International Journal on Cybernetics & Informatics (IJCI) Vol. 12, No.3, June 2023

ALICE – APPLYING BERT TO ITALIAN EMAILS

Pasquale Restaino and Liliana Saracino

Sogei S.p.A., Rome, Italy

ABSTRACT

ALICE is an Artificial Intelligence solution that allows the automatic classification of email-type documents based on their information content, analyzed almost in real-time. The emails are written in Italian language. The classification classes are equal to 591, and in the use of the service, they will be able to grow further. The aim of this paper is to explores the implementation of the BERT model in the ALICE email classification system for multiclass classification in the Italian language. The main objective of the ALICE solution is the automation of classification processes performed manually by dedicated operators. To improve the performance of the BERT model and to allow the addition of further classification classes, a transfer learning process has been envisaged.

The ALICE service, which uses the BERT model trained on the provided dataset, presently has a 76% accuracy.

KEYWORDS

BERT model, multiclass text classification, Italian emails

1. INTRODUCTION

The email classification is a very important issue in industrial realities. This is because it requires huge human work. Furthermore, it can be often more complex because it can be necessary to analyze not only the subject of the email but also its body, attachments and recipients. Artificial Intelligence algorithms can streamline this operation. The aim of this research is to speed up the email classification by using BERT. In particular the emails are written in Italian language. To realize the classification it is necessary to analyze the subject, the recipients and the names of the attachments. The continuous learning typical of Transformers can be the key to success in solving this type of problem. For this reason it was decided to experiment with the use of BERT for this scenario.

BERT (Bidirectional Encoder Representations from Transformers) is a type of transformer-based language model that is particularly effective at performing natural language processing tasks such as language translation, question answering, and text classification [1][6].

1.1. History of BERT Multiclass Classification

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based architecture for natural language processing tasks developed by Google [2]. It was introduced in a paper published in 2018 by researchers at Google, titled "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." [1].

One potential application of BERT is for multiclass email classification, where the goal is to classify an email into one of several predefined categories [2]. For example, an email classification system might classify an email as spam, promotional, or personal.
Before the development of BERT, various methods were used for email classification, including techniques based on hand-crafted features, such as the use of specific words or the presence of certain characters. With the advent of deep learning, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) were also applied to email classification tasks [3].

The use of the BERT model represents an innovation in natural language processing. BERT allows you to find and classify the relationships between words in a sentence more effectively than previous models. It does this by using attention mechanisms, which allow the model to focus on specific parts of the input when processing it. This allows BERT to achieve state-of-the-art performance on a wide range of natural language processing tasks, including email classification [4].

1.2. Classifying Emails using BERT's Method

To classify emails into different categories (e.g., spam, promotional, transactional, etc.), BERT would first preprocess the text of the emails by tokenizing it into individual words and subwords and then encoding each word using a pre-trained word embedding [5]. This process creates a numerical representation of the text that can be fed into a machine-learning model.

Next, the encoded text is input into the BERT model, composed of multiple layers of attention-based transformer encoders. The BERT model processes the input text in both a left-to-right and a right-to-left direction, which allows it to capture contextual relationships between words in the input text.

After processing the input text, the BERT model produces a fixed-length output vector that encodes the meaning of the input text. This output vector can then be fed into a classifier, such as a support vector machine (SVM) or a multi-layer perceptron (MLP), which will use the vector to predict the class of the input text [6].

In summary, to perform multiclass email classification using BERT, the email text would be preprocessed, input into the BERT model to generate a fixed-length output vector, and then passed to a classifier to predict the class of the email.

1.3. Prediction Phase

In the prediction phase, BERT can then be used to make predictions about the meaning or intent of a given piece of text by using the learned contextual representations of the words in the text. This is done by adding a classification layer on top of the BERT model and training the model to predict a label (such as the sentiment of a piece of text) given the input text.

1.4. Classifying Emails using ALICE

To classify emails using ALICE, we first needed to prepare a training dataset with a set of example emails labeled with the appropriate class for each email [7]. We then use this dataset to train a machine learning model, such as a support vector machine (SVM) or a random forest, to classify emails based on their content.

Paper Structure. In Section 2, ALICE is presented. The integration of the BERT model into ALICE is described in Section 3. Related work are discussed in Section 4 while the use case is explained in Section 5. Sections 6 and 7 described the experimentation performed with ALICE while the results are sketched in Section 8. The limitation of the study are described in Section 9. In Section 10 analyzed the ethical considerations. The potential risk of the study are addressed in Section 11. Conclusions concludes the paper.

