A star with a flower and text

Description automatically generated

**Tribhuvan University**

**Institute of Science and Technology**

**Central Department of Computer Science and Information Technology**

A Dissertation Report

On

**Exploring Transformer Based Models for Nepali News Classification: A Comparison of BERT, RoBERTa, and DistilBERT**

In the partial fulfillment of the requirements for the Master’s Degree in Computer Science and Information Technology

Under the Supervision of

**Asst. Professor Bikash Balami**

Submitted to:

**Central Department of Computer Science and Information Technology Kirtipur, Kathmandu, Nepal**

Submitted by:

**Subash Khatiwada (561/076)**

April 2025

A star with a flower and text

Description automatically generated

**Tribhuvan University**

**Institute of Science and Technology**

**Central Department of Computer Science and Information Technology**

**Student’s Declaration**

I hereby declare that the contents written in this report are original and no sources other than mentioned here have been used in this work.

…………………….....

**Subash Khatiwada**

Date : 11/04/2025

A star with a flower and text

Description automatically generated

**Tribhuvan University**

**Institute of Science and Technology**

**Central Department of Computer Science and Information Technology**

**Supervisor’s Recommendation**

I hereby recommend that the thesis titled “**Exploring Transformer Based Models for Nepali News Classification: A Comparison of BERT, RoBERTa, and DistilBERT**” prepared under my supervision by **Mr. Subash Khatiwada** in partial fulfillment for the degree of M.Sc. in Computer Science and Information Technology be processed for evaluation.

……………………………

**Asst. Prof. Bikash Balami**

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kritipur, Nepal

Date : 11/04/2025

A star with a flower and text

Description automatically generated

**Tribhuvan University**

**Institute of Science and Technology**

**Central Department of Computer Science and Information Technology**

**Letter of Approval**

This is to certify that we have read this thesis and, in our opinion, it is applicable for the scope and the quality of the thesis in partial fulfillment of the requirement for the degree of Master of Science in Computer Science and Information Technology.

|  |  |  |
| --- | --- | --- |
| Evaluation Committee | | |
| ………………………………………..  **Asst. Prof. Bikash Balami**  **Supervisor**  CDCSIT, Tribhuvan University |  | ………………………………….  **Asst. Prof. Sarbin Sayami**  **Head of Department**  CDCSIT, Tribhuvan University |
| ………….……………………………  **Internal Examiner** |  | ……………………………………  **External examiner** |

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **Asst. Prof. Bikash Balami**, Central Department of Computer Science and Information Technology for his constant mentorship, guidance and criticism throughout the journey to complete this thesis titled "Exploring Transformer Based Models for Nepali News Classification: A Comparison of BERT, RoBERTa, and DistilBERT ".

I am grateful to department head **Asst. Prof. Sarbin Sayami** for his essential guidance, support and assistance.

Also, I would like to appreciate the support from **Asst. Prof. Arjun Singh Saud, Asst. Prof. Nawaraj Paudel and Asst. Prof. Balkrishna Subedi** of Central Department of Computer Science and Information Technology who are involved directly or indirectly with this thesis work from the earliest proposal defense to the final submission.

I would also like to thank the Department of Computer Science and Information Technology for providing me this opportunity.

Finally, I thank all the faculty members and staffs for their incessant support, collaboration, assistance and feedback.Also, I would like to thank my all friends and family for continuous support.

* Subash Khatiwada

April 2025

ABSTRACT

This study investigates the effectiveness of transformer-based models - BERT, RoBERTa, and DistilBERT, for classifying Nepali news articles across 12 categories. A dataset of 14,081 articles was collected from Nepali news portals, preprocessed, and split into training (70%), validation (15%), and testing (15%) sets. Each model was fine-tuned over 4 epochs and evaluated using weighted accuracy, precision, recall, and F1-score. In BERT and DistilBERT, the validation loss increased after the third epoch, causing a rollback to their three-epoch checkpoints, whereas the RoBERTa model demonstrated consistent performance improvement across all four epochs. The final performance was then evaluated on the unseen test set, where RoBERTa achieved the highest accuracy 86.84%, followed by BERT 84.9% and DistilBERT 82.05%. Major categories like National News and Sports exhibited strong performance, while minority classes, such as Blog and Employment, underperformed due to class imbalance and contextual overlap.

