BBC News Classification Project: Machine Learning and Sequential Model Approach

Introduction

The courses of programme of Computational Linguistics has been ended, we therefore try to integrate some of our learning outcome into this project, not only tthe Machine Learning and Deep Learning methods, but also with some linguistics analysis so that we could benefit from conducting this practice. In this study, we explore using traditional Machine Learning methods and Sequential Model by contextual word embedding by case-sensitive BERT (a pre-trained language model) version to evaluate the performance of different models as our second assignment of Machine Learning module. At first, we gather dataset from Kaggle, conduct EDA, perform a series of linguistics analysis. We also preprocess the text data, including text cleaning, label encoding, label encoding and deduct dimension by UMAP, and one-hot encoding, to compare the performance of different approaches. We train 10 different Machine Learning classifiers by using Grid Search with k-fold Cross-Validation. For Sequential models, we choose Random Search, Hyperband Optimization, and Bayesian optimization with activation relu, elu, selu and leaky relu; for optimizer, we use sgd, rmsprop, adam, adamax and nadam. The result indicates that the Sequential Models have 3% higher accuracy than Machine Learning Methods.

Dataset

We use dataset from Kaggle, but the owner of the dataset does not specify the details about the License, so we further check if the BBC NEWS dataset has been published and need to be cited. We find:
The BBC NEWS dataset has ever been used for Kaggle competition and manually examine if the dataset is the same dataset. The result is that the labels and text content are the same.
We find the original the original BBC News Archive on Kaggle, yet the dataset has not been split into category and text content. It specified “ALL RIGHTS, INCLUDING COPYRIGHT, IN THE CONTENT OF THJE ORIOGINAL ARTICLES ARE OWNED BY BBC.”
We actually download the dataset from BBC News Classification on Kaggle because the dataset has been split into categories and text content, which is fit our learning object.
Therefore, we decide to cite these three resources and use the dataset from bbc-text article on Kaggle.

Methodology

Exploratory data analysis (EDA). There are 2 columns and 2,225 entries. The datatype is object. Surprisingly, we find the unique text is 2126, which is not equals to the total text counts.

	category	text
count	2225	2225
unique	5	2126
top	sport	blind student hears in colour a blind studen...
freq	511	2

Data Preprocessing: We assume the duplicate rows is a strategy for balancing the dataset. But after check the duplicate rows it seems not specific strategy to balance the dataset so we decide to remove 99 duplicate rows. There’s no missing value in this dataset.

```
category
sport          511
business       510
politics       417
tech           401
entertainment  386
Name: count, dtype: int64
```

```
category    0
text        0
dtype: int64
```

Text Cleaning: Since we decide to use case-sensitive BERT version as our embedding methods, so we keep the punctuations and remain case unchanged, only remove html tags, URL, mail box and extra whitespace characters.

Traditional text-preprocessing steps:

The advantage of BERT contextual word piece word embedding method:

- Necessary: Perform BERT WordPiece tokenization because OOV (out of vocabulary) and variations within words could be handled effectively.
- No need for lemmatization or stemming: BERT utilizes the Transformer architecture so BERT learns the words association through attention mechanism so that the model is able to capture the bidirectional contextual information.
- No need for removing punctuation and stopwords: The complete sentences (including punctuation and stopwords) are used as inputs because these elements provide critical clues about the sentence structure and context.
- No need for Specific Part-of-Speech Tagging: BERT learns various word features and potential Part-of-Speech information in a dynamic way.

After text cleaning, we process some linguistics analysis, however, since this is a machine learning project, so we aim to focus on the Machine Learning and Deep Learning methods here and not to explain all the linguistic features in this document. For further information, we suggest to check our .ipynb file directly.

Padding and Truncation:

As we discussed before, BERT is a kind of pre-trained language model and BERT is designed to process sequential data, such as natural language text. However, the length of text could be changeable, yet BERT need a fix input sequences and its max sequences set as 512 tokens (architecture and computing resources), so we need to perform padding and truncation. Since our dataset contains only 2,126 copies of texts, which is not especially a large dataset, so we decide to adopt:

Padding: The max padding 512 for [PAD] marks padded at the end of the sequence to make the length reach 512 for each text, ensuring that the BERT model can handle these shorter sequences.