2. ALICE

ALICE (ALgorithm Intelligent Email Classification) is a SaaS service developed in microservices that allows you to classify incoming emails to a PEC box based on their information content. ALICE predicts a classification code based on an envisaged ownership relative to the offices that will have to work the email.

ALICE consists of 4 microservices: api (microservice that receives the PECs to be classified), predict(microservice that processes the emails and predicts the proprietary code by exploiting the trained BERT model), transfer-learning (microservice which, by exploiting correction feedback, carries out the retraining phase of the BERT model).

3. INTEGRATING THE BERT MODEL INTO ALICE

The following were researched to integrate a BERT (Bidirectional Encoder Representations from Transformers) model into ALICE (ALgorithm Intelligent Classification Email). Preprocess the text of the emails that one wants to classify by tokenizing it into individual words and subwords and then encode each word using pre-trained word embedding. This process creates a numerical representation of the text that can be input into the BERT model. Train a BERT model on a labeled dataset of emails. This could be a dataset of emails already labeled and classified. Or alternatively, a subset of emails one could label.

Use the trained BERT model to generate a fixed-length output vector for each email in the dataset.The output vector encodes the meaning of the input text and can be used as a feature machine-learning model.

Train a machine learning model, such as a support vector machine (SVM) or a random forest, on the output vectors generated by the BERT model. This model will be responsible for making predictions about the class of each email based on the output vectors from the BERT model. Integrate the trained machine learning model into ALICE as a component that can be used to classify new, unseen emails. One can do this by writing code that takes an email as input, generates an output vector using the BERT model, and then passes the output vector to the machine learning model to predict the class of the email.

4. RELATED WORK

"Multi-Class Text Classification With Deep Learning Using BERT" is an article by Susan Li that discussesusing BERT, a state-of-the-art natural language processing (NLP) model, for text classification tasks. Li shows how to fine-tune BERT for text classification using the Transformers library and the IMDB movie review dataset [8].

Li begins by introducing BERT and explaining how it works. She describes how to fine-tune BERT for text classification using the IMDB movie review dataset. Next, shehe explains how to prepare the dataset, define the model, and train and evaluate the model.

Throughout the article, Li provides code examples and explanations of the different steps involved in fine-tuning BERT for text classification. She also discusses some of the challenges and considerations involved in using BERT for text classification tasks, such as the need to choose the appropriate learning rate and the importance of properly preprocessing the data.

Another study was done on Multi-class Text Classification using BERT-based Active Learning which revealed that BERT-based models can be effective for multi-class text classification in an active learning setting, and that different active learning strategies can be used to train BERT-based models cost-effectively with less data. The study also found that the EGL (expected gradient length) strategy performs reasonably well across datasets. [9]

It's worth noting that active learning is a technique that can be used to improve the efficiency of machine learning models by selectively choosing the most informative examples to label and include in the training set. By using active learning, it may be possible to train BERT-based models with fewer labeled examples and still achieve good performance.

Overall, the study suggests that BERT-based models are a promising option for multi-class text classification tasks and that active learning strategies can be an effective way to train these models efficiently with less data.

5. USE CASE

The goal was to create an Artificial Intelligence service capable of classifying emails in one of the 591 possible classification classes. Each class corresponds to an owner code and each owner code corresponds to an office to which the email must be assigned and which will have to work with it. The classification takes place on the basis of the information content of the email, in particular by exploiting the following information: sender email address, recipient email address, subject of the email and names of the attachments. The content of the email is not used because very often in this use case the users who send the emails do not specify anything in the content, but are emails sending attached documentation.

To create the service and in particular to develop the multiclass classification AI model, it was necessary to start from a dataset of 1 million emails already classified by humans.

To develop an AI model for classifying emails based on the information provided (sender, recipient, subject, and attachments), we used a supervised machine learning approach [10]. This involves training a model on a labeled dataset of emails, where each email is associated with a particular class.

One approach taken was to preprocess the email data by extracting relevant features from the information provided [11]. This could involve tokenizing the sender and recipient email addresses, subject, and attachment names and encoding them as numerical values. Then use these features as input to a multiclass classification model, such as a support vector machine (SVM) or a neural network.