**Keywords**: NLP, News Classification, Transformers, BERT, RoBERTa, DistilBERT, Nepali Language

Table of Contents

[ACKNOWLEDGEMENT i](#_Toc195355288)

[ABSTRACT ii](#_Toc195355289)

[Table of Contents iii](#_Toc195355290)

[List of Tables vi](#_Toc195355291)

[List of Figures vii](#_Toc195355292)

[LIST OF ABBREVIATIONS viii](#_Toc195355293)

[Chapter 1. Introduction 1](#_Toc195355294)

[1.1 Overview 1](#_Toc195355295)

[1.2 Motivation 1](#_Toc195355296)

[1.3 Problem Statement 2](#_Toc195355297)

[1.4 Objective 2](#_Toc195355298)

[1.5 Scope of the Work 2](#_Toc195355299)

[Chapter 2. Literature Review 3](#_Toc195355300)

[2.1 Background Study 3](#_Toc195355301)

[2.1.1 Nepali Language 3](#_Toc195355302)

[2.1.2 Text Classification 3](#_Toc195355303)

[2.1.3 Transformer 4](#_Toc195355304)

[2.1.4 Large pretrained models 7](#_Toc195355305)

[2.2 Literature Review 10](#_Toc195355306)

[Chapter 3. Methodology 12](#_Toc195355307)

[3.1 Data Collection and Annotation 12](#_Toc195355308)

[3.2 Data Preprocessing 12](#_Toc195355309)

[3.3 Split the dataset 12](#_Toc195355310)

[3.4 Selection of pre-trained model 13](#_Toc195355311)

[3.5 Model Training 13](#_Toc195355312)

[3.6 Performance Metrics 14](#_Toc195355313)

[Chapter 4. Implementation 16](#_Toc195355314)

[4.1 Implementation Configuration 16](#_Toc195355315)

[4.1.1 Hardware 16](#_Toc195355316)

[4.1.2 Development Environment 16](#_Toc195355317)

[4.1.3 Python Libraries 16](#_Toc195355318)

[4.2 Data Collection 18](#_Toc195355319)

[4.2.1 Data Crawling 18](#_Toc195355320)

[4.2.2 Loading dataset in CSV file 19](#_Toc195355321)

[4.3 Data Preprocessing 19](#_Toc195355322)

[4.3.1 Label Encoding 19](#_Toc195355323)

[4.3.2 Data Cleaning 19](#_Toc195355324)

[4.3.3 Data Splitting 20](#_Toc195355325)

[4.4 Tokenization 21](#_Toc195355326)

[4.4.1 BERT Tokenization 22](#_Toc195355327)

[4.4.2 RoBERTa Tokenization 22](#_Toc195355328)

[4.4.3 DistilBERT Tokenization 23](#_Toc195355329)

[4.5 Model Building 24](#_Toc195355330)

[4.6 Inference 24](#_Toc195355331)

[Chapter 5. Result and Discussion 26](#_Toc195355332)

[5.1 Training Loss and Validation Loss 26](#_Toc195355333)

[5.2 Model Performance on Testing Dataset 28](#_Toc195355334)

[5.2.1 Performance of BERT 29](#_Toc195355335)

[5.2.2 Performance of RoBERTa 31](#_Toc195355336)

[5.2.3 Performance of DistilBERT 33](#_Toc195355337)

[5.3 Comparison of BERT, RoBERTa and DistilBERT models 35](#_Toc195355338)

[Chapter 6. Conclusion and Future Recommendation 37](#_Toc195355339)

[6.1 Conclusion 37](#_Toc195355340)

[6.2 Limitations and Future Recommendation 38](#_Toc195355341)

[References 39](#_Toc195355342)

[Appendix 42](#_Toc195355343)

List of Tables

[Table 2.1: Nepali Alphabets and Symbols 3](#_Toc195353930)

[Table 3.1: 12 x 12 Confusion Matrix 14](#_Toc195353931)

[Table 4.1: Parameters settings for BERT, RoBERTa and DistilBERT 24](#_Toc195353932)

[Table 5.1: Training Loss and Validation Loss for BERT model 26](#_Toc195353933)

[Table 5.2: Training Loss and Validation Loss for RoBERTa model 27](#_Toc195353934)

[Table 5.3: Training Loss and Validation Loss for DistilBERT model 27](#_Toc195353935)

[Table 5.4: Per-Class Precision, Recall, F1-score and Accuracy for BERT Model 30](#_Toc195353936)

[Table 5.5: Per-Class Precision, Recall, F1-score and Accuracy for RoBERTa Model 31](#_Toc195353937)

