FINE TUNING OF BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS FOR OPTIMIZING NEWS CLASSIFICATION

Term paper report submitted in fulfillment of the requirements for the Degree of

Bachelor of Technology in Computer Science and Engineering

By

NITIN CHAUDHARY

(9921103163)

Course Name: Introduction to Contemporary Research and Term Paper Writing

Course Code: 24B12GE411

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

(Declared Deemed to be University U/S 3 of UGC Act)

A-10, SECTOR-62, NOIDA, INDIA(12)

December 2024

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and beliefs, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma from a university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: NOIDA

Date: 13-12-2024

Name: NITIN CHAUDHARY

Enrollment No.: 9921103163

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B.Tech CSE Term Paper Report entitled **"Fine Tuning of Bidirectional Encoder Representations from Transformers for Optimizing News Classification",** submitted by **Nitin Chaudhary (9921103163)** of **Jaypee Institute of Information Technology, Noida** is a bonafide record of his original work carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.

(Signature of Supervisor)

Dr. Himanshu Agarwal (Ph.D.)
Associate Professor
Department of Mathematics
Jaypee Institute of Information Technology
Date:
13-12-2024

ABSTRACT

This term paper report discusses fine-tuning of pre-trained Bidirectional Encoder Representations from Transformers (BERT) model to use for multi-class text classification. Exploring the need to classify news text into categories like health, technology, economy, etc. BERT has been selected to do the task of classification due to its excellent ability to understand contextual relationships in text as it leverages pre-trained language knowledge, and handles long text dependencies. A comparison has been made in this term paper with five other models during the model selection step. It is found that BERT outperforms all other models after fine-tuning in terms of precision, accuracy, recall and F1 score. This happens due to key features of BERT including Bidirectional Contextual Understanding, Pretraining and Fine-tuning, etc. This term paper discusses Machine Learning life cycle steps including Data Preparation (Pre-ML), Model Training & Inference (In-ML), and Model Management (Post-ML) by practical implementation of fine-tuning of BERT. BERT is trained on the dataset of news articles averaging 200–300 words, which are labeled correctly before training; this results in high-performance classification accuracy. Results tell us how BERT has achieved a state-of-the-art level of performance with its effectiveness and flexibility in real-world text classification. This work points out the ability of BERT in Natural Language Processing (NLP) tasks and establishes a foundation for further work on model optimization.

ACKNOWLEDGEMENT

I would like to place on record my deep sense of gratitude to Dr. Himanshu Agarwal, Associate Professor, Jaypee Institute of Information Technology, India for his generous guidance, help and useful suggestions. I express my sincere gratitude to his stimulating guidance, continuous encouragement and supervision throughout the course of present work. I also wish to extend my thanks to Dr. Anuj Bhardwaj, Dr. Neha Singhal, Prof. Anirban Pathak and Prof. Alka Tripathi for their insightful instructions to improve the quality of this project work.

Signature(s) of Students

Name: NITIN CHAUDHARY
Enrollment No.: 9921103163

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER-1: INTRODUCTION

1.1 General Introduction:

In the vast expanse of data-driven applications, machine learning has emerged as a key technology that has driven significant progress in a wide array of domains ranging from business intelligence to healthcare, e-commerce, and scientific research. The ability to analyze large amounts of data by using advanced or deep analytics that are sophisticated Machine Learning (ML) algorithms has indeed revolutionized the way we draw meaningful insights and then make informed data-driven decisions. This term paper looks at two of the key features in ML: Data Management [1] and fine tuning of the Bidirectional Encoder Representations from Transformers (BERT) [5] model for multi-class text classification to show the entire ML lifecycle from pre ML to post ML.

The high-performance level of machine learning models, especially those based on deep learning architectures, requires large amounts of data and substantial computational power. All steps, from preparing the data, to training models, and further managing models after training, pose different challenges that complicate the overall workflow. Traditional methods of machine learning focused more on the feature engineering process and used classical algorithms like Naive Bayes [12] and Support Vector Machine (SVM) [13]. They quite often faced the problems which rendered them ineffective to fully capture the subtleties in place in natural language and many other types of data too. Deep learning has witnessed the maximum advancement of highly advanced models in place. It also covers models such as Long Short-Term Memory (LSTM) [11] networks and Convolutional Neural Network (CNN) [11]. The former outperform the latter in measures like performance metrics, though they demonstrate a weakness when trying to understand the context in both ways. This is actually a quite important concept for getting a good understanding of the language processing tasks.

A breakthrough transformer-based architecture, and a far more precise entity, BERT, for Bidirectional Encoder Representations from Transformers, has set the scene for a most significantly transformative shift in the practice of natural language processing commonly termed NLP [10]. BERT [9] learns dynamic contextual embeddings by drawing upon both left and right contexts, and is thus powerfully effective for a great deal of NLP [10] tasks that include, but are by no means limited to text classification.

This term paper mainly focuses on thorough study and exploration of the various techniques of data management and in particular, there is emphasis on practical usage and focus on the technique to fine-tune BERT since it is a very potent model, which can be useful in the process of classification tasks involving text.

1.2 Problem Description

The task is to fine-tune a BERT [5] pre-trained model in order to achieve higher accuracy for multi-class text classification of news articles into one out of 26 categories. BERT is selected due to its several advantages compared to traditional methods due to its bidirectional approach, it captures the context of each word more effectively than the unidirectional models such as LSTM [5] and Recurrent Neural Network (RNN) [11]. Its pre-training on a massive corpus provides a good base while its fine-tuning makes it reach state-of-the-art performance [10]. It excels at capturing long-range dependencies, crucial for the understanding of complex sentences within news articles, and can consistently achieve higher accuracy and F1 scores compared with traditional models such as SVM [13] and Naive Bayes [12].

The main processing steps required for fine tuning BERT pre-trained model:

1.2.1 Text Cleaning & Preprocessing: Need to remove punctuation, URLs, special characters, and stop words and to tokenize and lemmatize or lower casing of text.

Example:

Input: "python courses python courses, python exercise, python exercises with solutions: learn the python programming language. python is a programming language high level."

Expected Output after preprocessing: "python course python course python exercise python exercise solution learn python programming language python programming language high level"

1.2.2 Feature Representation: Need to transform tokenize text into a numerical format which can be processed by the BERT. This requires creating some input representations like input_ids, attention_mask, and token_type_ids.

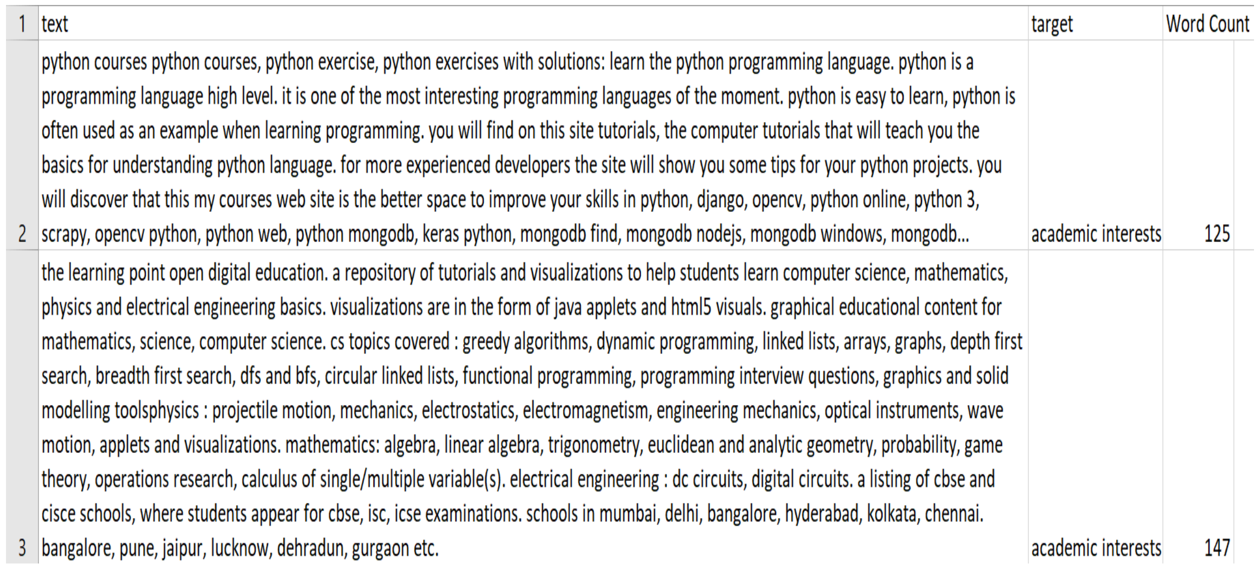
1.2.3 Fine-tuning the BERT Model: Key parameters for Fine-tuning are learning rate, batch size, number of epochs etc. to achieve higher accuracy of model.