Truncation: For sequences longer than 512, we truncate to 512 token for the beginning of the text because the begging part of a News article usually contains more important contextual information.

Adopt Stratify Sampling:

As we discussed in the Data Preprocessing section, since the task is classification and the dataset a slightly imbalance, adopting Stratify Sampling could:

Improve accuracy and ensure representativeness: Stratified sampling ensures the representative of sub-samples in the dataset are selected to reduce the variance in the dataset. Whereas 1700 + 468 = 2,126 stands for the total quantity of samples.

```
X_train shape: torch.Size([1700, 512])
X_test shape: torch.Size([426, 512])
y_train shape: (1700,)
y_test shape: (426,)
```

Adopt stratify sampling before apply Label Encoding and BERT Embedding to avoid data leaking.

Label Encoding:

Each category is mapped to an integer value:

```
label 0: category 'business'
label 1: category 'entertainment'
label 2: category 'politics'
label 3: category 'sport'
label 4: category 'tech'
```

One-hot encoding:

Transforms each category to an independent binary vector.

The label encoding could introduce a numerical magnitude relationship so that some classifiers may be affected and these classifiers could misinterpret that sequential relationships between encoded values.
We choose label encoding to transfer categories (objective data type) to numeric representative instead of one-hot encoding because:
For tree models (such as random forest and decision tree) are not sensitive to the numerical magnitude of features and so we apply more tree models for classification.
The trade-off is we don't particularly apply one-hot encoding for regression classifiers, such as Liner Regression and Logistic Regression.

Contextualized word embedding by BERT:

Because BBC NEWS text could include many proper noun, so the case is a critical consideration for processing word embeddings. BERT case sensitive version is a version of BERT-base, so it contains 768 hidden layers, meaning when we use BERT model for embedding, it generates a 768-dimensional vector representation for each token in the input sequence. As the advantage we mention before, we choose BERT as our embedding methods and we examine the embedding shapes, make sure it is (1700, 768) for X train and (468, 768) for X test.

```
(1700, 768)
(426, 768)
```

Dimensional Reduction by UMAP:

BBC NEWS texts contains more non-linear structures and should preserve as rich local structures as possible and we choose BERT embeddings for classifier to capture complex relationships in context of word semantics to train traditional Machine Learning classifiers, so it is necessary to reduce the dimension after perform BERT embeddings.

	UMAP	PCA
Advantages	Preserves local and global structure: beneficial for complex datasets.	Reduce redundancy: reduces noise in the dataset by extracting the most significant features.
	Suitable for non-linear structures.	Easy to understand: principal

		components.
	High Flexibility: can choose hyper parameters to adapt to different data type.	Good computational efficiency.
Disadvantages	More efficiency than techniques such as t-SNE, but high computational cost on large dataset.	Limited to linear relationships and less effective to non-liner structures.
	Sensitive to parameters: need to experiments to find the best settings for datasets.	Sensitive to outliers: affecting the result of principal component analysis.

Suitable Tasks for these 11 Machine Learning Classifiers:

	Classification	Regression	Both
Logistic Regression	V		
Kernel SVM	V		
Softmax Regression	V		
k-Nearest Neighbors	V		
Decsision Tree	V		V
Random Forest	V		V
Gradient Tree Boosting		V	V
XGBoost		V	V
ADABoost	V		
Binary Relevance	V		
Gaussian Naive Bayes	V		

Reason we choose these 11 Machine Learning Classifiers:

Logistic Regression:

Key: Suitable for binary and multi-class classification.

- The parameter 'C: strength of regularization; smaller values result in stronger regularization can help to prevent overfitting.
- The parameter 'penalty' uses L2 regularization, suitable for handling high-dimensional features.

Kernel SVM (Support Vector Machine):

Key: Suitable for not linearly separable data.