[Table 5.6: Per-Class Precision, Recall, F1-score and Accuracy for BERT Model 34](#_Toc195353938)

[Table 5.7: Comparative performance result of BERT, RoBERTa and DistilBERT 35](#_Toc195353939)

List of Figures

[Figure 2.1: The Encoder-decoder in the transformer architecture [9] 5](#_Toc195353940)

[Figure 2.2: Pre-training and Fine-tuning procedure for BERT [10] 8](#_Toc195353941)

[Figure 2.3 DistilBERT architecture, Source: [13] 9](#_Toc195353942)

[Figure 4.1: Total news articles 18](#_Toc195353943)

[Figure 4.2: Training dataset 20](#_Toc195353944)

[Figure 4.3: Validation dataset 21](#_Toc195353945)

[Figure 4.4: Testing dataset 21](#_Toc195353946)

[Figure 5.1: BERT, RoBERTa and DistilBERT training performance over epochs 28](#_Toc195353947)

[Figure 5.2: 12 x12 News Categories Confusion Matrix for BERT 29](#_Toc195353948)

[Figure 5.3: Summarized result for BERT model 30](#_Toc195353949)

[Figure 5.4: 12 x12 News Categories Confusion Matrix for RoBERTa 31](#_Toc195353950)

[Figure 5.5: Summarized result for RoBERTa model 32](#_Toc195353951)

[Figure 5.6: 12 x12 News Categories Confusion Matrix for DistilBERT 33](#_Toc195353952)

[Figure 5.7: Summarized result for DistilBERT model 35](#_Toc195353953)

[Figure 5.8: BERT, RoBERTa and DistilBERT performance comparison 36](#_Toc195353954)

LIST OF ABBREVIATIONS



# Introduction

## Overview

In recent years, NLP has become very popular and in great demand due to various applications such as text classification, speech recognition, information retrieval, sentiment analysis, name-entity recognition, machine translation, summarization etc. Text classification is one of the fields of NLP. There are various applications of text classification such as information filtering (email and short text message spam filtering), news categorization, document classification and so on. Most text classification and document categorization systems are used to separate the data into multi-class categories.

Numerous studies have explored news classification using traditional ML algorithms, including NB, LR, SVM, Hidden Markov Model, Decision Tree and Random Forest, and Ensemble Learning etc. These approaches have demonstrated promising results in classifying news. However, deep learning methods like transformer models have gained attention for their ability to utilize complex multi-layer neural networks to identify patterns and relationships within the data. [1]

This study presents the use of publicly accessible large pre-trained models like BERT, RoBERTa, and DistilBERT for classifying Nepali news articles. We perform a comparative analysis to assess the performance of these models by fine-tuning them on our dataset of Nepali news articles, which have categorized labels.

## Motivation

News plays a crucial role in keeping people informed about events happening around them. With the growth of internet use, the trend of consuming online news has grown significantly. Every day, thousands of news articles are published across various online platforms. Reading news articles without knowing their category can be challenging for some people. To address the problem created by unclassified news articles, there is a need for an automated tool to classify news and categorize these articles into proper labels such as “Business”, “Political”, “Sports”, “Technology” news and so on. [2] Finding an effective and robust categorization method allows online readers to read news easily and online news publishers to automatically classify their news.

The rapid advancement of large pre-trained models has allowed machines to vastly improve their understanding of human language. Despite the availability of various multilingual pre-trained models that support Nepali, there has been little research on their application to classify Nepali news articles. By fine-tuning these pre-trained models on labeled Nepali news data, we can accelerate development and improve the performance of Nepali news classification. This approach is not only more efficient than traditional machine learning but also offers a more effective alternative to building models from scratch.[3]

## Problem Statement

Online news consumption has increased dramatically over the past few years. Traditional ML classification techniques, including SVM, Naïve Bayes, and MLP Networks, LSTM and Bi-LSTM have been used in the past for classifying Nepali news [2], [4]. However, deep learning has recently become more popular for addressing machine learning problems compared to these traditional methods [1], [5]. This research aims to solve the Nepali news classification by developing a strong Nepali news classifier system by fine tuning on large pre-trained models. Transformer based models are state-of-art techniques that use self-attention to process input sequences. We use Nepali news datasets and advanced models like BERT, RoBERTa and DistilBERT to improve classification accuracy and efficiency.