CHAPTER 2: LITERATURE REVIEW AND IMPLEMENTATION

2.1 Data Preprocessing before Applying Machine Learning

2.1.1 Data Discovery

Data discovery is the very first step in the machine learning pipeline. Here, the appropriate data is identified and taken from kaggle. The training dataset contains text, target label and word count which need to be further cleaned before being used for training BERT. This has been done by relying on manual efforts or keyword searches, which consume lots of time and are likely to have errors. Advanced automated techniques for data discovery have emerged as web scraping, data lakes, and data warehouses. For instance, Infogether [1] relies on a massive corpus of HTML tables in the web to fill missing attributes in a base table. In a similar manner, SmartCrawl [1] fetches new attributes dynamically from external databases using keyword-based APIs. Automated data discovery is efficient and ensures that the data is comprehensive and relevant for a particular ML task involved.



1	text	target	Word Count
2	python courses python courses, python exercise, python exercises with solutions: learn the python programming language. python is a programming language high level. it is one of the most interesting programming languages of the moment. python is easy to learn, python is often used as an example when learning programming. you will find on this site tutorials, the computer tutorials that will teach you the basics for understanding python language. for more experienced developers the site will show you some tips for your python projects. you will discover that this my courses web site is the better space to improve your skills in python, django, opencv, python online, python 3, scrapy, opencv python, python web, python mongodb, keras python, mongodb find, mongodb nodejs, mongodb windows, mongodb...	academic interests	125
3	the learning point open digital education. a repository of tutorials and visualizations to help students learn computer science, mathematics, physics and electrical engineering basics. visualizations are in the form of java applets and html5 visuals. graphical educational content for mathematics, science, computer science. cs topics covered : greedy algorithms, dynamic programming, linked lists, arrays, graphs, depth first search, breadth first search, dfs and bfs, circular linked lists, functional programming, programming interview questions, graphics and solid modelling toolsphysics : projectile motion, mechanics, electrostatics, electromagnetism, engineering mechanics, optical instruments, wave motion, applets and visualizations. mathematics: algebra, linear algebra, trigonometry, euclidean and analytic geometry, probability, game theory, operations research, calculus of single/multiple variable(s). electrical engineering : dc circuits, digital circuits. a listing of cbse and cisce schools, where students appear for cbse, isc, icse examinations. schools in mumbai, delhi, bangalore, hyderabad, kolkata, chennai. bangalore, pune, jaipur, lucknow, dehradun, gurgaon etc.	academic interests	147

Fig. 1 News data set before cleaning

2.1.2 Data Cleaning

Data cleaning is the removal of inconsistencies, irrelevant information, and errors from raw data to make it better quality for machine learning models. Traditional rule-based methods are easy to implement but lack precision and flexibility. Modern approaches focus more on task-specific cleaning to improve the performance of the models. For example, ActiveClean [3] focuses on cleaning within budget and BoostClean[1] uses techniques through the boosting algorithms to figure the correct repair sequences in the processes. There exist versions as CPClean [1], by preferring records that influence predictions over model performance, while in DAGAN, an improved reduction of distributional divergence to minimize prediction errors and reduce discrepancies between training and test data. These would allow the data to end up being of high quality leading towards better model performance.

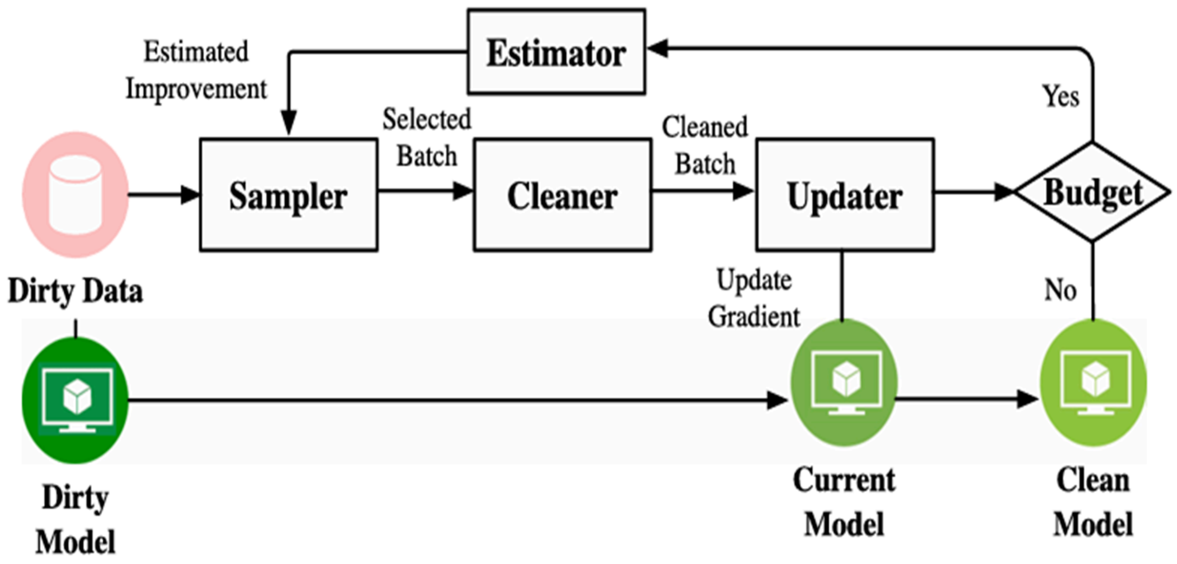
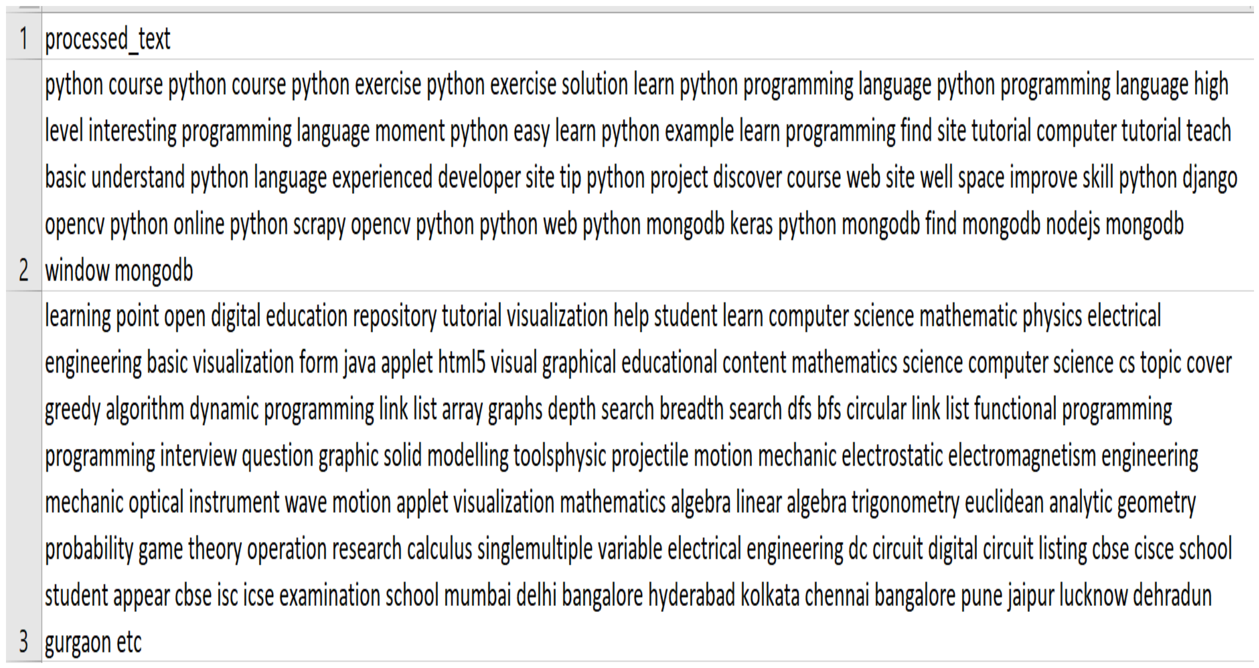


Fig. 2 Active Clean Framework

In this approach, data cleaning involves removing URLs, special characters, and unnecessary white spaces from text. Apart from tokenization and lemmatization- the removal of useless words that are stripped from their base forms-this cleaning operation is parallelized by MPI in order to split the workload into multiple processes for faster execution. Every process will pick up a portion of data, clean it, and combine afterwards to provide higher quality data for efficiently predicting the news class using fine tuned BERT.