- The parameter 'C': penalty of misclassification; larger values lead to heavier penalties for misclassification.
- 'kernel': linear, RBF (Gaussian), polynomial, and sigmoid kernels, allowing different data mappings to be tested to find the best fit for the model.
- The parameter 'gamma: the width of the Gaussian kernel; larger values make a more flexible decision boundary.

Softmax Regression:

Key: Suitable for multi-class classification problems.

- The parameter 'C: the strength of regularization; smaller values lead to stronger regularization that

helps to prevent overfitting.

k-Nearest Neighbors:

Key: Classifies based on the category of nearest neighbors.

- The parameter 'n_neighbors': the number of neighbors to consider; larger values make the classification more stable but could smooth the results.
- The parameter 'weights': weighting method for neighbors, which can be uniform or distance-weighted.
- The parameter 'metric': the distance measurement method, often use Euclidean or Manhattan distance.

Decision Tree:

Key: Easy to understand and explain.

- The parameter 'max_depth': the maximum depth of the tree; larger values make the model more complex and may result in overfitting.
- The parameter 'min_samples_split: the minimum number of samples required to split an internal node; larger values can reduce overfitting.
- The parameter 'min_samples_leaf': the minimum number of samples required at a leaf node; larger values can also reduce overfitting.

Random Forest:

Key: An ensemble of multiple decision trees, which can reduce overfitting and improve generalization.

- The parameter 'n_estimators' controls the number of trees; larger values make the model more stable but increase computational costs.
- Other parameters similar to those of the decision tree are used to control the complexity of individual trees and reduce overfitting.

Gradient Tree Boosting:

Key: Builds a strong learner by iteratively training weak learners (decision trees) and combining them.

- The parameter 'n_estimators' controls the number of boosting stages (trees).
- The parameter 'learning_rate': the contribution of each tree; smaller values make the model more conservative.
- The parameter 'max_depth': the maximum depth of each tree.
- The parameter 'subsample': the proportion of samples used for training each base learner that can prevent overfitting.

XGBoost (Extreme Gradient Boosting):

Key: Efficient implementation of gradient boosting trees.

- Parameters like 'n_estimators', 'learning_rate', 'max_depth', and 'subsample' are similar to those parameters that used in Gradient Tree Boosting.
- The parameter 'colsample_bytree': the proportion of columns sampled when building each tree.
- The parameter 'gamma': the minimum loss reduction required to make a further partition on a leaf node of the tree; larger values make the model more conservative.

ADABoost (Adaptive Boosting):

Key: Builds a strong learner by training a series of weak learners and adjusting sample weights.

- The parameter 'n_estimators': the number of weak learners.
- The parameter 'learning_rate': the contribution of each model to the final combination.

Binary Relevance:

Key: Transforms a multi-label classification problem into multiple independent binary classification problems.

- Parameters 'estimator__C' and 'estimator__kernel': the complexity and kernel function of the underlying SVM classifier.

Gaussian Naive Bayes:

Key: Suitable for continuous features, assuming features distribute as a Gaussian distribution.

- The parameter 'var_smoothing': the degree of variance smoothing; smaller values increase the influence of the prior probability.

Experiments Results:

Traditional Machine Learning Classifier-Label Encode	**Accuracy**
Logistic Regression-Label Encode	0.925
Kernel SVM(poly)-Label Encode	0.951
Softmax Regression-Label Encode	0.948
k-Nearest Neighbors-Label Encode	0.948
Decsision Tree-Label Encode	0.951
Random Forest-Label Encode	0.953
Gradient Tree Boosting-Label Encode	0.939
XGBoost-Label Encode	0.927
ADABoost-Label Encode	0.901
Binary Relevance-Label Encode	0.948
Gaussian Naive Bayes-Label Encode	0.948
Sequential Model	**Accuracy**
Sequential Model-Label Encode_ UMAP_Random	0.951
Sequential Model-One-Hot Encode-Random	0.987
Sequential Model-One-Hot Encode-Hyperband	0.991
Sequential Model-One-Hot Encode-Bayesian	0.981