## Objective

The key objectives of this study are:

* To construct Nepali news classifiers by fine-tuning the multilingual models of BERT, RoBERTa and DistilBERT
* To perform a comparative analysis of the fine-tuned models

## Scope of the Work

This study aims to contribute to the field of NLP, Nepali text classification, and can be used to develop various automatic systems for Nepali news media monitoring, information retrieval. The scope of this study covers following areas:

* Nepali news data collection and preprocessing
* Selection and implementation of models
* Performance evaluation of each model
* Future work and recommendation

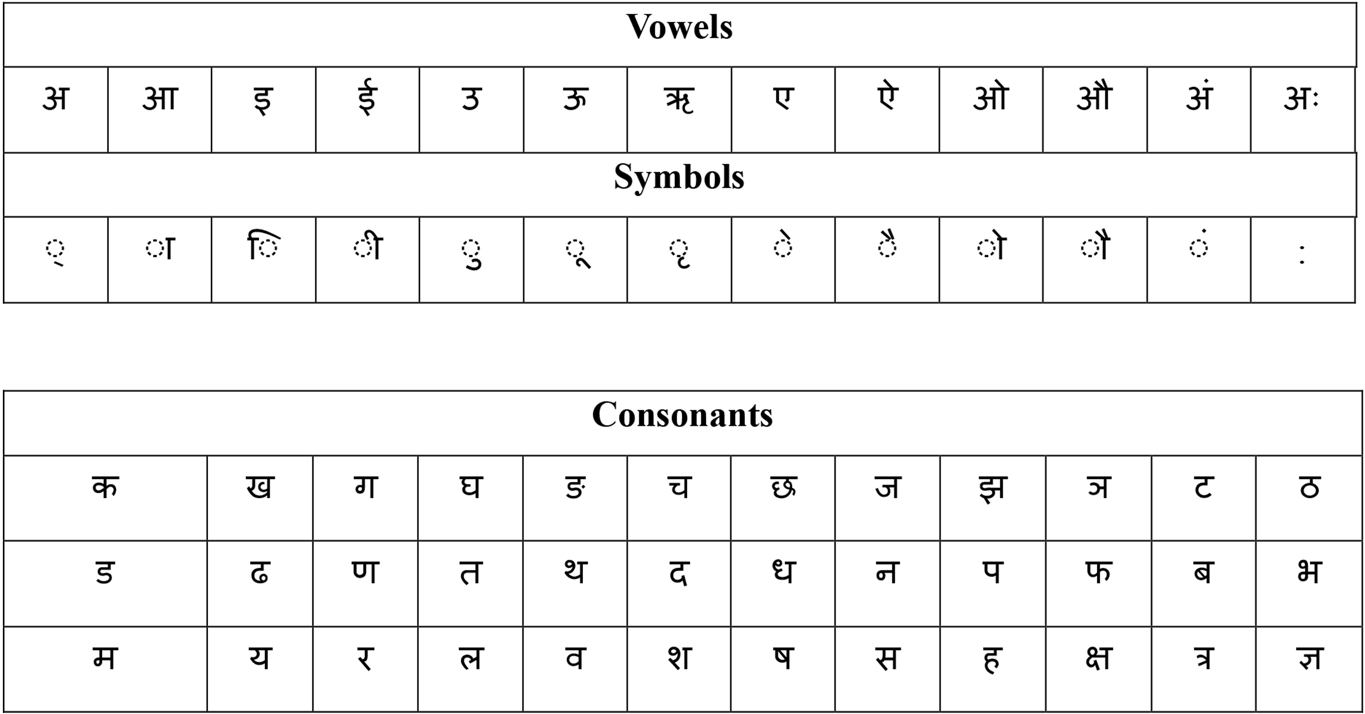
# Literature Review

## Background Study

### Nepali Language

Nepali language is spoken by more than 17.6 million people in the world, primarily in Nepal and India, with significant communities in countries such as Bhutan, Myanmar, and Brunei. There are no capital letters in the Devanagari script, and it is written horizontally left to right. The Nepali language consists of 13 vowels and 36 consonants. [2]

Table 2.1: Nepali Alphabets and Symbols



### Text Classification

NLP has gained a lot of popularity in the past few years due to its many uses, such as text classification, informational retrieval, speech recognition, sentiment analysis and more. Text classification, or text categorization, involves assigning labels to text data and is a fundamental NLP task. Text classification can be employed in various applications like news categorization, sentiment analysis, topic labeling, question answering, document classification, name entity recognition and POS (Part-of-speech) tagging, spam email filtering etc. [6]. Much research has been done in the field of text classification. Based on the approaches used in text classification, it can be separated into two approaches: Rule-based and Data-driven.