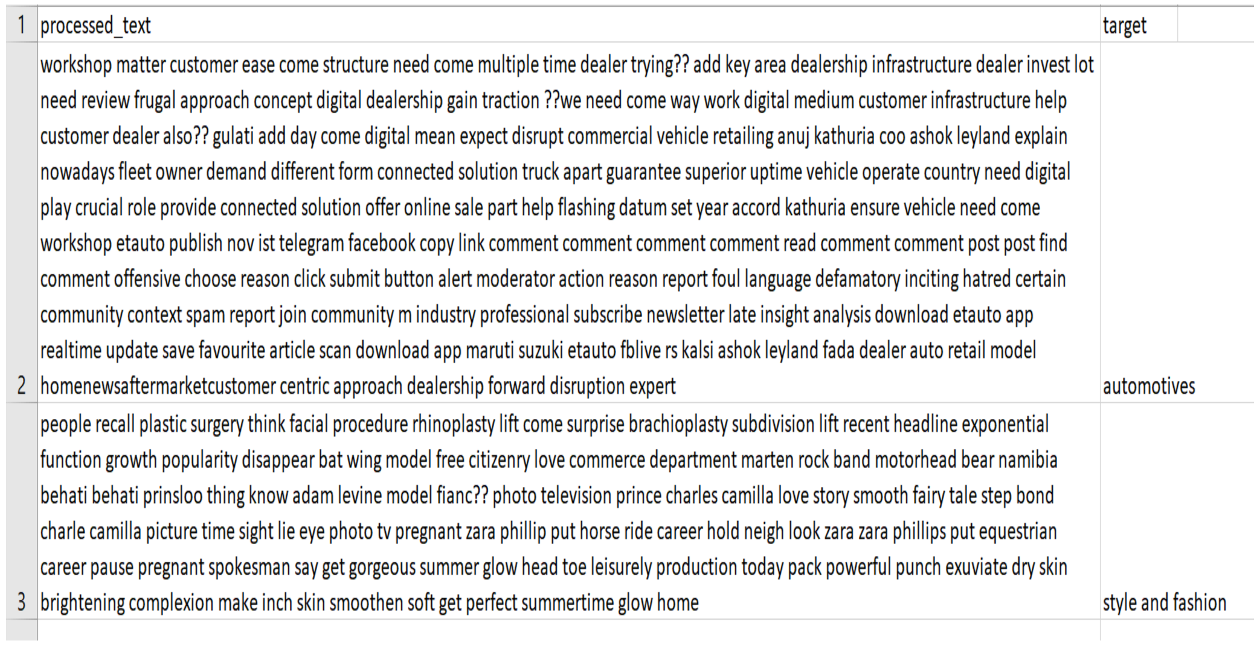
1	processed_text
2	python course python course python exercise python exercise solution learn python programming language python programming language high level interesting programming language moment python easy learn python example learn programming find site tutorial computer tutorial teach basic understand python language experienced developer site tip python project discover course web site well space improve skill python django opencv python online python scrapy opencv python python web python mongodb keras python mongodb find mongodb nodejs mongodb window mongodb
3	learning point open digital education repository tutorial visualization help student learn computer science mathematic physics electrical engineering basic visualization form java applet html5 visual graphical educational content mathematics science computer science cs topic cover greedy algorithm dynamic programming link list array graphs depth search breadth search dfs bfs circular link list functional programming programming interview question graphic solid modelling toolsphysic projectile motion mechanic electrostatic electromagnetism engineering mechanic optical instrument wave motion applet visualization mathematics algebra linear algebra trigonometry euclidean analytic geometry probability game theory operation research calculus singlemultiple variable electrical engineering dc circuit digital circuit listing cbse cisce school student appear cbse isc icse examination school mumbai delhi bangalore hyderabad kolkata chennai bangalore pune jaipur lucknow dehradun gurgaon etc

Fig. 3 News dataset text after cleaning

2.1.3 Data Labeling

In the case of supervised learning, the quality of labeling will contribute the most to the model. The conventional methods relied on labor-intensive and costly manual labeling. With the coming of crowdsourcing platforms like Amazon Mechanical Turk [1], labeling data has been easy, though it is a matter of quality. Some of the active learning techniques are uncertainty sampling and query-by-committee, where experts label the most informative examples to enhance the quality of labeling. For example, weak supervision approaches Snorkel creates weak labels by using different labeling functions combined to give the final labels.

High-quality labeled data is necessary for good model performance for tasks like multi-class text classification using BERT. In this implementation, we depend on pre-labeled datasets in which every text sample has been assigned to one of 26 categories. Good labeling is thus important for ensuring the BERT model learns patterns and relationships in the data.



	processed_text	target
1	processed_text	target
2	workshop matter customer ease come structure need come multiple time dealer trying?? add key area dealership infrastructure dealer invest lot need review frugal approach concept digital dealership gain traction ??we need come way work digital medium customer infrastructure help customer dealer also?? gulati add day come digital mean expect disrupt commercial vehicle retailing anuj kathuria coo ashok leyland explain nowadays fleet owner demand different form connected solution truck apart guarantee superior uptime vehicle operate country need digital play crucial role provide connected solution offer online sale part help flashing datum set year accord kathuria ensure vehicle need come workshop etauto publish nov ist telegram facebook copy link comment comment comment comment read comment comment post post find comment offensive choose reason click submit button alert moderator action reason report foul language defamatory inciting hatred certain community context spam report join community m industry professional subscribe newsletter late insight analysis download etauto app realtime update save favourite article scan download app maruti suzuki etauto fblive rs kalsi ashok leyland fada dealer auto retail model homenewsaftermarketcustomer centric approach dealership forward disruption expert	automotives
3	people recall plastic surgery think facial procedure rhinoplasty lift come surprise brachioplasty subdivision lift recent headline exponential function growth popularity disappear bat wing model free citizenry love commerce department marten rock band motorhead bear namibia behati behati prinsloo thing know adam levine model fianc?? photo television prince charles camilla love story smooth fairy tale step bond charle camilla picture time sight lie eye photo tv pregnant zara phillip put horse ride career hold neigh look zara zara phillips put equestrian career pause pregnant spokesman say get gorgeous summer glow head toe leisurely production today pack powerful punch exuviate dry skin brightening complexion make inch skin smoothen soft get perfect summertime glow home	style and fashion

Fig. 4 Labeled dataset for Supervised training of BERT

2.2 In-ML Model Training and Inference in the Machine Learning Context

2.2.1 Feature Selection

Feature selection is one of the most important stages of the ML pipeline that revolves around finding and using the most useful features to improve model training. Filtering and wrapper-based methods were expensive initially because selections could be inaccurate unless done manually with extra care. Recent developments have included automatic methodologies for feature selection. For example, Columbus [3] applies batching and materialization strategies to reduce the I/O costs of testing feature subsets effectively. Helix [1] optimizes the complete ML workflow with specific feature engineering using materialization techniques. These techniques try to automate and speed up feature selection so that the trained models are on the most relevant feature. This automates feature selection, reducing manual effort required and streamlines the ML pipeline in increased efficiency and effectiveness.

Feature Extraction occurred implicitly in using the BERT tokenizer. The BERT tokenizer takes text and turns it into its own WordPiece embeddings that carry rich contextual information and semantic relations text. During fine-tuning, BERT dynamically identifies the most relevant features for a specific task to focus on, like important tokens or phrases that help in classification decisions. It automatically streamlines feature selection, reduces manual effort, and ensures that the model captures the most meaningful information for improved classification accuracy.

2.2.2 Model Selection

Model selection refers to the process of choosing an appropriate architecture for the model and then defining specific hyperparameters to a task. Although traditional algorithms, such as grid search and random search, compute a lot, scientists have recently developed efficient ways for model selection: AutoWeka [3] and MLbase [3], which adopt multi armed bandit algorithms to automate the selection of an algorithm and the tuning of hyperparameters in an adaptive way. Longview integrates model management functionality into the DBMS [3] with the aim of automating the selection of models. The methods attempt to optimize the process of model selection such that only the best models are selected for training. Leverage advanced optimization techniques through which modern approaches of model selection ensure better and more efficient identification of the best models for a given ML task.

