

Text Classification Using NLP



By

Fahid Bin Tariq

(Registration No: 330491)

Department of Electrical Engineering

School of Electrical Engineering & Computer Science (SEECS),

National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(March 2024)

Text Classification Using NLP



By

Fahid Bin Tariq

Fall-2020-MS-EE 330491 SEECS

Supervisor

Dr. Ahmad Salman

Department of Electrical Engineering

A thesis submitted in partial fulfillment of the requirements for the degree of Masters
of Science in Electrical Engineering (MS EE)

In

School of Electrical Engineering & Computer Science (SEECS),

National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(March 2024)

Approval

It is certified that the contents and form of the thesis entitled "Text Classification Using NLP" submitted by Fahid bin Tariq have been found satisfactory for the requirement of the degree

Advisor : Dr. Ahmad Salman

Signature:  _____

Date: 25-Sep-2023

Committee Member 1:Dr. Salman Abdul Ghafoor

Signature:  _____

25-Sep-2023

Committee Member 2:Dr. Khawar Khurshid

Signature:  _____

Date: 02-Oct-2023

Signature: _____

Date: _____

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis entitled "Text Classification Using NLP" written by Fahid bin Tariq, (Registration No 00000330491), of SEECs has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.


Signature: _____  _____

Name of Advisor: Dr. Ahmad Salman _____

Date: 25-Sep-2023 _____

HoD/Associate Dean: _____ 

Date: 25-Sep-2023 _____

Signature (Dean/Principal):  _____

Date: 25-Sep-2023 _____

National University of Sciences & Technology
MASTER THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by: (Student Name & Reg. #) Fahid bin Tariq [00000330491]

Titled: Text Classification Using NLP


be accepted in partial fulfillment of the requirements for the award of Master of Science (Electrical Engineering) degree.

Examination Committee Members

1. Name: Salman Abdul Ghafoor


Signature: 
08-Apr-2024 10:32 AM

2. Name: Wajid Mumtaz

Signature: 
08-Apr-2024 10:32 AM

Supervisor's name: Ahmad Salman


Signature: 
10-Apr-2024 3:14 PM


Salman Abdul Ghafoor
HoD / Associate Dean

08-April-2024
Date

COUNTERSIGNED

15-April-2024
Date


Muhammad Ajmal Khan
Principal


Dedication

This thesis is dedicated to all the deserving children who do not have access to quality education especially young girls.

Certificate of Originality

I hereby declare that this submission titled "Text Classification Using NLP" is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEecs or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEecs or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Student Name:Fahid bin Tariq

Student Signature: 

Acknowledgments

Glory be to Allah (S.W.A), the Creator, the Sustainer of the Universe. Who only has the power to honour whom He please, and to abase whom He please. Verily no one can do anything without His will. From the day, I came to NUST till the day of my departure, He was the only one Who blessed me and opened ways for me, and showed me the path of success. Their is nothing which can payback for His bounties throughout my research period to complete it successfully.

Fahid Bin Tariq

Abstract

This research is based on the fine tuning of BERT model and applying long short term memory layers (LSTM). Bert which is already well known for text classification is being used along with other layers to further enhance the accuracy of text classification. Many existing Specific Bert models are available but they are only trained for a specific task. This paper shows the classification of four different classes: chats, emails, news and tweets. The method is pretty simple, at first dataset for each target class is collected and preprocessed using NLP libraries to remove extra and useless data from the datasets. The data-loader is prepared to feed the testing and validation data into a BERT base model. Before that, Bert tokenizer is used as Bert only takes data which is presented in a specific format having Special tokens ([CLS] and [SEP]). Using a recommended approach Bert is fine-tuned one by one for all target classes. The innovation is the introduction of LSTM layers merged with fully connected (FC) and some pooling layers in case. The trained output of Bert, which is Bert-embedding, is being used as an input to the LSTM model. Although, Bert alone could perform well, but for some complex datasets these additional layers have provided an edge. As LSTM layers are being used in bi-direction to further capture the feature in the text to classify them more efficiently. Accuracy is enhanced overall. However, for binary classification with limited datasets there is a minor change in accuracy by introducing LSTM layers but for multi-classification with complex data, the accuracy is noticeable. For chats achieved accuracy is 99%; for emails, 98%; for news, 97%; and for tweets (complex data with multi-label sentiment analysis), 85%. These accuracies in comparison with alone Bert model are more efficient.