In rule-based methods, texts are categorized into organized categories by using a set of predefined rules and domain specific knowledge. Data driven methods are also known as machine learning methods. Data-driven methods use pre-labeled training data to build a model, and models then learn to classify the given text by observing patterns and relationships. The accuracy and effectiveness of the model largely depends on the quality and quantity of provided training data. Many researchers have grouped machine learning techniques used in text classification into two approaches: shallow classification approaches and deep learning approaches. [1], [5], [6], [7]

#### Shallow Learning Approaches

Shallow learning approaches are the traditional machine learning algorithms which rely on relatively simple mathematical models for learning and inference [7]. The conventional shallow classification techniques used in text classifications are: Probabilistic classification, K-NN based classification, SVM, Decision Tress and Random Forests, Logistic Regression, Ensemble Learning (Gradient boost algorithm, AdaBoost, XGBoost).

#### Deep Learning Approaches

Deep learning approaches utilize artificial neuron networks with multiple layers to learn complex patterns in data [1]. The popular deep learning approaches used in text classification are: Feed Forward Neural Networks, Recurrent Neural Network based models, Convolutional Neural Network, Graph Neural Networks [8], Transformer and pre-trained language models.

### Transformer

The Transformer model is an encoder-decoder based deep learning architecture first proposed by Vaswani in their paper “Attention is all you need” [9]. This model is used for transforming the input sequence to output sequence using the mechanism called self-attention. The transformer was created to address the weaknesses of earlier models such as RNNs and LSTMs at capturing long-range dependencies in sequences. The transformer model fundamentally relies on a self-attention mechanism to process and generate representations of its inputs and outputs, effectively bypassing the need for recurrent layers or convolutional operations, thereby enhancing computational efficiency and scalability. RNNs need to process data sequentially, whereas attention mechanisms can be parallelized, making the training process faster.