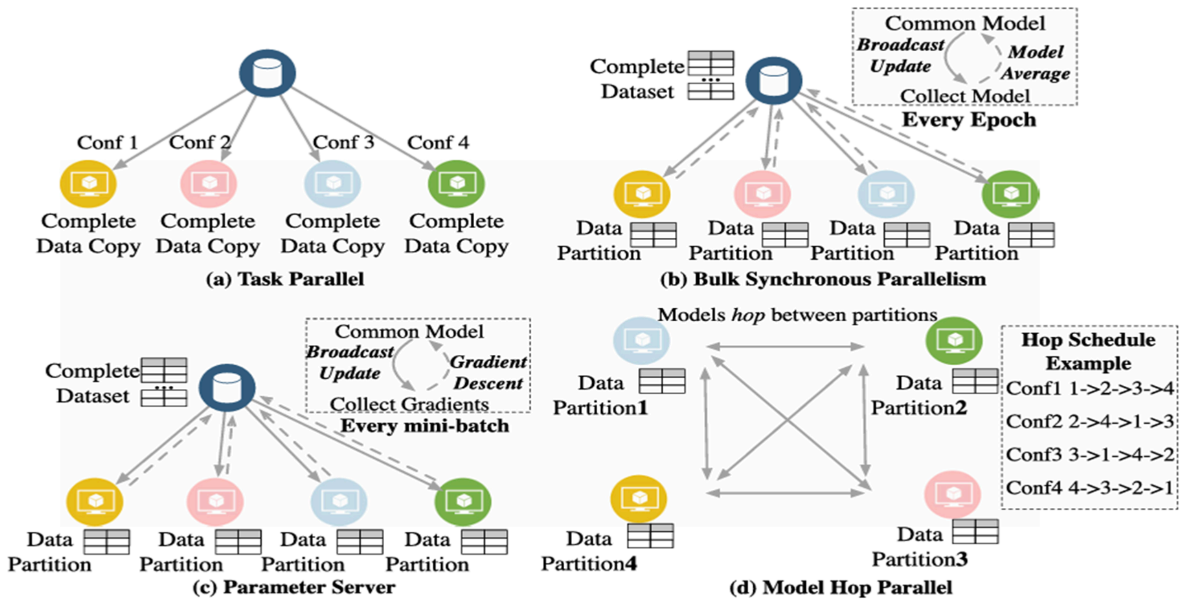


Fig. 5 Database Techniques for Model Selection

2.2.2.1 Model selection for news classification:

For classification of news articles into 26 categories, BERT is chosen because of its exceptional performance and its advantage over traditional ML models as well as other deep learning methods.

Why BERT?

Bidirectional Contextual Understanding: Because of its bidirectional approach, BERT captures the context of every word, it understands both its prior words and succeeding words or simply it reads text from both left-to-right and right-to-left simultaneously, ensuring a deeper understanding of language. This makes it even more effective than unidirectional models like LSTM and RNN.

Pretraining and Fine-tuning: BERT is pre-trained on a colossal corpus of text (BooksCorpus and English Wikipedia) and can be fine-tuned on specific tasks with minimal data. This makes it both powerful and efficient for the task of classification, where it can quickly adapt to new contexts with limited task-specific training.

Long Dependencies: BERT is also excellent in handling long dependencies in the text, which is actually necessary to understand complex sentences in news articles.

Performance: BERT achieves state-of-art results in NLP tasks because it surpasses the traditional SVM and Naive Bayes-based models, and even surpasses some deep learning models, specifically the LSTM and CNN models in particular with respect to their accuracy and F1 scores.

Comparison of BERT with 5 other models:

Feature/Model	**BERT** (Bidirectional Encoder Representations from Transformers)	**SVM** (Support Vector Machine)	**Naive Bayes**	**LSTM** (Long Short-Term Memory)	**CNN** (Convolutional Neural Network)	**RNN** (Recurrent Neural Network.)
Contextual Understanding	Excellent (Bidirectional)	Poor	Poor	Good (Unidirectional)	Moderate	Good (Unidirectional)
Pre-training	Yes	No	No	No	No	No
Fine-tuning	Yes	No	No	Yes	Yes	Yes
Handling Long Dependencies	Excellent	Poor	Poor	Good	Poor	Moderate
Performance on NLP Tasks	Excellent (State-of-the-art)	Moderate	Moderate	Good	Good	Good

Table 1 Model selection comparison

2.2.3 Training Acceleration

This training acceleration actually accelerates the train process of ML models. Most traditional methods depend on single-threaded executions, being very slow and inefficient; new progress brought parallel computing and distributed computing techniques with the aim of accelerating training processes. For example, SystemML utilizes fused operators and code generation to minimize the creation of intermediates and optimize I/O of matrix computation. TensorFlow and GraphLab provide asynchronous computation to the iterative ML algorithms, which remove global locking to converge better.

All the training for fine tuning of BERT is done on Kaggle GPU, thereby reducing much processing time when dealing with large data sets of news text.

2.3 Model Management (Post-ML)

2.3.1 Storage and Versioning

Model storage and versioning would store the trained models and track various differences between versions for analysis and deployment. Traditional methodologies tend to depend on human effort, thus the more that humans are involved the error-prone and inefficient the system becomes. The newly proposed system has introduced automated approaches to model storage and versioning. For instance, using ModelDB [1], one tracks and indexes ML models concerning ModelHub[3] management of deep learning models but with the possibility of versions and querying. It brings down and automates model versioning as well as storage and which then leads to improvement in the use of the models. These systems offer provisions for an orderly approach to automatic storage and versioning in models in a manner of improving cycle efficiency as well as the reliability of machine learning.

The fine-tuned BERT model is saved in the SafeTensors format to store and load efficiently. The tokenizer along with its model configuration is saved for subsequent use.

2.3.2 Querying

This querying retrieves the model and metadata as needed for the analysis of the deployment. Most of the methods with traditional practices relied on purely manual effort, hence very time-consuming and error-prone. Some recent introductions include models that are automated querying systems. Examples include ModelDB, which develops a web interface to browse and analyze models; ModelHub, that presents a domain-specific language for querying deep learning models; and through automation in the process of querying, they make retrieval easy and efficient. These systems allow the ML models to reach the users and become usable with the offering of the mechanism toward the users for an easy way to interrogate the ML models.

After the storage step, both Model and the Tokenizer get loaded in the inferencing process to process the new data through texts and make appropriate predictions for the user queries.

2.3.3 Model Deployment

It refers to how the trained models are put in place within production environments. Traditional model deployments were laborious and prone to errors. The current models have also been deployed using automated systems for their deployments. For example, Clipper [1] would serve prediction services of low latency with high throughput and TensorFlow Serving provides an integration point for serving the deployments of TensorFlow models. These systems would thus reduce the burden in a natural way and enable simple automatic to ensure easy deployment. In this respect, models that could actually be deployed correctly would yield better reliability in applications to its performance.

Here, the model is deployed on the local system for answering the queries of news classification.

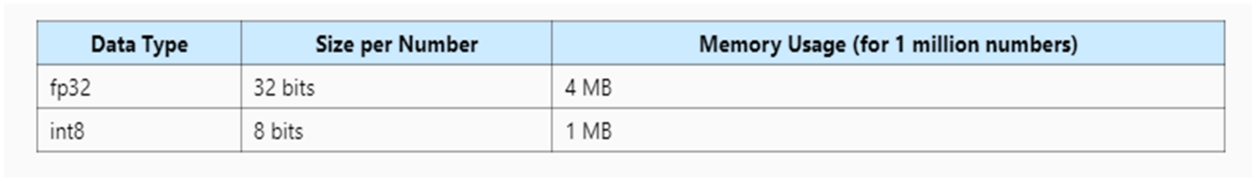
2.3.4 Debugging

Debugging of models can be defined as the identification and removal of bugs from the deployed models. Traditionally, it is a slow and error-prone task that is performed manually. Recently, an automated model debugging system has emerged. For example, MISTIQUE [1] logs and queries model intermediates for diagnosis, while DeepEverest accelerates top-k interpretation queries. These systems try to make the debugging process more efficient and automatic so that the deployed models are diagnosed for potential problems. This is how these systems improve the reliability and performance of ML models by virtue of their advanced debugging capabilities.

2.4 Model Compression Techniques: On-Device ML

2.4.1 Quantization

Quantization [2] is a reduction in the precision of model parameters to decrease size and increase speed. The traditional methods were time-consuming and prone to errors since they are manual. Quantization in recent times has focused on automation. TensorFlow Lite offers tools to quantize models, whereas PyTorch supports dynamic quantization. This has ensured that the size and complexity of the models reduce thus making them deployable on device. The technique commonly known as quantization reduces the precision of model parameters and makes it efficient and performance enhancing on resource-constrained devices.