Contents

Introduction and Motivation.....	1
1.1 Problem Statement and Contribution.....	1
Literature Review.....	4
2.1 Text Classification using NLP	4
Design and Methodology.....	8
3.1 Pre-processing the Data.....	8
3.2 Preparing the data-loaders.....	8
3.3 Defining the BertLSTM model	11
Implementation and results.....	12
4.1 Chats Model	12
4.2 Emails Model	13
4.3 News Model	13
4.4 Tweets Model.....	14
4.5 Text Identifying Model	14
4.6 Results	15
Conclusion and Future Work.....	16
5.1 Future Work	16
References.....	17

List of Tables

Table 1: Text into Tokens	Table 2: Special Tokens and Attention Mask.....	9
Table 3: Binary Classification Comparison with Other Models		15
Table 4: Multi Classification Comparison with Other Models.....		15

List of Figures

Figure 1: Text Tokens into Tensors	10
Figure 2: Validation Accuracy for Chats Classification.....	12
Figure 3: Validation Accuracy for Emails Classification.....	13
Figure 4: Validation Accuracy for News Classification.....	13
Figure 5: Validation Accuracy for Tweets Classification	14
Figure 6: Validation Accuracy for Text Identifier Classification	14
Figure 7: Implementation Result of all five classification models.....	15

Introduction and Motivation

Text classification and sentiment analysis are coming in the limelight. Especially with the introduction of chat GPT, research in this field has increased a lot. Where the research in the field of Natural Language Processing has increased there its scope has become broaden as well. Text classification is normally used to classify text from spam to ham, fake to real and to place text into mentioned categories. Similarly, sentiment analysis is quite important as this social media era is full of such comments and tweets that identifying the emotion behind a text has become the need of the time. Luckily we have a system which can interpret the textual data into digits format. This digit thing is the language of computer and hence computer can learn from these digits. Natural Language Processing is domain of artificial intelligence where textual data is converted into digits which can only be understood by a computer. Then, there are some machine learning models which provide the facility to utilize the textual data and help the machine (computer) learn from it. In short, NLP acts like a bridge between human and a computer. The understanding of how NLP works is further linked to this paper [1].

1.1 Problem Statement and Contribution

Over the years many machine learning models were introduced to deal with textual data. Some of those did quite well but there was always a room for improvement. Machine learning models were performing well on the image data but when the same models were used to handle textual data, they got failed in achieving the desired accuracy. The main reason to that is the difference of data preprocessing techniques. Preprocessing of image data is different than that of textual data. Further, textual data needs to be transformed into specific digit format so that it can be used to train models. Natural language processing did this thing but the question to use which model remained obscure. Until, Recurrent Neural Network started to show some results in the textual data classification. Probably, this was because of the unique style of RNN where it remembers its previous

state to predict the next state including the introduction of NLP tokenizer. Following the same pattern BERT model was introduced to perform various operations on the text including classification of text, analysis of text, text tagging etc. Bert was only made to handle text data. After some time, innovation started to touch Bert models. Various Bert models were introduced for specific target classification and so on. This paper specifically sheds light on the use of Bert for various tasks [2]. Despite all that, accuracy is something which can be enhanced further and further with the introduction of new processes. The problem with Bert is, it is made specific and people have to fine tune it for their specific task. Normally, they use it for one target class and they achieve pretty desirable accuracy with it but while using Bert for complex data with multi-classification, it is hard to achieve good accuracy.

To achieve good accuracy for complex datasets Bert is used along with other machine learning layers to further enhance its accuracy. However, to do so, we have to follow some process which takes us to the fine tuning of Bert and then use of other layers. This process is defined in a hierarchal format. At first, we have to upload target datasets on which models are to be trained. Then by using NLP libraries we have to preprocess textual datasets to remove useless stuff from the data. That stuff includes removing stop words, stemming of text, tokenization, lower casing the data and normalization of data etc. As this process is almost same for all NLP tasks but the difference arises where we have to use main model.

Many machine learning models are also providing some good accuracy like recurrent neural network (RNN), [3]. This model uses a two directional path to capture features from the data. In this model, after preprocessing of textual data, NLP libraries are used to make text tokenizable in order to feed into the RNN model. Learning phase of RNN is something like remembering its previous state and creating a link with the next state. In this way it learns features from the textual data. However, RNN does provide limited accuracy depending on the context of the dataset.

In the similar manner, Bidirectional Recurrent Neural Network (BiRNN) was introduced which did a little better than simple RNN model. Whatsoever, the accuracy depends not only on the model but also on the nature and type of target class. One such paper on BiRNN [4], shares the achieved accuracy of 84% on heart failure detection from clinical notes. Although it is not that bad but it has so much tendency for improvement.

Another machine learning model, Bidirectional Gated Recurrent Unit (BiGRU), performed quite well in the sentiment analysis of textual data. Although this model

outperformed existing RNN models as it was a mix of RNN and Gated unit which enhanced the overall accuracy of the model. Beside all these models, Bert remained prominent and the credit goes to its structure which makes it easily fine tunable with good accuracy. This thesis, however, discusses not only the fine tuning of Bert but also the use of additional LSTM layers which is a key point in this paper.

The process is regular but the technique is different, at first datasets are loaded using panda library and preprocessed to clean the data. After that, Bert tokenizer is used which tokenizes the data so that it can be fed to Bert model. Then, fine tuning is performed and on the top of Bert model, LSTM layers are added to further enhance the accuracy of the model. Following five classes are being used to train the model and to compare the accuracies with the existing approaches.