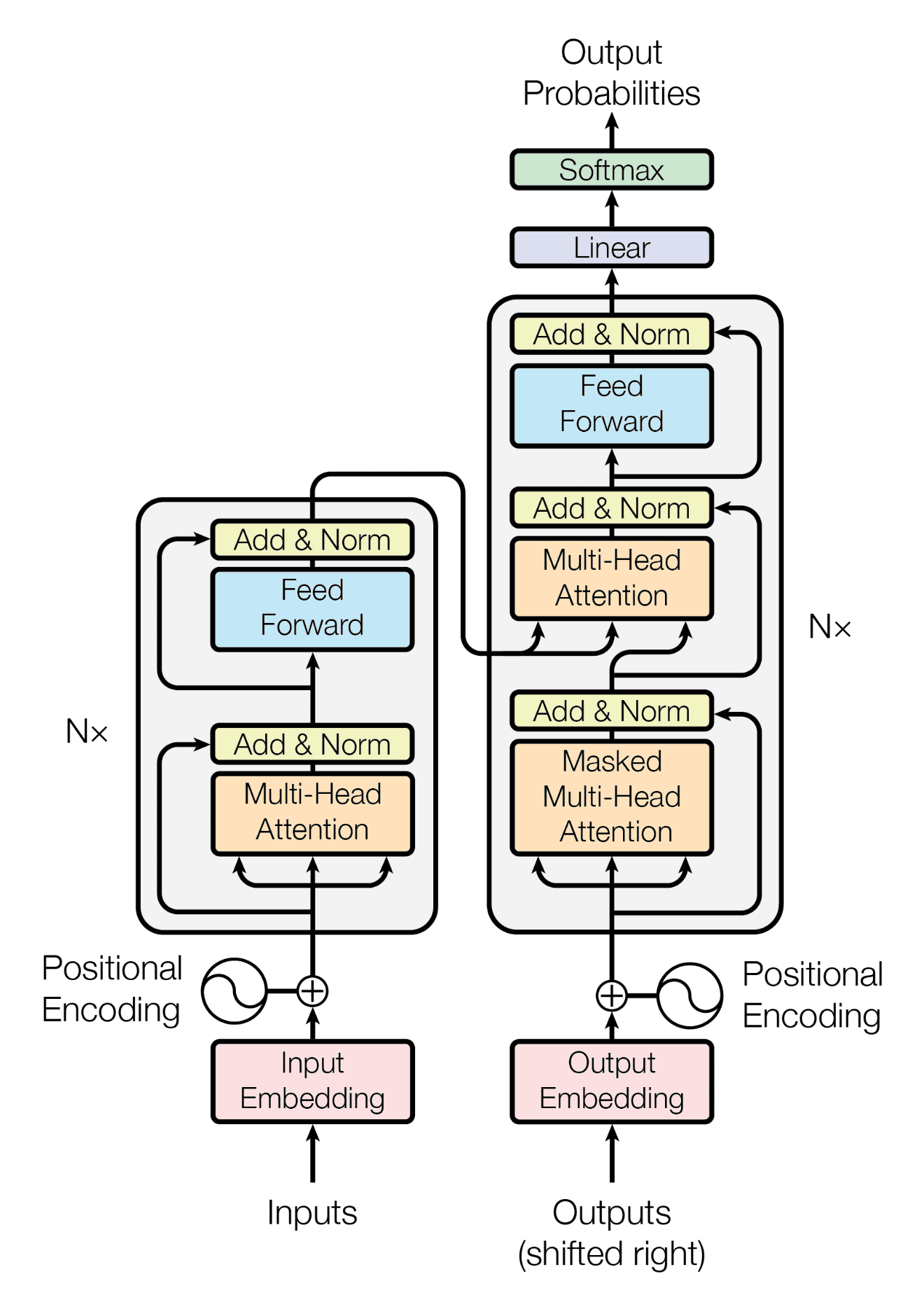


Figure 2.1: The Encoder-decoder in the transformer architecture [9]

The Transformer model has an encoder-decoder architecture. Both the encoder and decoder are composed of multiple stacks. The encoder stack is made up of 6 identical layers, each of which includes two key sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. Both sub-layers are accompanied by a normalization layer. On the other hand, the decoder stack features a multi-head attention layer, a masked multi-head self-attention mechanism, and a feed-forward neural network, with each layer also followed by a normalization layer.

The encoder is responsible to take the input embeddings (with positional encoding added), process them through multiple layers, and produce a high-level representation of the input. This output is then passed to the decoder. The decoder uses the encoder's output and the target sequence shifted right (to predict the next token). But during inference, it generates tokens one by one, using its previous outputs. The decoder must handle generating the sequence step by step, ensuring that each step only depends on previously generated tokens.

Input sequence first is converted into token and each input token is represented into a fixed sized vector representationby input embedding. Positional encoding uses sine and cosine functions to ensure similar positions have similar encodings and added to the input embedding.

A black background with a black square

AI-generated content may be incorrect.



Where, is the position and is the dimension and, and is the dimension vector.

Multi-head attention layer is based on attention mechanism, consists of multiple scaled dot-product attention functions. It runs multiple attention heads parallel to capture various aspects of input. The multi-head self-attention mechanism runs multiple attention operations in parallel.



Where,

Q: the matrix of queries,

K: the matrix of keys and

V: the matrix of values

,

A black background with a black square

AI-generated content may be incorrect.

The scaled dot-product attention function consists of queries and keys of dimension , and values of dimension . It uses SoftMax function to obtain weight.

h: number of heads

, ,

The output from Encoder stack can be defined by:



Where, Position-wise Feed Forward Network is used to find out the complex pattern in data by applying two linear transformations with a ReLU activation in between:



In each residual connection, there is a normalization layer which can be defied as:

The encoder generates continuous vector representations, which are then passed to the decoder in the transformer architecture. The decoder is composed of six identical layers, each containing three primary sub-components: a masked self-attention mechanism, an encoder-decoder multi-head attention (also known as cross-attention), and a feed-forward neural network.

### Large pretrained models

Large pre-trained models are neural networks trained on extensive datasets to perform a broad spectrum of tasks, and they can be further fine-tuned to adapt to domain-specific applications. These models generally consist of billions of parameters and rely on the principles of transfer learning [3].

#### BERT

BERT is a transformer-based model created by Google AI Language in 2018. It works in two main steps: pre-training and fine-tuning. In pre-training, the model learns from a large amount of unlabeled text data to understand general language patterns. Then, during fine-tuning, it is adjusted using smaller, labeled datasets to perform specific tasks like answering questions or classifying text [10]. BERT's ability to understand context from both directions (left and right) makes it highly effective for various natural language processing tasks.

