>THRESHOLD)[0])
    # cluster the sentences in training set by error records
    ▼ if sim_rows !=[]:
        ▼ if err_cluster.empty:
            df_temp=df_comp.iloc[sim_rows]
        ▼ else:
            df_temp=df_comp.iloc[sim_rows]
    ▼ if df_temp.empty==False:
        df_temp['score']=sim[0][sim_rows]
        df_temp['cluster']=df_err['id'].iloc[i]
        err_cluster=pd.concat([err_cluster, df_temp], axis=0, ignore_index=True)
```

## Credit

- In this project, inspired by [yvespeirsmanas's BERT notebook](#), we customized our codes to run the three BERT models as well as making comparisons among preprocessing approaches. In addition, Captum Tutorial<sup>6</sup> for 4.2.3.3 Error Analysis Using LIME