- At first, chats data is loaded and after performing all the necessary operations on it, it is finally fed to Bert for fine tuning and then LSTM layers are used on the Bert output embedding. This method increase the chats classification accuracy by 99%.
- Similarly, emails data is used and accuracy is checked which show 98% increase than mere use of Bert tuning.
- Following on, news data is fed to the model and 97% accuracy increase was observed.
- Then, tweets data is used which is a multiclass sentiment analysis. Although its achieved accuracy (85.5%) is comparatively low than the cases mentioned above. However, its accuracy is better than other existing techniques in the field so far.
- In the end, a model to identify the text is used where the input sentence is first checked and then the relevant model is called to further define the classification or sentiment analysis.

This method introduces the use of pre-trained Bert model merged with LSTM layers. The mentioned accuracies for the above mentioned target classes: chats, emails, news and tweets, depict that text classification and analysis can be improved with bringing a little innovation in the existing models. In the next chapter, we are going to discuss literature review where detailed analysis of some latest approaches is discussed.

Literature Review

This chapter discusses some recent research papers in the domain of text classification using NLP and machine learning models. These models start from basic Recurrent Neural Network and leads to Transformer models. With the passage of time, many machine learning models were introduced for the relative tasks. Even Bert, which is a pre-trained model, was fine-tuned with respect to specific project. Many names of Bert including IndoBert, BioclinicalBert etc were introduced, but all of them were made for some specific tasks. Let's dig into some latest research papers to better understand text classification and to justify this research paper's method.

2.1 Text Classification using NLP

Word preprocessing, feature advance, and classifier model training are tasks related to text classification. Deep text-based word processing methods primarily employ word vector approaches to extract indirect text features, in contrast to conventional text-based word processing methods. This is a study of the sparse data problem [5] [6]. The three components of assembly are implementation, feature function, and classifier design. The training body is always clean and split in the input area utilizing the previously developed language model to process the words, learn the input details of the input sentence, and translate verb structure. Every communication network's foundation is feature advance. It customizes the text without losing important information by using feature curves created by specified character extraction networks to extract repeated features from text data. Based on the training data's class label information and text feature vector, the analyzer determines the model's prediction outcome. This is typically a completely ensemble network or a traditional trainer like softmax. Similar text prepends are used during testing, and the trained model is used to gather test data from the experimenter. Finally, the classifier's performance is evaluated using to the results. The unique feature of this mechanism in the WC-GCN model is that it resembles a vector with weights and employs a self-attention mechanism. Its goal is to use the correlation between words in text sentences and between sentences as learned parameters, which translate into higher-level text representations. Self-care is a widely recognized method that can dramatically increase a network's accessibility. The initial WC-GCN output is added to the Self-Attention mechanism, causing the network to concentrate on the intrinsic relationships of the features themselves and ultimately enhancing the model's capacity for self-

expression through the weighted average output. This is a generalized view of how machine learning models perform their duties in the domain of text. Next, models from literature are mentioned which are developed from time to time regarding specific problems.

Text and message categorization are essential for many real-world applications, including handling spam emails [7], ticket routing [8], article sentiment analysis [9], and more. Accurate message classification could enhance crucial situations like routing tickets in call centers based on topics [8], marking very essential alert messages in alert systems [10], and classifying incoming messages to automatically declutter emails [7] [11]. The availability of extra variables, including sender information, timestamps, attached images, audio, affiliations, and more, is the primary difference between text and message classification. However, many machine learning models are introduced to do text classification, sentiment analysis and next sentence prediction etc. Some of these models which are adopted with time to deal with text classification using NLP are as follows:

At first, Machine learning models started to grab the attention while dealing with textual datasets. For medical tasks, Deep learning models were utilized by Liang et al. [12] to extract crucial clinical data from electronic medical records. Deep learning models demonstrated great diagnostic accuracy across a number of organ systems, helping physicians analyze vast amounts of data and carry out diagnosis and evaluation. SERA is an artificial intelligence algorithm created by Goh et al. [13] that combines structured and unstructured data to predict and diagnose sepsis. Clinical data and machine learning algorithms were employed by Le et al. [14] to diagnose young heart failure in children by predicting the main causes of heart failure in children using clinical notes, based on symptoms classification. Although the application looks different yet the method of these algorithms are based on mere classification stuff as they have to rightly classify malign stuff from the benign one in order to detect a medical issue in the body.

Similarly, on text topic mining, Chen et al.'s team [15] merged the LDA and negative matrix factorization (NMF). Several studies used word vector and other computational tools to make medically assisted diagnosis of electronic medical information. Model LDA [16]. In order to create a topic model based on the categorization of medical datasets, Selvi et al. [17] used the LDA model. The LDA was utilized by Zhu et al. [18] to look into topics with various levels of hazard while the SNA was used to characterize the major keyword of adverse medical events. In order to find popular topic keywords, they merged SNA and LDA. The probable actions of doctors and nurses in these situations with various levels of harm were then examined. Sparse semantics and a lack of co-occurrence information in the special extraction of healthcare evaluations were issues that Gao et al. [19] sought to address.