To train the model, split the labeled email dataset into a training set and a test set. The model would be trained on the training set and subsequently evaluated on the test set. We used a variety of metrics to evaluate the performance of the model, such as precision, recall, and F1 score [12]. Once the model has been trained and evaluated, it was deployed as a service for classifying new emails.

6. DATA PREPARATION

In the preliminary analysis phase, the dataset proved to be particularly unbalanced. In particular, for some classification classes, only 1 email was present within the dataset, while for other classification classes, and there were more than 10,000 significant samples.

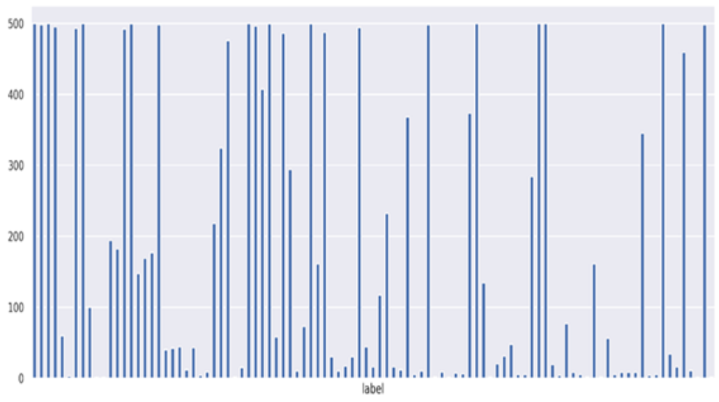

Figure 1. Distribution 100 classes

The dataset was identified as imbalanced, with some classes having significantly more examples than others. To address this issue, the SMOTE (Synthetic Minority Oversampling Technique) method is used to oversample the minority classes by synthesizing new examples from the existing ones [13].

We decided to oversample the less numerous classes using the SMOTE methodology to obtain accurate and significant performances. The SMOTE methodology was applied only on the training dataset.

6.1. SMOTE

SMOTE is a well-known strategy for coping with uneven datasets in classifying challenges. It operates by picking a minority class and locating its k nearest neighbours. It then generates a new example by randomly choosing one of the closest neighbors and constructing a linear combination of the chosen example and its neighbour, with the coefficients drawn arbitrarily from the interval [0, 1]. This method is replicated for each example of a minority class, producing a new, big dataset with a more stable supply of cases across the various classes [14].

Imbalanced classification entails building prediction models on classified datasets with a high degree of class imbalance.

Dealing with class imbalance can be problematic since the minority class could be underestimated,thus making it difficult to gain knowledge. Moreover, it can result in poor performance on the minority class, which is very essential in some applications (such as fraud detection or medical diagnosis).

Several approaches can be used to address imbalanced classification problems. One approach is to use algorithms specifically designed to handle imbalanced datasets, such as balanced random forests or balanced gradient boosting. Resampling the dataset to achieve a balanced dataset via overrunning the minority class or downsamplinthe majority class would be another option.

Other techniques for addressing imbalanced classification include adjusting the class weights or thresholds used by the model or using techniques such as synthetic minority oversampling (SMOTE) to generate synthetic data points for the minority class. It is crucial to closely study the model's performance on the minority class while dealing with imbalanced datasets, as this may be the most important class for the application.

Oversampling the minority class was one strategy we employed to correct imbalanced datasets. The most basic method is copying examples from the minority class, even if these examples offer no additional knowledge to the model. Instead, new instances can be created by combining old ones. The Synthetic Minority Oversampling Technique (SMOTE) is a method of data enrichment for the minority class.

SMOTE creates additional examples for the minority class. Fitting and evaluating machine learning models on SMOTE-transformed trained data. Using SMOTE features to create synthetic cases among the class decision boundary [15].

Once the dataset balancing phase was completed, the email preprocessing phase and the cleaning of the possible noise in the model training phase were carried out.

Each email has been subjected to preprocessing activities. In particular, the spelling correction, removal of duplicates, removal of file extensions, removal of special characters, removal of stopwords and excess numbers and spaces has been implemented.

Preprocessing is an important step in preparing data for model training [16]. By cleaning and formatting the data, one can improve model performance and make it more accurate. Spelling correction and duplicate removal can aid in reducing distortion in data and making it simpler for the model to grasp. Removing file extensions, special characters, and excess numbers and spaces can also help to improve the model's performance. Removing stopwords, commonly used words that do not contain much meaning, can help the model focus on the more important words in the data. These preparation methods can enhance data quality while also making it more suited for model training.