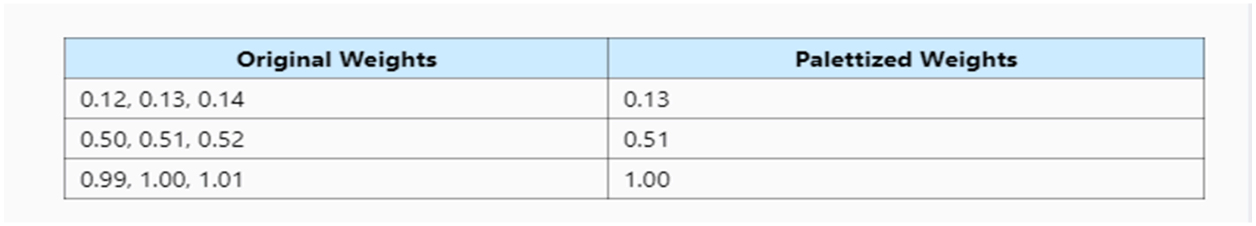


Data Type	Size per Number	Memory Usage (for 1 million numbers)
fp32	32 bits	4 MB
int8	8 bits	1 MB

Fig. 6 Example of quantization

2.4.2 Palettization

Palettization [2] is the technique in which weights are mapped to a discrete set of values; hence, reducing the size of the model. Traditionally, these techniques were dependent upon very time-consuming, error-prone manual efforts. Recently, automated techniques of palettization have been introduced. For example, palettization functionality is supported by Google's Edge TPU for on-device efficient inference. Palettization reduces model sizes so that they become deployable on-device. This is achieved through the mapping of weights to a discrete set of values that enhance efficiency and performance in ML models within scarce resource devices.



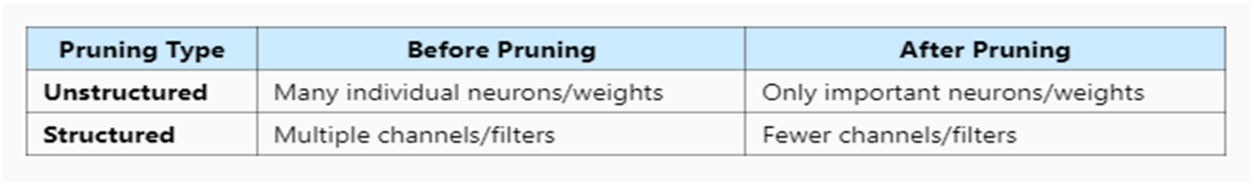
Original Weights	Palettized Weights
0.12, 0.13, 0.14	0.13
0.50, 0.51, 0.52	0.51
0.99, 1.00, 1.01	1.00

Fig. 7 Example of Palettization

2.4.3 Pruning

Pruning [2] is removing redundant neurons or weights for the simplification of the model. Most conventional methods depended on human efforts that were both time-consuming and prone to error. The newest advancement entails the use of automated pruning. Model pruning is implemented in the TensorFlow Model Optimization Toolkit, and PyTorch etc.

In both these techniques, it reduces the size of the models along with computation demands so that they can easily be deployed on mobile devices. Pruning techniques enhance not only the efficiency but also performance of ML models on resource-constrained devices. It removes less critical neurons or weights.



Pruning Type	Before Pruning	After Pruning
Unstructured	Many individual neurons/weights	Only important neurons/weights
Structured	Multiple channels/filters	Fewer channels/filters

Fig. 8 Example of Pruning

2.4.4 Distillation

Distillation [2] is the process through which a smaller model learns to mimic the behavior of the larger model. Before the discovery of techniques for distillation, old methods had only manual effort, which was quite tedious and error-prone. Some of the newer techniques have brought out automatic methods for distillation. This is an example in that DistilBERT [7] is the reduced version of BERT that carries almost all its performances. These techniques reduce the size and computational complexity of models so they can be run on devices. Training the performance of a larger model using a smaller model improves the efficiency and performance of ML models in resource-constrained devices.

Example:Voice Recognition System

Teacher Model: A large, complex model with high accuracy.

Student Model: A smaller, simpler model.

Process: The teacher model is used to generate predictions. The student model is trained to replicate the teacher model's predictions.

Result: The student model becomes nearly as accurate as the teacher model but is much smaller and faster, making it suitable for use on mobile devices.

2.5 Evaluating Compressed Models

Compression Goal: Reduce model size while preserving accuracy.

Compression Trade-off: Compression can degrade accuracy, sometimes improves generalization.

Key strategies of evaluation:

2.5.1 Trade-off Curve Visualization

Plot Metrics: Compare accuracy vs. performance metrics (e.g., size, latency).

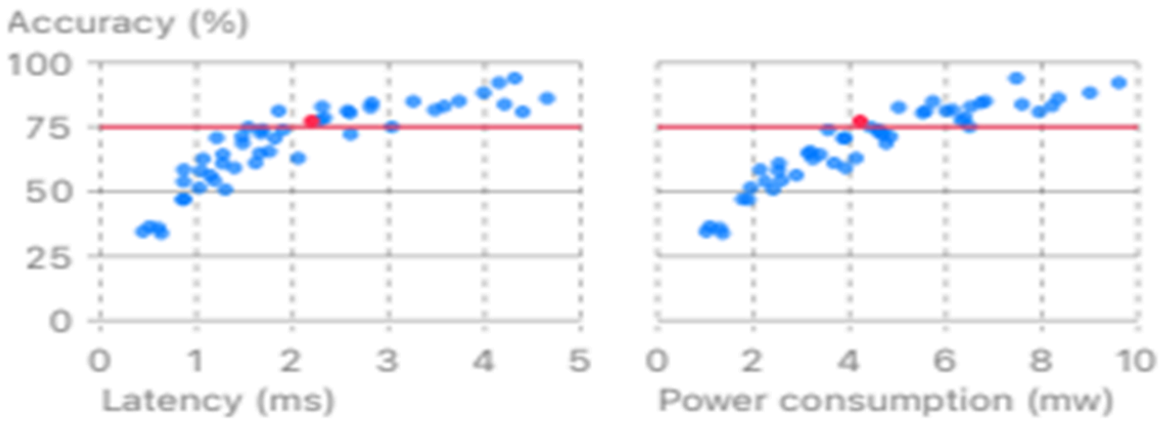


Fig. 9 Accuracy vs. performance metrics

2.5.2 Estimate Savings Early

Quantize a model with random weights to estimate savings as it will benefits before expensive training.

2.5.3. Robust Evaluation Pipeline

Structured approach to evaluate ML models, particularly after techniques like compression (quantization, pruning, etc.) are applied.

2.5.4. Demo User Experiences

Load models on-device and build demo applications. To get the user experience in the hands of people as fast as possible.

2.6 Technology Stack used for Implementation

1. Programming Language:

Python 3.9: Python is used due to its rich ecosystem of machine learning libraries and frameworks.

2. Key Libraries and Tools:

Transformers Library: Hugging Face's transformers for leveraging pretrained BERT models and tokenization utilities.

PyTorch: The primary deep learning framework for model fine-tuning, providing flexibility in customization.

mpi4py: For parallel data processing using MPI, enabling efficient distributed computation across multiple processes.

spaCy: Used for text preprocessing, including tokenization, lemmatization, and stopword removal.

re: Python's regular expressions library for text cleaning, such as removing URLs, punctuation, and extra whitespace.

Scikit-learn: Used for dataset splitting, performance metrics, and label encoding.