Apart from medical stuff, text extraction from visual spots is also done using some other techniques. TF-IWF model is used to extract text from the visual spots as TF-IWF model's weighting mechanism can lessen the influence of texts of the same kind in the corpus on the weight of words, more precisely conveying the significance of words in the text. On TF-IWF, a lot of study has been done. The TF-IWF-IDF model, which combines IWF and term frequency-inverse document frequency (TFIDF), was proposed by Yan et al. They developed a multi-topic center upon which they constructed a two-step, single-channel algorithm. In the area of financial hotspot discovery and tracking, their program attained a high accuracy rate. A real-time Chinese text keyword extraction method based on on-screen visual hotspots was proposed by Zhang et al. [20]. They built a Chinese text extraction model using the TF-IWF, position statistical distribution, and word distance, and their model produced successful results.

With the passage of time, innovation in this domain of text, chats, emails, medical record or tweets kept on increasing. Previously mentioned models did well for their corresponding tasks, but innovation demands more of everything from a model. So, RNN and BiRNN models are tested to perform various classification tasks. The BiRNN model has the benefit that it can finish the text categorization process by combining the data on the left and right sides of the current word. In text categorization work, the BiRNN model is frequently employed. The RNN model and conditional random field (CRF) were utilized by Leevy et al. [21] to investigate related work on automatic free text recognition. To perform disease code assignment, Yu et al. [22] presented the multilayer attention bidirectional recurrent neural network (MA-BiRNN) model. In order to encode the grammatical perception representation of medical reports hierarchically, Chen et al. [23] adopted hierarchical attention BiRNN. By merging the grammatical and semantic perception representations of medical records, they finally classified breast tumors. A new model dubbed Att-BiRNN-Att, developed by Ma et al. [24], combines the BiRNN with two attention mechanisms to conduct medical answer selection.

Based on classification terms, the KTI-RNN model put out in this work broadens the scope of text classification for emails, tweets and medical notes. Additionally, KTIRNN when comprised with the following three modules: input module for LDA subject word and TF-IWF keyword extraction, output from the GA-BiRNN module, and classification module, increases the overall accuracy of the model.

Putting all models aside, the attention-based transformers architecture [25] represents a significant development in the field of NLP. This group of techniques is particularly good at identifying little relationships between words and enhancing sentence comprehension. The Bidirectional Encoder Representations from Transformers (BERT) [26] and its variants [27] [28] [29], which achieved certain benchmarks [30] [31], are a prime example. Such models are made available and simple to

use by a number of packages, including Huggingface Transformers [32], which also offers versions that have already been taught. Additionally, one can further train BERT on their data via transfer learning [33], resulting in a model that is customized for the particular task at hand.

That is why, the release of BERT [26] marked a turning point in the field of text classification. On challenging tasks like question and answer, named entity recognition, and textual entailment, the authors showed great accuracy [31]. Numerous authors have since researched enhanced architectures and modifications, including RoBERTa [27], ALBERT [28], DistilBERT [29], and others. Some models are developed to perform better on benchmark tasks, while others are lighter versions that use less computational power while maintaining competitive accuracy. Other ideas bring rival architectures to BERT (also employing transformers), such as XLNet [34] and GPT-3 [35]. Benchmarks for these models include SQuAD 2.0 [30], SuperGLUE [31], GLUE, and others.

As BERT is an open-source pre-trained model. The range of open-source code bases that make it simple for data scientists to experiment with various designs and then apply it in their applications is another reason that has contributed to the growing popularity of transformers. With a wide selection of base models and simple implementation, the Huggingface transformers package [36] is a Python library that may be used to train and improve models. Similar to many previous implementations, the GPT-3 [35] has been made available as open source and provides a practical application programming interface (API). We note the existence of other libraries for text classification without the use of machine learning, including NLTK [37], spaCy [38], and others. These are likewise simple to access and include sophisticated NLP feature extraction in addition to other text analysis capabilities.

Text classification entails a variety of tasks, each of which may be regarded as a separate academic discipline. One well-liked method is sentiment analysis, which attempts to categorize texts as positive or negative, or neutral. The survey in [39] outlines the difficulties in this field as well as the most recent advancements. Another illustration is the Spam or Ham exercise, in which participants are asked to distinguish between emails that are important and those that aren't (such as advertisements or phishing efforts) [40]. Similarly, classifying a product's category based on customer reviews, a thread's category based on postings, a restaurant's culinary specialties based on customer reviews, a product's kind based on customer reviews, an incoming email's category, and so on. As literature review has discussed already used models and methods to classify text in different ways. The aim of this research paper is to use existing pre-trained BERT model, which is discussed in the last section of literature review, along with the addition of LSTM, Pooling and Fully Connected layers.