6.2. TRAINING

BERT is a state-of-the-art natural language processing model that has been shown to outperform many other models various tasks. However, it is not necessarily always the best choice for every task.

In general, BERT is well-suited for tasks that involve understanding the meaning and context of words in a piece of text, such as text classification, question answering, and language translation [17]. It is less well-suited for tasks that do not require this type of understanding, such as image classification or numerical prediction tasks.

Many factors can influence model performance, including the specific task, the quality of the training data, and the hyperparameters used to train the model. So, if one requires to incorporate image classification or numerical prediction tasks with multi-class classification can train a BERT model for this task.

Trying multiple different models and evaluating their performance on specific tasks in order to determine which model works best is very crucial. By adding a training phase will broaden one’s horizons, enabling coverage of everything using BERT’s model. Using BERT (or any other machine learning model) in the training phase can also allow one to fine-tune the model to their specific task and dataset. Fine-tuning involves adjusting the hyperparameters and training the model on one’s own data, which can help to improve the performance of the model on specific task.

The preprocessing activity has allowed obtaining a clean and optimized dataset for the training phase of the BERT model. However, before using the BERT model, other multiclass classification models have been used,for example, the algorithms of the sklearn library: LinearSVC, RandomForest, DecisionTree, KNeighbors, Naive Bayes,etc.

However, all these algorithms even following the search for the best parameters using the GridSearch, did not allow achieving good model accuracy. In particular, the results will be analyzed in the next section.

LinearSVC, RandomForest, DecisionTree, KNeighbors, and Naive Bayes are different the types of multiclass classification algorithms that can be used to classify emails [18]. Each algorithm has its unique characteristics and may work better or worse depending on the specific dataset one is using.

LinearSVC (Support Vector Classification) is a linear model that performs binary classification [19]. It tries to find the hyperplane in the feature space that maximally separates the two classes. To use LinearSVC for classification, One will need to provide it with a training dataset consisting of labeled examples. One can then use the fit method to train the classifier on the training data. Once the classifier has been trained, one can use the predict method to classify new examples. LinearSVC is a powerful and efficient classifier that is widely used in a variety of applications. It is particularly well-suited for linear classification tasks and can be a good choice when one have limited data or when one is working with high-dimensional datasets [20].

RandomForest is an ensemble training approach for categorization, regression, and many other tasks typically works by training a large number of decision trees and then outputs the class that is the method of the classes (classification) or the average forecast (regression) of the individual trees.

To use Random Forest for classification, one will need to use the RandomForestClassifier class from the scikit-learn library. In addition, one will need to provide the classifier with a training dataset consisting of labeled examples. One can then use the fit method to train the classifier on the training data.

Once the classifier has been trained, one can use the prediction method to classify new examples. One can also use the predict_proba method to obtain the class probabilities for each example. There are several hyperparameters that one can tune to control the behavior of the Random Forest classifier, such as the number of decision trees to include in the ensemble and the maximum depth of each tree. Cross validation and grid search can be used to find the best combination of hyperparameters for one’s dataset.

Overall, Random Forest is a powerful and widely-used method for classification tasks. It is particularly well-suited for tasks where the data is noisy or where there are a large number of features, and it can often achieve good performance even with little hyperparameter tuning [21].

DecisionTree is a managerial device that employs a tree-like flowchart layout or framework of options and their possible results, incorporating outcomes, input prices, and usefulness [22]. To use Decision Tree for classification, one can use the DecisionTreeClassifier class from the scikit-learn library. To train a Decision Tree classifier, one will need to provide it with a training dataset

consisting of labeled examples. Then the fit method to train the classifier on the training data is used.
Once the classifier has been trained,the predict method can be used to classify new examples.Also, the predict proba method can be used to obtain the class probabilities for each example. There are several hyperparameters that one can tune to control the behavior of the Decision Tree classifier, such as the maximum depth of the tree and the minimum number of samples required to split a node.
Overall, Decision Tree is a simple and widely-used method for classification and regression tasks. It is particularly well-suited for tasks where the data is structured, and the relationships between features are easy to understand. It is also often used as a baseline method for comparison with more sophisticated models [23].

KNeighbors is a non-parametric method used for classification and regression [24]. In both cases, the input consists of the k closest training examples in the feature space.