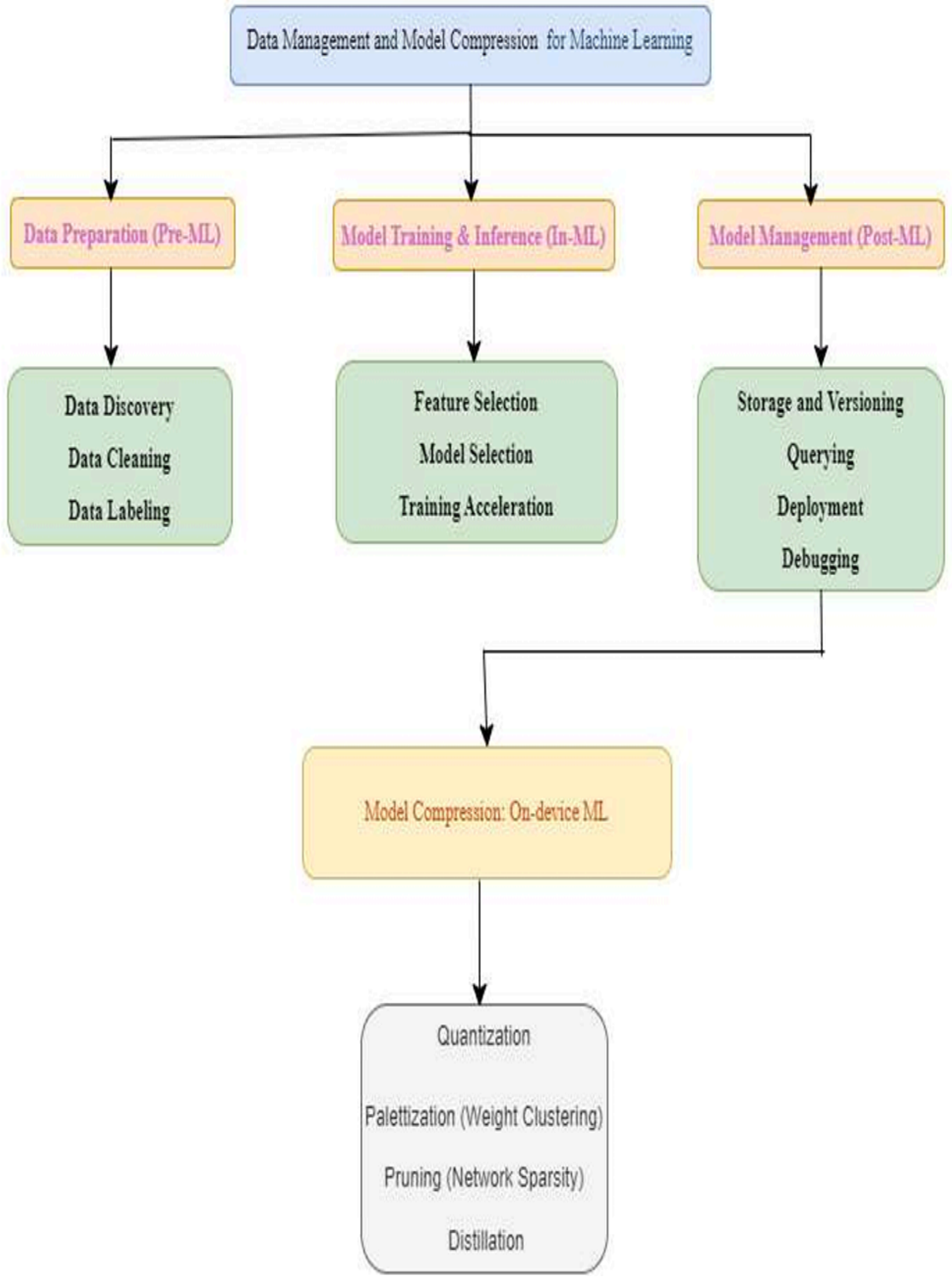
Pandas and NumPy: For efficient data handling and preprocessing.

CSV Library: For reading and writing datasets in .csv format.

time: Used for measuring and tracking the time taken for training, preprocessing, and data handling processes.

Matplotlib and Seaborn: For visualizing training metrics and evaluation results.

3. Hardware: The training and fine-tuning process were accelerated using Kaggle GPU via CUDA. This significantly reduced the time for processing large datasets.



Data Management and Model Compression for Machine Learning
Data Preparation (Pre-ML)
Model Training & Inference (In-ML)
Model Management (Post-ML)
Data Discovery
Data Cleaning
Data Labeling
Feature Selection
Model Selection
Training Acceleration
Storage and Versioning
Querying
Deployment
Debugging
Model Compression: On-device ML
Quantization
Palettization (Weight Clustering)
Pruning (Network Sparsity)
Distillation

Fig. 10 Brief overview of pre, in and post ML

CHAPTER 3: ARCHITECTURAL WORKING OF BERT MODEL

BERT is a transformer-based deep learning model developed by researchers at Google AI [9]. It excels at capturing context in text by processing input bidirectionally. This model was chosen for its state-of-the-art performance on a variety of NLP tasks, including text classification [10].

3.1 BERT Architecture:

3.1.1 Input Representation: Each input text is tokenized and converted into embeddings using the following components:

Token Embeddings: Word tokens represented as fixed-size vectors.

Segment Embeddings: Differentiates between segments in multi-sentence inputs.

Position Embeddings: Encodes the position of tokens in the sequence**.** A combination of these embeddings is passed into the transformer layers.

3.1.2 Input Sequence Format

The input to BERT is formatted as a sequence of tokens. The typical input format for BERT includes:

[CLS]: A special classification token added at the beginning of the sequence. It is used for classification tasks like sentiment analysis.

[SEP]: A separator token used to differentiate between sentence pairs. For single-sentence tasks, the [SEP] token is placed at the end of the sequence.

Tokens: The actual words or subword units that make up the sentence(s).

Example of tokenized input for two sentences: [CLS] I love BERT [SEP] It is transformer [SEP]

3.1.3 Transformer layers: Stacks of self-attention and feed-forward layers to capture relationships between words. Usually, BERT-Base has 12 transformer layers, and BERT-Large has 24 layers. These layers are a backbone of BERT for capturing complex relationships between words.

3.1.4 Output Representation: At the end of the transformer stack, BERT outputs a contextualized embedding for each token in the sequence. These embeddings capture word meaning in the context of the entire sentence, not like traditional word embeddings, such as Word2Vec or GloVe, which give static representations of the particular words.

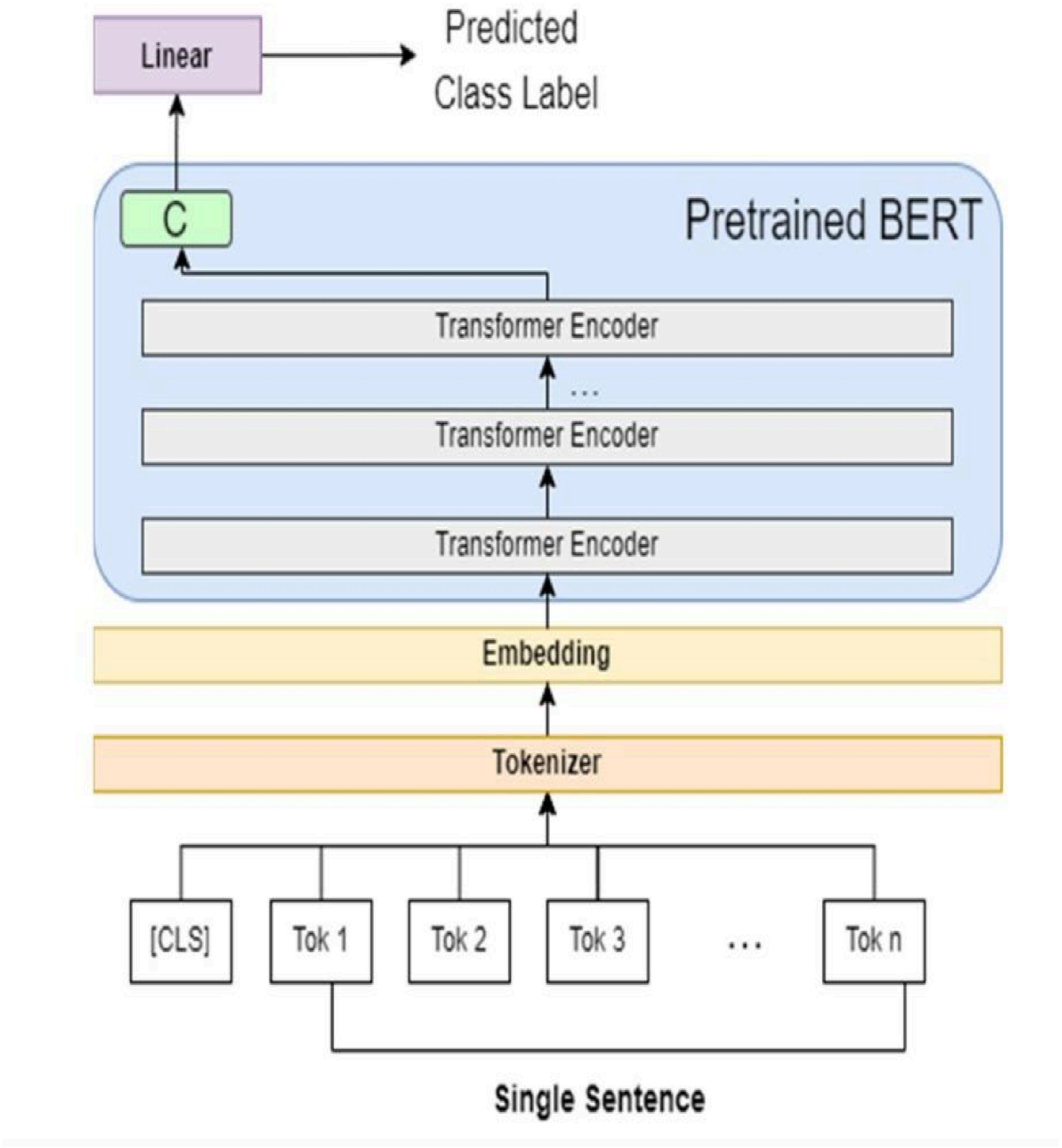


Fig.11 BERT Architecture

3.2 Parameters of the BERT for fine tuning: Here are the key parameters used during the fine-tuning of BERT for news classification:

1. Learning Rate (lr)

Value: 2e-5, A lower learning rate was chosen to fine-tune the pretrained BERT model without significantly disrupting its weights, ensuring stable updates during training.