Design and Methodology

In this chapter, the process of Bert fine tuning and use of LSTM layers in the main model is discussed. With four different types of datasets, methodology section is divided into four sub-sections out of which the last one will be discussed in the next chapter. These include pre-processing the text, preparing the data-loaders, defining and fine-tuning the BertLSTM model, and testing the model.

3.1 Pre-processing the Data

At first, using panda library, datasets, which are downloaded from the internet, are loaded. These datasets include chats, emails, and news and tweets data. All containing labels and textual body. However, first three datasets are for binary classification (either ham or spam thing) and the last one is for multi-classification (sentiment analysis of tweets). After loading all of these datasets, the very first thing is to remove unwanted text, stop words, special characters etc. from the data. To do that, I have used simple python library named import re. Then I made five classes for each dataset, four for above mentioned data and the last one for a text identifier. After cleaning the datasets, I had to map my textual labels: spam, ham, non-spam, real, fake, positive, neutral and negative into digits format either 1 or 0. After doing that each dataset is examined and further textual body and labels are separated using .values method. This is done in order to prepare data-loaders easily for the later tasks.

3.2 Preparing the data-loaders

After cleaning the datasets in the first step, datasets are prepared to make training and validation data-loaders one by one for each class. As we have five classes including four main and one testing class. The model which is being used here is a Bert_base_uncase. So Bert needs input in a special format. It requires text in the form of tokens and further in order to clarify the starting point and ending point of the text special token ([CLS], [SEP]) are added along with other tokens. Let's dig into this Bert preprocessing in detail. First of all by using Bert Tokenizer, text is encoded into tokens along with their token ids. The length, padding, truncation etc. is defined in the same encoder class. After using the Bert tokenizer, text is being fed into the encoder class which breaks each input sentence words into tokens and their corresponding token ids. During the process, text length is adjusted which is set to 40 in this paper. This length decides which text to truncate and to pad accordingly. These tokens and their corresponding token Ids are collected and stored in arrays list. Later that array list is used to make them

into tensor as model requires input in the tensor format. To do that torch.cat command is used and text is successfully converted into tokens and their token ids.

Table 1: Text into Tokens

Table 2: Special Tokens and Attention Mask

Tokens	Token IDs	Tokens	Token IDs	Attention Mask
where	2073	[CLS]	101	1
s	1055	i	1045	1
my	2026	m	1049	1
boy	2879	home	2188	1
##toy	29578	[SEP]	102	1
i	1045	[PAD]	0	0
miss	3335	[PAD]	0	0
you	2017	[PAD]	0	0
what	2054	[PAD]	0	0
happened	3047	[PAD]	0	0

Table 1 shows the very first step of Bert tokenizer where it converts every word in a sentence into a special token format which can be seen. In table 2, further data is prepared to be fed to Bert model. As Bert takes input data in a special format so that is why, special token [CLS] and [SEP] are added in every input sentence. The purpose of these special tokens is just to let the model know about the starting and ending point of the input sentence. CLS with token ID 101 tells Bert model that a new sentence is about to start while SEP with token ID 102 tells Bert model that sentence is ending. Further padding is done to make each sequence of same size in a batch, it helps in resolving any mismatch issues in later stages. After adding padding, attention mask addition becomes important because it helps the model identify which tokens to consider for training and which ones to drop. As special and input sentence tokens are labelled with attention mask 1 while the padding is labelled with attention mask 0. In this way, model knows that it has to avoid including padding in the training process. Even it has no impact with the training procedure as it is just a series of zeros.

As Bert model only takes special token IDs and attention masks of every sentence along with special tokens. The points is, after converting every sentence into tokens and attention masks. The data looks

rouge. Now, the tensor making thing comes in the lime light. These tensors store all the converted sentences token IDs and attention masks in a big array holding sub array of 40 sequence length. These tensor can be observed in the figure 1.

```

tensor([[ 101, 2175, 2127, ..., 0, 0, 0],
        [ 101, 7929, 2474, ..., 0, 0, 0],
        [ 101, 2489, 4443, ..., 4374, 4443, 102],
        ...,
        [ 101, 12063, 2001, ..., 0, 0, 0],
        [ 101, 1996, 3124, ..., 102, 0, 0],
        [ 101, 20996, 10258, ..., 0, 0, 0]])
tensor([[1, 1, 1, ..., 0, 0, 0],
        [1, 1, 1, ..., 0, 0, 0],
        [1, 1, 1, ..., 1, 1, 1],
        ...,
        [1, 1, 1, ..., 0, 0, 0],
        [1, 1, 1, ..., 1, 0, 0],
        [1, 1, 1, ..., 0, 0, 0]])
tensor([0, 0, 1, ..., 0, 0, 0])

```

Figure 1: Text Tokens into Tensors

As about tensors mentioned above. Each portion of text, main text body has its first token IDs tensors then second attention mask tensors and then third label tensors. This process of tokenization is performed for all datasets one by one and they are stored with corresponding variable names. Now, data is almost ready to be fed into the Bert model but one thing is remaining which is splitting the whole data of each target class into training and validation datasets and then assigning a batch size, 16 for binary and 32 for multi-label classes. This batch size ensures that from tensors, 16 and 32 lines are to be fed to the main model in parallel. As Bert is a big pre-trained model and thus can handle multiple lines at a time in parallel formation.