To use KNN for classification, one can use the KNeighborsClassifier class from the scikit-learn library. To train a KNN classifier, there is always the need to provide it with a training dataset consisting of labeled examples. In addition, one will be required to specify the number of neighbors (K) that they want the classifier to consider when making predictions.

There are several hyperparameters that one can tune to control the behavior of the KNN classifier, such as the number of neighbors to consider and the distance metric to use.Cross validation and grid serach can be utilized to find the best combination of hyperparameters for the dataset.

Overall, KNN is a simple and widely-used method for classification and regression tasks. It is particularly well-suited for tasks where the data is structured, and the relationships between features are easy to understand. It is also often used as a baseline method for comparison with more sophisticated models [25].

Naive Bayes is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

To use Naive Bayes for classification, one can use the GaussianNB, MultinomialNB, or BernoulliNB classes from the scikit-learn library. These classes implement the Gaussian, multinomial, and Bernoulli versions of Naive Bayes, respectively. To train a Naive Bayes classifier, there will be a need to provide it with a training dataset consisting of labeled examples. one can then use the fit method to train the classifier on the training data.

Once the classifier has been trained, one can use the predict method to classify new examples. Also, predict_proba method can be used to obtain the class probabilities for each example. Overall, Naive Bayes is a simple and efficient method for classification that can be a good choice when there is limited data or when one is working with high-dimensional datasets. It is also often used as a baseline method for comparison with more sophisticated classifiers [26][27][28].

After trying out these different algorithms, one can compare their performance and choose the one that works best for their dataset. Trying out BERT, which is e of the art language model that has shown great excellent results in many natural language processing tasks is very crucial.

For the training phase of the BERT model, the library ktrain(https://github.com/amaiya/ktrain). In particular, the dataset has been split into three datasets. 80% of the dataset was used for the

training phase of the BERT model, 10% for the TEST phase and the remaining 10% for the model accuracy validation phase.
In particular, in this subdivision, having at least one example of email in the training dataset, in the test dataset and in the validation dataset was always considered.

Splitting one's dataset into training, test, and validation sets is a common practice in machine learning [29]. The training set is used to train the model, the test set is used to evaluate the model's performance, and the validation set is used to tune the model's hyperparameters.

It is important to ensure that each set is representative of the overall dataset, and includes at least one example of each type of email. This helps to ensure that the model is not overfitted or underfitted to the training data.

Using ktrain to train the BERT model is a good choice, as it is a library specifically designed to make it easy to use BERT for text classification tasks. The ktrain API can be used to define one's model, load in their data, and train the model using the training set. The library also includes functionality for evaluating the model's performance on the test set and fine-tuning the model's hyperparameters using the validation set [30].

7. FINE-TUNING BERT ON ITALIAN EMAILS

To start the training phase of the BERT model, Next, we instantiate a Learner object and call the lr_find and lr plot methods to help identify a good learning rate [31]. The lr find method allows to briefly similuate the training of the BERT model to find the best learning rate. The lr plot method allows to visually identify the best learning rate. Finally, we will use the fit_onecycle method that employs a 1 cycle learning rate policy and train 3 epoch with a learning rate=3e-5.

It is a good strategy to use the lr_find and lr_plot methods to find a good learning rate before training one's model. The learning rate determines how fast or slow the model updates its weights during training. A learning rate that is too high can cause the model to converge too quickly and potentially miss the optimal weights, while a learning rate that is too low can cause the model to take a long time to converge. Using the lr_find method, one can automatically search for a good learning rate by training the model over a range of learning rates and plotting the resulting loss values. You can then use the lr_plot method to visualize the results and choose a learning rate that is near the minimum of the loss curve.

Using the fit_onecycle method implies that that the learning rate increases linearly from a low value to a high value during the first half of training, and then decreases linearly from the high value back to the low value during the second half of training. This can help the model to converge faster and potentially improve its performance. Training for 3 epochs with a learning rate of 3e-5 should be sufficient for most text classification tasks.

8. RESULTS

By exploiting the algorithms of the sklearn library, during the model training phase, little comforting results were obtained regarding the accuracy and precision of the trained classification model. In particular, considering the dataset defined above, the following results were achieved (Precision(P), Recall(R), F1 score (F1)):

- LinearSVC: P-68%; R-59%; F1-61%;
- RandomForest: P-61%; R-55%; F1-60%;

- DecisionTree:P-47%; R-46%; F1-47%;
- KNeighbors: P-52%; R-50%;F1-50%;
- Naïve Bayes: P-62%; R-54%; F1-60%;

By training the BERT model on the defined dataset, 76% accuracy of the final model, evaluated on the TEST and VALIDATION datasets, was achieved. In particular, below are the performance metrics calculated for the trained model:

- Precision: 76%;
- Recall: 74%;
- F1 score: 75%

We obtained good results when training the BERT model on our dataset. A 76% accuracy is a good benchmark for text classification tasks, and it suggests that the BERT model can accurately classify the emails in our dataset.