2. Batch Size

Value: 16, A batch size of 16 strikes a balance between memory usage and training stability. Larger batches would require more memory, while smaller batches might slow down convergence.

3. Epochs

Value: 3, Since BERT is already pretrained, only 3 epochs were needed for fine-tuning. This is sufficient to adapt the model to the specific task without overfitting.

4. Warm-up Steps

Value: 500, Warm-up steps help stabilize the early stages of training by gradually increasing the learning rate, avoiding large updates that might destabilize the model.

5. Weight Decay

Value: 0.01, A small weight decay value helps regularize the model and prevent overfitting by penalizing large weights.

6. Gradient Accumulation Steps

Value: 1, Gradient accumulation is set to 1, meaning the gradients are updated after each mini-batch, suitable for the batch size used.

7. Max Sequence Length

Value: 128, A sequence length of 128 tokens ensures that the model processes enough context while keeping memory and computation efficient for news articles.

8. Dropout Rate

Value: 0.1, A dropout rate of 0.1 helps prevent overfitting by randomly disabling 10% of neurons during training, improving generalization.

CHAPTER 4: EXPERIMENTAL RESULTS AND ANALYSIS

Data Set: The dataset [15] is of size 823.354 MB in Excel format and contains 3 columns text, target label and word count. There are a total 26 classes in the target label. The dataset contains 697,528 rows and it is divided into training and testing categories. 558,022 rows are used in training and the remaining 139,506 rows are used for testing the model.

Classes in Dataset: There are total of 26 classes in the dataset which includes**-**

1. academic interests
2. arts and culture
3. automotives
4. books and literature
5. business and finance
6. careers
7. family and relationships
8. food and drinks
9. health
10. healthy living
11. hobbies and interests
12. home and garden
13. movies
14. music and audio
15. news and politics
16. personal finance
17. pets
18. pharmaceuticals, conditions, and symptoms
19. real estate
20. shopping
21. sports
22. style and fashion
23. technology and computing
24. television
25. travel
26. video gaming

Results:

Samples Per Category	Precision	Accuracy	Recall	F1-Score
10	0.7781	0.7712	0.7711	0.7715
50	0.8282	0.8261	0.8262	0.8243
100	0.8180	0.8161	0.8162	0.8160

Table 2. Performance Results

Formulas used to evaluate performance results:

1. Precision

Precision measures the ratio of correctly predicted positive observations to the total predicted positive observations. It reflects how precise the model's predictions are.

Precision = (TP + FP) / TP

2. Recall (also called Sensitivity or True Positive Rate)

Recall measures the ratio of correctly predicted positive observations to the total actual positives. It indicates the model's ability to capture positive instances.

Recall = TP/ (TP + FN)

3. F1-Score

F1-Score is the harmonic mean of Precision and Recall, providing a balance between the two. It is especially useful when class imbalance exists.

F1-Score = 2× (Precision × Recall) / (Precision + Recall)

4. Accuracy

Accuracy is the ratio of correctly predicted samples (both positives and negatives) to the total number of samples. It is a general measure of the model's correctness.

Accuracy = (TP+TN) / (TP+TN+FP+FN)

Where:

TP (True Positive): Correctly predicted positive samples.

TN (True Negative): Correctly predicted negative samples.

FP (False Positive): Incorrectly predicted positive samples.

FN (False Negative): Incorrectly predicted negative samples.