Train and test split is performed using train-test-split library of Scikit-Learn. Validation is set to 0.2 which means about 20% of shuffled data from the main tensor is to be selected for validation section and remaining for the training section. After that dataloaders are made using the above mentioned split and for training phase random sampler is selected in order to avoid overfitting in the model while sequential sampler is selected for the validation phase as it only needs to be tested on the trained model. Now, data loaders phase has been completed successfully. The next step is to define Bert and LSTM customized models for all mentioned target classes.

3.3 Defining the BertLSTM model

After preparing data loaders, it is time to define Bert and LSTM models one by one for all target classes. As Bert can be fine-tuned separately and then LSTM layers can be embedded on top of Bert model but doing so creates an issue which is the poor accuracy of LSTM layers as Bert embedding hardly leave some space for other layers to further capture the accuracy. However, by embedding the LSTM layer along with Bert, it directly influences the gradients update of Bert model and hence accuracy of the model is increased. Following are the steps of defining and fine tuning both Bert and LSTM model together.

- Custom classifier BertLSTM is defined and further Bert base model and LSTM layer is defined in it.
- Parameters of LSTM layer, hidden size, input size etc. are defined inside the classifier.
- Scheduler and optimizer is defined and then models (chats, news, emails, tweets and identify) with their corresponding data loaders are called.
- During forward pass, Bert output embedding's are calculated and then fed to LSTM to get output logits.
- Loss is calculated based on these logits during training process.
- Then optimizer and scheduler is adjusted accordingly using. Step () method in the backward pass.
- In this way gradients are adjusted for the whole model as LSTM layers play their role to influence Bert embedding's.
- After training, validation phase is activated which runs the validation data loaders of each class and checks the accuracy of the model.
- Based on the training loss and validation accuracy, model is fine-tuned.
- Hidden size, Number of layers, Learning rate in optimizer, dropout ratio and batch size is adjusted using hit and trial method.

Implementation and results

In Methodology chapter all prior procedures before implantation has been performed. In this chapter all trained models (model_chats, model_emails, model_news and model_tweets) are implemented inside another model named model_identify. This model is trained on the all above models data and it classifies the input text into either of these four models and further that model is using to further classify that input text.

We are sharing all five models accuracies one by one and then implantation phase is observed.

4.1 Chats Model

Chats model uses following parameters to perform Bert LSTM fine tuning.

- Sequence length =40
- Batch size = 16
- Number of labels = 2
- LSTM hidden size = 512
- Learning rate = 5e-5
- Input size to LSTM = 768
- Gradient clipping = 1

```
Training: 100%|██████████| 279/279 [00:42<00:00, 6.62it/s]
Validation: 100%|██████████| 70/70 [00:02<00:00, 26.26it/s]
```

```
Epoch 3/3
Training Loss: 0.0066
Validation Accuracy: 99.55%
Validation F1-score: 99.02%
```

Figure 2: Validation Accuracy for Chats Classification

4.2 Emails Model

Emails model uses following parameters to perform Bert LSTM fine tuning.

- Sequence length =40
- Batch size = 16
- Number of labels = 2
- LSTM hidden size = 512
- Learning rate = 5e-5
- Input size to LSTM = 768
- Gradient clipping = 1

```
Training: 100%|██████████| 259/259 [00:38<00:00, 6.68it/s]
Validation: 100%|██████████| 65/65 [00:02<00:00, 25.30it/s]
```

```
Epoch 3/3
Training Loss: 0.0035
Validation Accuracy: 99.03%
Validation F1-score: 98.82%
```

Figure 3: Validation Accuracy for Emails Classification

4.3 News Model

News model uses following parameters to perform Bert LSTM fine tuning.

- Sequence length =40
- Batch size = 16
- Number of labels = 2
- LSTM hidden size = 512
- Learning rate = 5e-5
- Input size to LSTM = 768
- Gradient clipping = 1

```
Training: 100%|██████████| 316/316 [00:47<00:00, 6.71it/s]
Validation: 100%|██████████| 79/79 [00:03<00:00, 25.10it/s]
```

```
Epoch 3/3
Training Loss: 0.0445
Validation Accuracy: 94.69%
Validation F1-score: 94.69%
```

Figure 4: Validation Accuracy for News Classification

4.4 Tweets Model

Tweets model uses following parameters to perform Bert LSTM fine tuning.