It's also interesting to see how the other classification algorithms we tried performed on our dataset. The LinearSVC, RandomForest, Naive Bayes, and KNeighbors algorithms achieved relatively good accuracy results, but the DecisionTree algorithm performed significantly worse. This could be due to our dataset's specific characteristics of or the way the DecisionTree algorithm works. In general, DecisionTree algorithms work well on datasets with a large number of features, but they can be prone to overfitting when there are many features relative to the number of samples.

We tried out various algorithms to see which one works best for our specific dataset and fine-tune the hyperparameters of each algorithm to see if we can improve their performance.

For the evaluation of the results, the PRECISION and RECALL curve and the classification report relating to all the metrics of the classification models (precision, recall, f1score) were considered.

We considered a variety of evaluation metrics, such as precision and recall, when evaluating the performance of one's model. These metrics can provide a more comprehensive understanding of how well the model is performing and how it is making its predictions [32].

Precision is a measure of the accuracy of the model's positive predictions. It is calculated as the number of true positive predictions made by the model divided by the total number of positive predictions made by the model. A model with high precision is accurate in its positive predictions, but it may still make many false negatives.

Recall is a measure of the model's ability to find all relevant instances in the data. It is calculated as the number of true positive predictions made by the model divided by the total number of relevant instances in the data. A model with high recall is able to find most of the relevant instances in the data, but it may still make many false positives.

The precision-recall curve is a useful tool for visualizing the trade-off between precision and recall for different classification thresholds. A high precision means that the model is good at returning only relevant results, while a high recall means that the model is able to return most of the relevant results. A model with a high precision and a high recall is generally considered to be a good model.

The classification report is another useful tool for evaluating the performance of a classification model [33]. It provides a summary of the precision, recall, and f1-score for each class, as well as

the overall accuracy of the model. The f1-score is the harmonic mean of precision and recall, and it is a good metric to use when there is need to balance precision and recall.
The model BERT is able to classify 79 classification classes with an accuracy between 100-90% and only 13 classification classes are predicted with an accuracy in the 0-19% range.This suggests that the model is performig well overall, with only a small number of classes being misclassified.

An incremental training of the BERT model was also carried out, which made it possible to highlight how performance improves as the number of emails per classification class increases.
Incremental training is a technique where an individual gradually train a model on increasingly large amounts of data. This can be useful for a number of reasons. For example, the dataset available is very large, it may be too time-consuming or computationally expensive to train the model on the entire dataset at once. By incrementally training the model on smaller chunks of data,the model can be trained in a more reliable and efficient manner [34].

Incremental training can also be useful for improving the performance of a model as the number of examples per class increases [35]. This is because the model can learn more about each class as it sees more examples of it. As a result, the model may be able to make more accurate predictions on the test set.

Overall, incremental training can be a useful technique for improving the performance of a model and making it more efficient to train.

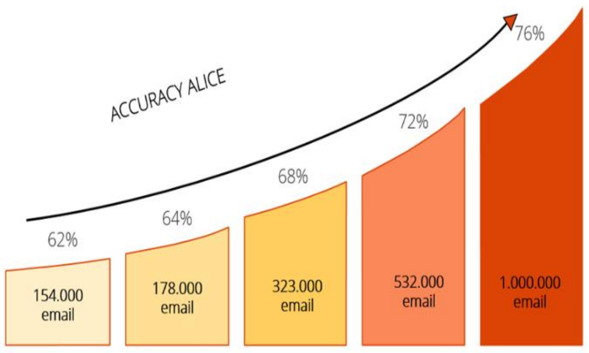

Figure 2. Learning curve of ALICE

Overall, incremental training can be a useful technique for improving the performance of a model and making it more efficient to train.

9. LIMITATIONS

One of the major limitation of this research study involves the small dataset that was utilized. The study utilized a relatively small dataset which may bring in limited generalization of the results obtained. For this kind of research, a large dataset with more examples would be ideal in order to authenticate the effectiveness of the BERT model in real-world setting.

