More text mining can enhance the model?

March 26, 2023

1 Abstract

2 Background

Fernandes et al. explored five different machine learning models, Random Forest (RF), Adaptive Boosting (AdaBoost), Support Vector Machine (SVM), K-Nearest Neighbors (KNN) and Naive Bayes (NB), for a classification problem: whether an article has more or less than 1,400 shares. Some variables used in the modeling indicate characteristics of context and structure of writing such as rate of non-stop words, sentiment polarity of title and content, etc. However, there are also variables that describe features of an article that are not related to the text: when they are published, the number of images and videos included in the article, and characteristics of links attached or embedded in the article, etc.

According to the study RF is the best performing model considering five criteria, accuracy, precision, recall, f1, and auc score. Also, based on their final model, variables considered to be "important" were related to the variables that are closely associated with the context.

The study was published back in 2015, when the transformer architecture was not yet introduced. With the transformer architecture, there has been many large language models (LLM) as well as multi-modal models. As Fernandes suggested in the paper, if context seems to be more important than other features of an article, it may seem appropriate to utilize LLM to provide more contextual information to a classification model.

3 Materials and Methods

3.1 Data Preprocessing

The dataset provided by Fernandes through the UCI machine learning repo was fairly clean, but required additional cleaning and pre-processing. For example, "is_weekend" variable was the sum of "weekday_is_saturday" and "weekday_is_sunday". In addition, "n_non_stop_words" was removed as this was also fully overlapped with a set of other variables. In the paper, the response variable in the study was $I(shares > 1400)$ and therefore, this binary response variable was added to the dataset and used for the classification task.

To train a LLM, texts should be collected from the urls. Using the Beautiful Soup python package, only the title and the content of each article was collected. However, due to some difficulties in collecting text data from some web pages due to some unique HTML structure or pages no longer available, 36 articles were dropped out of the final text mining dataset for a LLM. As the LLM selected for this experiment is DistilBERT, the tokenizer used for the model was used to tokenize the texts. For the domain adaptation task, a title and content of each article was concatenated and they were duplicated three times and shuffled. Then it was split by every 800 letters as the maximum token length of DistilBERT is 512.

The dataset was split into training and validate subsets with the ratio 8 to 2. The validate subset will not used for domain adaptation as well as training the final model.

3.2 Large Language Model

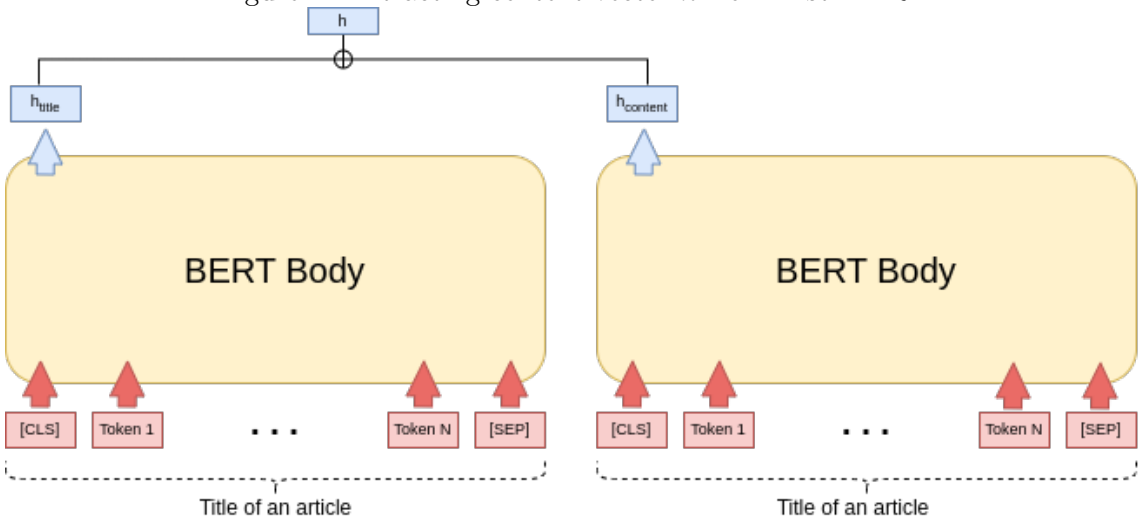
When selecting a LLM to be used for this experiment, there were several limits that led to choose DistilBERT. All the domain adaptation and fine-tuning for classification task must be complete within less than a week to submit the result to the UKY data science competition committee. In addition, a limited compute resource results in choosing a model that is light and requires less pre-training tasks. DistilBERT is a distilled version of the orginal BERT-base model, which has only 80 million parameters but performs similar to BERT.