- Sequence length =40
- Batch size = 32
- Number of labels = 3
- LSTM hidden size = 512
- Learning rate = 3e-5
- Input size to LSTM = 768
- Gradient clipping = 1

```
Training: 100%|██████████| 366/366 [01:30<00:00, 4.05it/s]
Validation: 100%|██████████| 92/92 [00:06<00:00, 13.26it/s]

Epoch 3/3
Training Loss: 0.1937
Validation Accuracy: 85.18%
Validation F1-score: 80.57%
```

Figure 5: Validation Accuracy for Tweets Classification

4.5 Text Identifying Model

Text identifying model uses following parameters to perform Bert LSTM fine tuning.

- Sequence length =40
- Batch size = 16
- Number of labels = 4
- LSTM hidden size = 512
- LSTM layers = 2
- Dropout = 20%
- Learning rate = 3e-5
- Input size to LSTM = 768
- Gradient clipping = 1

```
Training: 100%|██████████| 373/373 [00:58<00:00, 6.41it/s]
Validation: 100%|██████████| 94/94 [00:04<00:00, 23.19it/s]

Epoch 3/3
Training Loss: 0.0372
Validation Accuracy: 96.24%
Validation F1-score: 96.74%
```

Figure 6: Validation Accuracy for Text Identifier Classification

4.6 Results

Input text is first converted into token IDs and attention masks and then it is fed to Text identifying model. Text identifying model identifies the text type: chat, email, news or tweet. Then that respective model is called and further binary or multi-classification result is calculated. However, how all models perform on an input text while performing respective classification is as follows:

```
Input Sentence: @VirginAmerica it's really aggres
Text is identified as class: Tweet
Predicted Class for Chats: Non-spam
Predicted Class for Emails: Spam
Predicted Class for News: Fake
Predicted Class for Tweets: Negative
```

Figure 7: Implementation Result of all five classification models

Table 3: Binary Classification Comparison with Other Models

Model (2 Label Classification)	F1 score	Accuracy
FastText	73.31	73.12
TextCNN	80.67	80.93
BiRNN	80.02	81.68
TextRNN	75.99	78.76
RCNN	80.90	78.41
HAN	82.63	82.62
Attention-BiRNN	83.44	83.42
KTI-RNN	85.57	85.59
BERT-LSTM	94.69	94.69

Table 4: Multi Classification Comparison with Other Models

Model (3 Label Classification)	Accuracy
BERT-FC	66
BERT-Metadata-FC	65
BERT-Random Forest Classifier	61
BERT-Metadata-Random Forest Classifier	61
BERT-LSTM	85.18

Conclusion and Future Work

Bert fine tuning and LSTM layers inclusion have provided better accuracy for classification task. As discussed in this paper, three different binary classification classes and two multi-classification classes are used to fine tune Bert and LSTM model. Their accuracy clearly shows that this combined model has performed better than previous techniques where pre trained Bert along with other layers: fully connected layers, random forest classifier, concatenated Meta data and additional fully connected layers, and concatenated Meta data with random forest classifier is used. It shows that this method of using Bert along with LSTM layers is proven worthy. The accuracy for all mentioned classes have improved than mere use of Bert fine tuning. This concludes that using Bert along with LSTM layers is an effective method to make an application based on classification as this customized fine-tuned model achieves professional level accuracy.

5.1 Future Work

Whereas Bert and LSTM structure have provided an edge there lies a room for improvement as well. When the dataset is complex and has multi-labels, then only fine tuning of Bert, although, is not enough yet LSTM layers cannot perform that effectively either. It is because Bert itself being a large pre trained model captures enough features from the data and it hardly leaves some space for LSTM layers to further capture the features from the data. However, it also lies in the method we have used in this paper, amalgamation of LSTM with Bert, in some other approach one might work on fine tuning Bert and LSTM separately on the same data and then use some emerging process to increase the overall accuracy. However, for complex and multi-label classification, improvements can be made.