This research study only concentrated on the BERT model and did not consider putting a comparison between the performance of the BERT model and other state-of-the-art models such as Transformer-XL. In order to have a proper investigation on the performance of the BERT model, further research ought to consider highlighting the comparison of different models in order to be in a better position of ruling out which model is the most effective in classification of emails.

The research study did not provide an evaluation of the interpretability of the BERT model. This is a very crucial aspect which could have been incorporated in the research as it plays a very key role in real-world application of the BERT model. Therefore, further research ought to scrutinize the interpretability of the BERT model and how it can be utilized in the processes of explaining the classification decisions.

Finally, the other limitation of this study is that the training dataset could have manually made classification errors which affects the generalization of the results obtained.

10. Ethical Considerations

Transparency and accountability: One of the major complexities of the BERT model is that it is not easy to interpret. This can be a potential risk in regard to accountability and transparency in real-world applications of the model. Therefore, it is very crucial to put into consideration ways on how an interpretable and transparent explanation of the decisions of the model can be made in order to ensure accountability.

Fairness and ethical use: there is a need to put into consideration the ethical use of the BERT model and how it affects different people from different walks of the world. This can incorporate activities such as making sure that the model does not discriminate based on who they are. By doing this, the model can be put in use in a more fair and just manner.

11. Potential Risk of the Study

One of the potential risk of this study is associated with generalization of its results. In this study, there was no check on the manual classification of the emails that made up the training dataset. Therefore, there might be errors which occurred during the manual classification process which has an impact on the results obtained. Also, classification of emails may be similar. Thus, generalizing the results of this study maybe risky as it may lead to misleading results in future studies.

This research study did not put into consideration ways in which the BERT model affects privacy and security in real-world application. This is a very key aspect left out by this study. The BERT model necessitates access to large amounts of data. This can pose a risk of privacy and security especially if the data being handled is not protected in a proper manner.

The other potential risk of this study is that the BERT model may not be ascendable for large-scale classification of emails. This is because, the model is a deep learning model which requires a significant amount of computational resources, which may not be achievable for tasks which involve large-scale email classifications.

12. Conclusions

The ALICE service, which exploits the BERT model trained on the described dataset, has currently achieved an accuracy of 78%, thanks to the retraining phase implemented. In particular, for each email, ALICE provides a classification among the possible 591 classification classes. The result of ALICE is validated by an operator and if it is incorrect, it is modified by the operator himself. The correct classification code and correction are recorded by ALICE and used to retrain the AI model.

We get really good results by implementing a system that combines the use of ALICE with the BERT model for email classification.

It's interesting to see that the system has achieved an accuracy of 78% after the retraining phase. By incorporating human corrections and using them to retrain the model, we may be able to improve the performance of the system over time.

It's also good to see that the system is being validated by an operator to ensure that the classification results are accurate. This can help to improve the reliability of the system and ensure that it is making accurate predictions.

REFERENCES

[1] BERT intro article (BERT Basics: What It Is, Creation, and Uses in AI (h2o.ai)

[2] Devlin, Jacob & Chang, Ming-Wei & Lee, Kenton & Toutanova, Kristina. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[3] K., Radhika & K R, Bindu & Parameswaran, Latha. (2018). A text classification model using convolution neural network and recurrent neural network. International Journal of Pure and Applied Mathematics. 119. 1549-1554.

[4] Tida, Vijay Srinivas & Hsu, Sonya. (2022). Universal Spam Detection using Transfer Learning of BERT Model. 10.24251/HICSS.2022.921.

[5] Sahmoud, Thaer & Mikki, Dr. (2022). Spam Detection Using BERT. 10.48550/arXiv.2206.02443.

[6] Build a Text Classification Model using BERT and Tensorflow (https://www.section.io/engineering-education/classification-model-using-BERT-and-tensorflow/)

[7] Bacchelli, Alberto & Sasso, Tommaso & D'Ambros, Marco & Lanza, Michele. (2012). Content classification of development emails. Proceedings - International Conference on Software Engineering. 375-385. 10.1109/ICSE.2012.6227177.

[8] Multi Class Text Classification With Deep Learning Using BERT by Susan Li (Multi Class Text Classification With Deep Learning Using BERT | by Susan Li | Towards Data Science)

[9] Prabhu, Sumanth & Mohamed, Moosa & Misra, Hemant. (2021). Multi-class Text Classification using BERT-based Active Learning.