The articles in the text dataset are all from Mashable, and as seen in several articles, their general tone and writing style are quite different from text resources DistilBERT was pre-trained from: wikipedia articles and book-corpus. Therefore, domain adaptation, which helps a LLM to become familiar with new set of texts, was deemed necessary. The Masked Language Modeling (MLM) was the only method used for the domain adaptation as it was shown that the Causal Language Modeling does not meaningfully improve the model. The training had 3 epochs and the masking probability of 0.15.

3.3 Fine-tuning and final models

After the domain adaptation process, the LLM becomes familiar with the general style of writing and context of texts to be use for tasks such as classification and regression. However, this is not the end. Note that, unlike the MLM task for domain adaptation, title and content of each article were individually considered per the way original dataset from UCI repo is structured. In other words, a hidden state vector for [CLS] token for both title and content, h_{title} and $h_{content}$, are inputs of the three models; the hidden states are concatenated, $h = h_{title} \oplus h_{content}$, and used for a classification model(Figure 1). For convenience, h is called the context vector. The context vector is a vector representation of the summary of a sentence or a sequence of sentences passed into the LLM. The context vector for a given sentence has slightly changed through the domain adaptation process, but this needs further modification in order to address the property of a specific task for this model to be used; this process is called "fine-tuning". However, due to the limited time and compute resource during the competition week, fine-tuning was not

Figure 1: Extracting context vector h from DistilBERT



achievable; therefore, the context vector is simply from the LLM without training for tine-tuning task. Per Fernandes' fitting algorithm, the hyper-parameter for number of trees was to be chosen from 10, 20, 50, 100, 200, 400. However, due to lack of compute resource as well as time, 400 was arbitrarily chosen. In addition, since no further details of the final model used for Fernandes' study, we compared model with and without the context vector. The RF model with the context vector will be compared to the one without the context vector with resepct to not only the metrics such as accuracy, f1, and roc score but also importance scores of variables and their ranks.

4 Result

	accuracy	precision	recall	f1	auc
random forest with context vector	0.619288	0.619715	0.589245	0.604096	0.618869
random forest without context vector	0.633173	0.631759	0.613316	0.622401	0.632897

The result shows that the random forest model with context vector does not do well across the entire metrics. There are several reasons that can explain this result. First of all, the model with context vector may need more trees in the forest model as there are more variables after the context vector is added. When the number of trees for the model with context vector has increased to 1500, the performance has been returned as below.

5 Future Works

With enough compute resource and time in the future, the followings should be done throughout the training process. First of all, domain adaptation process should have more epochs. The training and validation loss were constantly reduced throughout the three epochs in our process, and this suggests more epochs may make the model more akin to Mashable articles. In addition, deep neural network models with the context vectors must be considered. It was tried mainly with huggingface python package, but it tunred out that huggingface was not a right choice for this kind of complicated modeling. When it comes to the random forest models, hyper-parameter should be carefully chosen with a cross-validation method. In this study, the number of trees of 400 is arbitrarily chosen, but for the model with context vector may require more trees than the one without as there are far more variables to consider. Lastly, the response variable can be expanded to a categorical varible with 20 levels by defining he 20-quantiles of "shares".