Key Analysis:

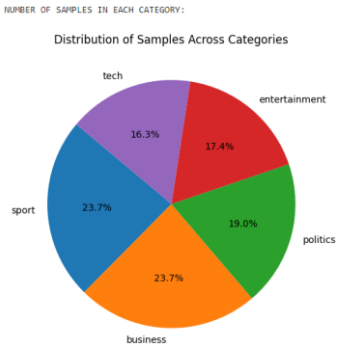
	Incorrect Predictions	Best parameters
Logistic Regression	Total incorrect predictions: 32 Highest error in Label tech: 11/69 incorrect predictions (15.94%)	{'C': 1, 'penalty': 'l2'}
Kernel SVM	Total incorrect predictions: 20	{'C': 0.01, 'gamma': 'auto', 'kernel':

	Highest error in Label tech: 10/69 incorrect predictions (14.49%)	'poly'}
Softmax Regression	Total incorrect predictions: 22 Highest error in Label tech: 9/69 incorrect predictions (13.04%)	{'C': 10}
k-Nearest Neighbors	Total incorrect predictions: 21 Highest error in Label tech: 10/69 incorrect predictions (14.49%)	{'metric': 'manhattan', 'n_neighbors': 7, 'weights': 'distance'}
Decision Tree	Total incorrect predictions: 21 Highest error in Label tech: 9/69 incorrect predictions (13.04%)	{'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 5}
Random Forest	Total incorrect predictions: 20 Highest error in Label tech: 10/69 incorrect predictions (14.49%)	{'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100}
Gradient Tree Boosting	Total incorrect predictions: 26 Highest error in Label tech: 13/69 incorrect predictions (18.84%)	{'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 200, 'subsample': 0.8}
XGBoost	Total incorrect predictions: 31 Highest error in Label tech: 11/69 incorrect predictions (15.94%)	{'colsample_bytree': 1.0, 'gamma': 0.1, 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 200, 'subsample': 1.0}
ADABoost	Total incorrect predictions: 42 Highest error in Label tech: 21/69 incorrect predictions (30.43%)	{'learning_rate': 1.0, 'n_estimators': 50}
Binary Relevance	Total incorrect predictions: 22 Highest error in Label tech: 10/69 incorrect predictions (14.49%)	{'estimator__C': 10, 'estimator__kernel': 'rbf'}
Gaussian Naive Bayes	Total incorrect predictions: 22 Highest error in Label tech: 9/69 incorrect predictions (13.04%)	{'var_smoothing': 1e-09}

We can find that all 11 classifiers have the highest incorrect predictions in Label Tech. In the UMAP plot, we can also find that Label Tech has higher overlap with all the category. Moreover, we perform dimensionality reduction into 2 dimension could also lead the important feature lost during the procession. Second, the sample of Tech account for the lowest portion of the dataset, even if we already adopted stratify sampling.

```
Counts for each category:
category
sport            504
business         503
politics         403
entertainment    369
tech             347
Name: count, dtype: int64

Total number of rows: 2126
```



Conclusion:

- Label Cncode and UMAP appears to hit the limits of the accuracy, no matter in traditional Machine Learning classifiers or Sequential Model, the values is no higher than 0.953.
- In Sequential Model, accuracy of One-Hot Encode models is 3%~4% higher than the Label Encode_UMAP model.
- Among Random Search, Hyperband Optimization, and Bayesian Optimization, Hyperband reaches the highest accuracy rate 0.991.
- The final test accuracy of the best Sequential Model on Test Set is 0.971.

Future Works:

- For UMAP dimensional deduction, we should try other value, such as 3,4 or 5, instead of 2 in both Machine Learning Classifiers and Sequential Model.
- Deeply examine TensorBoard function in Sequential Model of Random search, Hyperband optimization and Bayesian optimization.
- Conduct linguistics analysis in an appendix section.
- Compare the Cased BERT to Uncased BERT embeddings.
- Literally build BERT architecture to train models.
- Use SMOTE for imbalanced dataset classification task.