[10] Li, Wenjuan & Meng, Weizhi. (2015). An empirical study on email classification using supervised machine learning in real environments. 7438-7443. 10.1109/ICC.2015.7249515.

[11] Hassan, Muhammad Ali & Mtetwa, Nhamoinesu. (2018). Feature Extraction and Classification of Spam Emails. 93-98. 10.1109/ISCMI.2018.8703222.

[12] Fatourechi, Mehrdad & Ward, Rabab & Mason, Steven & Huggins, Jane & Schlögl, Alois & Birch, Gary. (2009). Comparison of Evaluation Metrics in Classification Applications with Imbalanced Datasets. 777 - 782. 10.1109/ICMLA.2008.34.

[13] SMOTE (Synthetic Minority Oversampling Technique) method (https://medium.com/@corymaklin/synthetic-minority-over-sampling-technique-smote-7d419696b88c)

[14] Han, Hui & Wang, Wen-Yuan & Mao, Bing-Huan. (2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. Adv Intell Comput. 3644. 878-887. 10.1007/11538059_91.

[15] An Introduction to SMOTE (https://www.kdnuggets.com/2022/11/introduction-smote.html)

[16] Data preprocessing : Computational Learning Approaches to Data Analytics in Biomedical Applications (https://www.sciencedirect.com/science/article/pii/B9780128144824000024)

[17] BERT language model (https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model)

[18] Rabbimov, Ilyos & Kobilov, s.s. (2020). Multi-Class Text Classification of Uzbek News Articles using Machine Learning. Journal of Physics: Conference Series. 1546. 012097. 10.1088/1742-6596/1546/1/012097.

[19] Linear SVC Machine Learning SVM example with Python (https://pythonprogramming.net/linear-svc-example-scikit-learn-svm-python/)

[20] Bi, Jinbo & Zhang, Tong. (2004). Support Vector Classification with Input Data Uncertainty.. Adv. Neural Inf. Process Syst.. 17.

[21] Kothapally, Nithesh Reddy & Kakulapati, Vijayalakshmi. (2021). Classification of Spam Messages using Random Forest Algorithm. Xi'an Dianzi Keji Daxue Xuebao/Journal of Xidian University. 15. 495-505. 10.37896/jxu15.8/048.

[22] Decision trees. (https://developers.google.com/machine-learning/decision-forests/decision-trees)

[23] Cavor, Ivana. (2021). Decision Tree Model for Email Classification. 1-4. 10.1109/IT51528.2021.9390143.

[24] K-Nearest Neighbor(KNN) Algorithm for Machine Learning (https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning)

[25] Li, Baoli & Yu, Shiwen & Lu, Qin. (2003). An Improved k-Nearest Neighbor Algorithm for Text Categorization.

[26] Raschka, Sebastian. (2014). Naive Bayes and Text Classification I - Introduction and Theory. 10.13140/2.1.2018.3049.

[27] Zhang, Haiyi & Li, Di. (2007). Naïve Bayes Text Classifier. 708-708. 10.1109/GRC.2007.4403192.

[28] Zhang, Wei & Gao, Feng. (2011). An Improvement to Naive Bayes for Text Classification. Procedia Engineering. 15. 2160-2164. 10.1016/j.proeng.2011.08.404.

[29] Xu, Yun & Goodacre, Royston. (2018). On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning. Journal of Analysis and Testing. 2. 10.1007/s41664-018-0068-2.

[30] Maiya, Arun. (2020). ktrain: A Low-Code Library for Augmented Machine Learning.

[31] How to Decide on Learning Rate (https://towardsdatascience.com/how-to-decide-on-learning-rate-6b6996510c98)

[32] Zahera, Hamada & Sherif, Mohamed. (2020). ProBERT: Product Data Classification with Fine-tuning BERT Model.

[33] BERT-Based GitHub Issue Report Classification (https://ieeexplore.ieee.org/document/9808503)

[34] Shan, Guangxu & Xu, Shiyao & Yang, Li & Jia, Shengbin & Xiang, Yang. (2020). Learn#: A Novel Incremental Learning Method for Text Classification. Expert Systems with Applications. 147. 113198. 10.1016/j.eswa.2020.113198.

[35] Geng, X., Smith-Miles, K. (2009). Incremental Learning. In: Li, S.Z., Jain, A. (eds) Encyclopedia of Biometrics. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-73003-5_304