References

- [1] "S. Schmidt, S. Schnitzer, and C. Rensing, "Text classification based filters for a domain-specific search engine," *Computers in Industry*, vol. 78, pp. 70–79, 2016."
- [2] "B. Liu, X. Li, W. S. Lee, and P. S. Yu, "Text classification by labeling words," *Artificial Intelligence*, vol. 34, pp. 425–430, 2004."
- [3] "Y. Yu, M. Li, L. Liu, Z. Fei, F.Wu, and J.Wang, Automatic ICD code assignment of Chinese clinical notes based on multilayer attention BiRNN, *Journal of Biomedical Informatics*, vol. 91, p. 103114, 2019."
- [4] "Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018."
- [5] "J. L. Leevy, T. M. Khoshgoftaar, and F. Villanustre, Survey on RNN and CRF models for de-identification of medical free text, *Journal of Big Data*, vol. 7, no. 1, pp. 1–22, 2020."
- [6] "D. Y. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in Neural Information Processing Systems*, vol. 16, pp. 321–328, 2004."
- [7] "Asif Karim, Sami Azam, Bharanidharan Shanmugam, and Krishnan Kannoorpatti. Efficient clustering of emails into spam and ham: The foundational study of a comprehensive unsupervised framework. *IEEE Access*, 8:154759–154788, 2020."
- [8] "Jianglei Han, Jing Li, and Aixin Sun. Uftr: A unified framework for ticket routing, 2020."
- [9] "Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113, 2014."
- [10] "Dheeraj Gupta, P.S. Joshi, A.K. Bhattacharjee, and R.S. Mundada. Ids alerts classification using knowledge-based evaluation. In 2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012), pages 1–8, 2012."
- [11] "Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Machine Learning: ECML 2004*, pages 217–226, Berlin, Heidelberg,".
- [12] "H. Liang, B. Y. Tsui, H. Ni, C. C. S. Valentim, S. L. Baxter, G. Liu, W. Cai, D. S. Kermany, X. Sun, J. Chen, et al., Evaluation and accurate diagnoses of pediatric diseases using artificial intelligence, *Nature Medicine*, vol. 25, no. 3, pp. 433–438, 2019."
- [13] "K. H. Goh, L. Wang, A. Y. K. Yeow, H. Poh, K. Li, J. J. L. Yeow, and G. Y. H. Tan, Artificial intelligence in sepsis early prediction and diagnosis using unstructured data in healthcare, *Nature Communications*, vol. 12, no. 1, pp. 1–10, 2021"
- [14] "T. -D. Le, R. Noumeir, J. Rambaud, G. Sans, and P. Juvet, Detecting of a patient's condition from clinical narratives using natural language representation, *arXiv preprint arXiv: 2104.03969*, 2021."
- [15] "Y. Chen, H. Zhang, R. Liu, Z. Ye, and J. Lin, Experimental explorations on short text topic mining between LDA and NMF based schemes, *Knowledge-Based Systems*, vol. 163, pp. 1–13, 2019."
- [16] "X. -Y. Jin, D. -X. Pu, Y. -Z. Lan, and L. -J. Li, Medical aided diagnosis using electronic medical records based on LDA and word vector model, in *Proc. 2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, Changsha,".
- [17] "M. Selvi, K. Thangaramya, M. S. Saranya, K. Kulothungan, S. Ganapathy, and A. Kannan, Classification of medical dataset along with topic modeling using LDA, in *Nanoelectronics, Circuits and Communication Systems*, V. Nath and J. K. Mandal, eds. Singapor"
- [18] "L. Zhu, I. Reyhchav, R. McHaney, A. Broda, Y. Tal, and O. Manor, Combined SNA and LDA methods to understand adverse medical events, *International Journal of Risk & Safety in Medicine*, vol. 30, no. 3,".
- [19] "H. Y. Gao, J. W. Liu, and S. X. Yang, Identifying topics of online healthcare reviews based on improved LDA, (in Chinese), *Transactions of Beijing Institute of Technology*, vol. 39, no. 4, pp. 427–434, 2019."
- [20] "T. Zhang, W. Wang, Y. Huang, K. Liu, and X. Hu, Method of real-time keyword extraction from Chinese

- short-text based on visual hotspot on screen, (in Chinese), Journal of the China Society for Scientific and Technical Information, vol. 35, no. 12, pp."
- [21] "J. L. Leevy, T. M. Khoshgoftaar, and F. Villanustre, Survey on RNN and CRF models for de-identification of medical free text, Journal of Big Data, vol. 7, no. 1, pp. 1–22, 2020."
- [22] "Y. Yu, M. Li, L. Liu, Z. Fei, F.Wu, and J.Wang, Automatic ICD code assignment of Chinese clinical notes based on multilayer attention BiRNN, Journal of Biomedical Informatics, vol. 91, p. 103114, 2019."
- [23] "D. Chen, M. Huang, and W. Li, Knowledge-powered deep breast tumor classification with multiple medical reports, IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 18, no. 3,"
- [24] "J. Ma, C. Che, and Q. Zhang, Medical answer selection based on two attention mechanisms with BiRNN, MATEC Web of Conferences, vol. 176, no. 8, p. 01024, 2018."
- [25] "Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017."
- [26] "Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018."
- [27] "Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019."
- [28] "Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019."
- [29] "Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019."
- [30] "Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018."
- [31] "Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In In the Proceedings of ICLR. 2019."
- [32] "Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le S"
- [33] "Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10):1345–1359, 2009."
- [34] "Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach,"
- [35] "Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh,"
- [36] "Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le S"
- [37] "Steven Bird, Ewan Klein, and Edward Loper. Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc.", 2009."
- [38] "Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017"
- [39] "Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. Ain Shams engineering journal, 5(4):1093–1113, 2014."
- [40] "Asif Karim, Sami Azam, Bharanidharan Shanmugam, and Krishnan Kannoorpatti. Efficient clustering of emails into spam and ham: The foundational study of a comprehensive unsupervised framework. IEEE Access, 8:154759–154788, 2020."