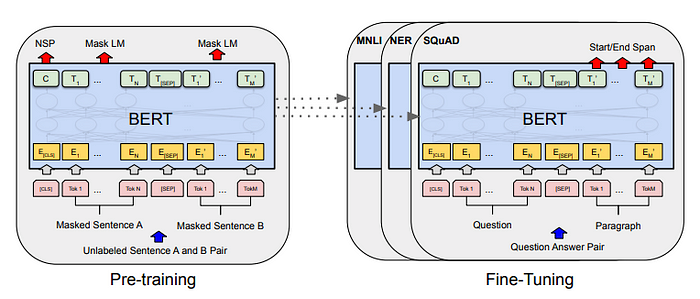


Figure 2.2: Pre-training and Fine-tuning procedure for BERT [10]

To create the pre-trained model, BERT uses two unsupervised learning tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In MLM, certain words in the input text are randomly hidden, forcing the model to predict them by analyzing both the preceding and following context. This helps BERT learn bidirectional language representations. On the other hand, NSP trains the model to determine whether one sentence logically follows another, improving its understanding of sentence relationships. By combining the power of Transformers with MLM and NSP, and training on a massive dataset, BERT develops a deep and comprehensive understanding of language structure and context.

Researchers have released the BERT's architecture variants like BERTBASE and BERTLARGE, differing in layers and parameters in their paper. BERTBASE model consists of 12 transformer layers, 768 dimensions, 12 attention heads and 110M parameters. BERTLARGE model consists of 24 transformers layers, 1024 dimensions, 16-heads, 340M parameters which is pre-trained on dataset of BooksCorpus (800M words) and English Wikipedia (2500M words). There is another variant of BERT which is trained in over 104 languages, called multilingual BERT (mBERT).

#### RoBERTa

RoBERTa was introduced by researchers at Facebook AI in 2019[11]. It builds upon the original BERT model but introduces several key improvements to the pretraining process. While RoBERTa retains the self-attention mechanism used in BERT, it incorporates dynamic masking techniques during training. Additionally, RoBERTa is trained on larger datasets and for longer durations, which significantly enhances its performance compared to BERT. A multilingual variant of RoBERTa, known as Cross-lingual Language Model RoBERTa (XLM-R), has also been developed to support multiple languages, further expanding its applicability in natural language processing tasks.

#### DistilBERT

DistilBERT is a smaller, faster, and more efficient version of the BERT model, introduced by Hugging Face in 2019. It follows the core architecture of BERT but with fewer layers and hidden units, making it 40% smaller and 95% faster [12]. The model achieves its efficiency through knowledge distillation and a modified training process. DistilBERT has fewer layers and smaller hidden units compared to BERT, with only 6 layers and 2,048 hidden units, versus BERT's 12 layers and 7,680 hidden units. The knowledge distillation involves teaching DistilBERT to replicate the behavior of the larger BERT model. During training, BERT's outputs serve as soft targets for DistilBERT, guiding it to approximate BERT's performance. [13].

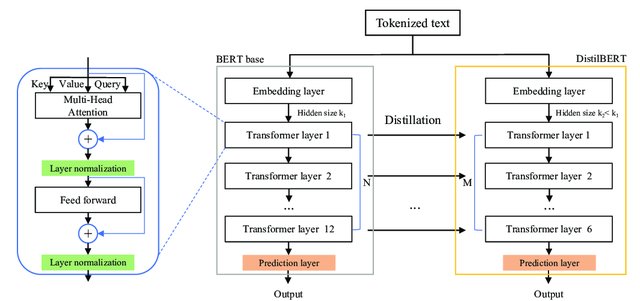


Figure 2.3 DistilBERT architecture, Source: [13]

## Literature Review

TB Shahi and AK Pant compares three machine learning approaches: Naive Bayes, SVM, and MLP for classifying Nepali news articles. Researchers collected and categorized 4,964 articles from online news portals into 20 distinct categories. Preprocessing involved sentence tokenization, removing special characters, numbers, and stop words. Features were extracted using TF-IDF. Five exclusive observations were performed for each algorithm with different train-test splits of 10%, 20%, 30%, 40%, and 50%. The experiment shows the SVM with a Radial Basis Function (RBF) achieved the highest accuracy of 74.65%, followed by Linear SVM with 74.62%, MLP with around 72.99%, and Naive Bayes with the lowest accuracy of 68.31%. [2]