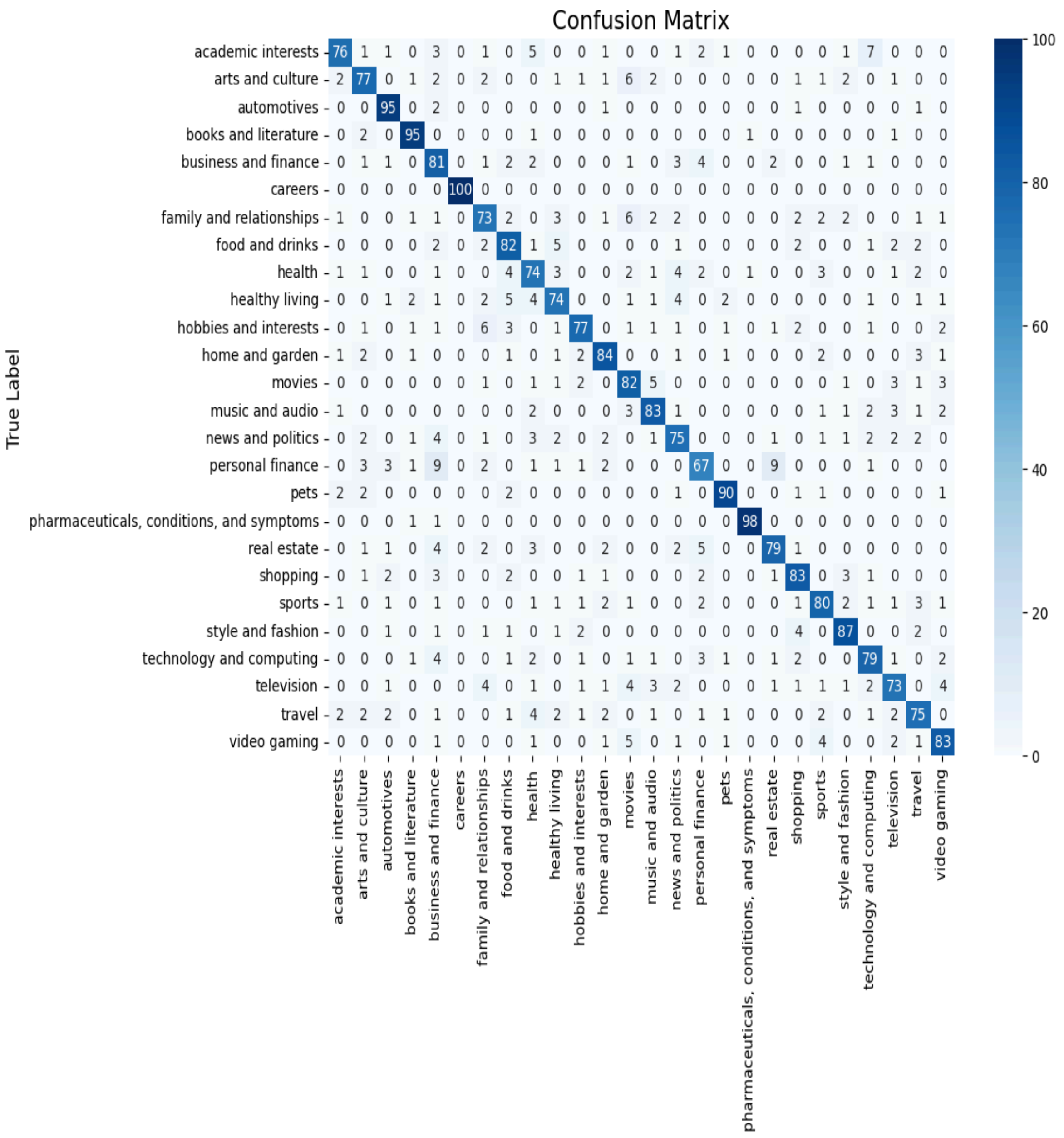


Fig 13. Confusion Matrix

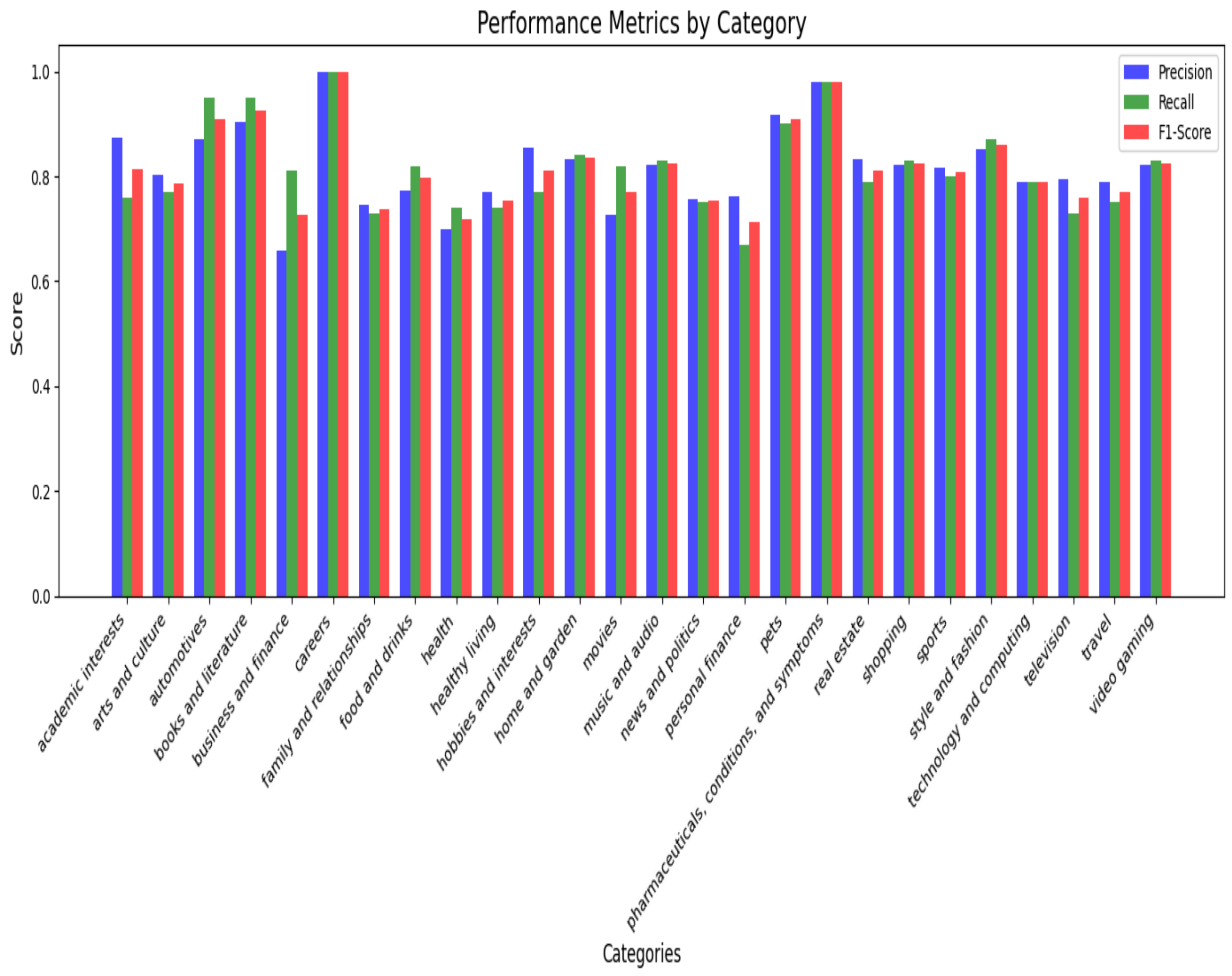


Fig 15. Performance Chart by Category

CHAPTER 5: CONCLUSION

Bidirectional Encoder Representations from Transformers (BERT) model is used in this report to classify news articles into 26 predefined categories. In this, we fine-tuned a pretrained BERT model on a curated news dataset to leverage its deep language understanding for achieving high accuracy in the task of text classification.

Key takeaways are:

Data Preprocessing: A good quality training dataset requires the process of data discovery, cleaning, and labeling.

Model Training and Inference: Feature selection, model selection, and speeding up the training of a model make the entire training process efficient and effective.

Model Selection: Since BERT can extract contextual relations in text, it is mostly suited for text classification when fine-tuned to a specific dataset.

Model Management: Stored, versioned, queried, deployed, and debugged BERT appropriately.

Parameter Tuning: It really had to do with how well the training went that the learning rate, the batch size, and also the number of epochs performed very well. This way, there was no chance for the model to overfit once it was fine-tuned at the given small learning rate, alongside an ideal batch size.

Performance: The BERT model actually scores very high with very precise, recall, and F1-score that indicates it will pretty much perform well even on class-imbalanced news article classifications.

Challenges and Improvements: While the model can perform in most cases, it is not to say that no challenges will be encountered at all. Indeed, imbalance of the dataset and handling longer sequences are big issues. Techniques with the inclusion of more data augmentation or even higher architectures for the models such as RoBERTA [6] or DistilBERT [10] can be envisioned as potential future work. As it stands today, BERT is used only for the purpose of transfer learning; in other words, when properly fine-tuned, it becomes a robust model for text classification. This term paper report, therefore, emphasizes the capability of performance in transformer-based models within tasks like natural language processing and news classification.

REFERENCES

[1] C. Chai, J. Wang, Y. Luo, Z. Niu, and G. Li, “Data Management for Machine Learning: A Survey,” IEEE Transactions on Knowledge and Data Engineering. Institute of Electrical and Electronics Engineers (IEEE), pp. 1–1, 2022. doi: 10.1109/tkde.2022.3148237.

[2] F. Hohman, M. B. Kery, D. Ren, and D. Moritz, “Model Compression in Practice: Lessons Learned from Practitioners Creating On-device Machine Learning Experiences,” Proceedings of the CHI Conference on Human Factors in Computing Systems, vol. 22. ACM, pp. 1–18, May 11, 2024. doi: 10.1145/3613904.3642109.

[3] A. Kumar, M. Boehm, and J. Yang, “Data Management in Machine Learning,” Proceedings of the 2017 ACM International Conference on Management of Data. ACM, May 09, 2017. doi: 10.1145/3035918.3054775.

[4] https://link.springer.com/book/9780387310732 [Christopher M. Bishop Pattern Recognition and Machine Learning]

[5] C. Sun, X. Qiu, Y. Xu, and X. Huang, “How to fine-tune bert for text classification?,” Lecture Notes in Computer Science, pp. 194–206, 2019. doi:10.1007/978-3-030-32381-3_16

[6] K. Nemkul, “Use of bidirectional encoder representations from Transformers (Bert) and robustly optimized Bert pretraining approach (Roberta) for Nepali News Classification,” Tribhuvan University Journal, vol. 39, no. 1, pp. 124–137, Jun. 2024. doi:10.3126/tuj.v39i1.66679

[7] Y. Arslan et al., “A comparison of pre-trained language models for multi-class text classification in the financial domain,” Companion Proceedings of the Web Conference 2021, pp. 260–268, Apr. 2021. doi:10.1145/3442442.3451375

[8] H. M. Zahera, I. A. Elgendy, R. Jalota, and M. Ahmed Sherif, Fine-tuned BERT model for multi-label tweets classification, 2019. doi:10.6028/nist.sp.1250.incident-dice_upb

[9] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805, 2018. [Online]. Available: https://arxiv.org/abs/1810.04805.

[10] T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 38–45. [Online]. Available: https://huggingface.co/transformers.

[11] S. Islam et al., “A comprehensive survey on applications of Transformers for deep learning tasks,” Expert Systems with Applications, vol. 241, p. 122666, May 2024. doi:10.1016/j.eswa.2023.122666

[12] E. Azeraf, E. Monfrini, and W. Pieczynski, “Improving usual naive Bayes classifier performances with neural naive Bayes based models,” Proceedings of the 11th International Conference on Pattern Recognition Applications and Methods, pp. 315–322, 2022. doi:10.5220/0010890400003122

[13] C. Silva and B. Ribeiro, “Enhancing svms for text classification,” Studies in Computational Intelligence, pp. 51–70, 2010. doi:10.1007/978-3-642-04533-2_3

[14] “Automatic Text Classification Model based on machine learning,” Machine Learning Theory and Practice, vol. 4, no. 1, Mar. 2023. doi:10.38007/ml.2023.040106

[15] “Data Set Link”: https://uc.hackerearth.com/he-public-data/dataset1c2f4b7.zip