Salehin analysis examines a dataset of 75,951 Bangla news samples across 12 categories. Researchers use bNLTK toolkit for data processing. They employ TF-IDF for feature extraction in various models, including Logistic Regression, Multinomial NB, SVM, Random Forest, XGBoost, and MLP, while a tokenizer is used for the LSTM model. Performance metrics, such as accuracy, precision, recall, and F1 score, revealed that the LSTM model achieved the highest accuracy at 0.87, compared to 0.78 for the SVM model. However, limitations include a restricted selection of feature extraction methods and a lack of quality Bangla stemming and lemmatization libraries, suggesting that future work could focus on expanding feature selection techniques and enhancing language processing tools. [14]

Wagle and Thapa present the performance of three different deep learning techniques: LSTM, Bi-LSTM, and Transformer Model for classifying Nepali news articles. The researchers scraped over 200,000 news articles from various news portals and classified them into 17 categories. The preprocessed dataset was then divided into training and testing sets with an 80-20 split. They used a 256-dimensional input embedding and an LSTM layer with 128 cells, connected to two fully connected layers and a final SoftMax layer for classification. For the Transformer model, they pre-trained it on all available training news articles without any labels using Masked Language Modeling. After fine-tuning, the Transformer model achieved a training F1-score of 96.54% and a testing score of 95.35%, outperforming all other models.

A comparative analysis of word embeddings techniques for Italian new categorization compares three different word embedding approaches for training eight different machine learning models. Word embeddings technique captures the semantic and syntactic similarity, and their relationship with other words. Researchers used three approaches (Word2Vec, GloVe and FastText) of word embedding for comparing eight models (Gaussian NB, Bernoulli NB, KNN, SVC, Decision Tree, Random Forest, AdaBoost, XGBoost). Support Vector Classification algorithm performed best performance. [15]

Deep learning is evolving rapidly, and there has been limited work on news classification, especially in low-resource languages like Nepali. Recently, some monolingual pre-trained models for the Nepali language have been developed. NepBERTa, a model based on BERT, has been trained on a large Nepali corpus and was introduced by G. Milan and T. Sulav. Researchers collected data from various news websites and used Nepali Wikipedia data to build this pre-trained model [[16]. In the paper by [17], compares the performance of several multilingual pre-trained models, such as mBERT and XLM-RoBERTa, with Nepali-specific monolingual pre-trained models such as, NepaliBERT, NepBERT, distilbert-base-nepali, deberta-base-nepali specifically for Nepali text classification as the downstream task.Recent studies have explored the use of BERT and RoBERTa for Nepali news classification, addressing a critical gap in NLP research for low-resource languages [18]. Notably, RoBERTa has demonstrated superior performance, achieving 95.3% accuracy in classification task. However, DistilBERT, a lightweight and efficient variant of BERT remains unexplored for Nepali news classification, despite its potential to balance accuracy and efficiency.

# Methodology

## Data Collection and Annotation

Deep learning approaches rely heavily on large datasets, the effectiveness of model depends upon the quality and quantity of data used. A pre-trained model has already been trained on vast amounts of data, but fine-tuning is necessary to adapt the model to classify the Nepali news into specific categories. The procedure for collecting news articles from online news are:

* Identify the popular online Nepali news portals
* Identify the HTML structure and parser of the published articles
* Use tools such as Beautiful Soup to scrap news articles
* Annotate the label to the collected news articles

## Data Preprocessing

The collected data has various unwanted characters, noises, and stop words. Filtering those noises helps to speed up the modeling process and simultaneously improves the results. Following steps are carried out for processing the Nepali news articles:

* Label encoding
* Removal of URLs
* English letters and numbers removal
* Special symbols removal
* Nepali stop words removal
* Removal of duplicate news articles

## Split the dataset

The preprocessed dataset undergoes a stratified split to create three subsets for model development and evaluation. The dataset is split into three subsets which are following:

1. Training set (70%): It serves as the primary dataset for model fitting and parameter optimization.
2. Validation set (15%): It facilitates hyperparameter tuning and monitors for overfitting during the training phase.
3. Test set (15%): It provides an unbiased evaluation of the final model’s performance.

## Selection of pre-trained model

Not all pre-trained models are designed to support the Nepali language, as many language models are primarily trained on high-resource languages like English, Chinese, or Spanish. However, there are multilingual variants that include Nepali in their training data, making them suitable for tasks involving the language. Some of the most notable models include:

1. mBERT (Multilingual BERT) – A multilingual version of BERT trained on the top 104 languages using Wikipedia text. Since Nepali is included in its dataset, mBERT can handle tasks like text classification, sentiment analysis, and named entity recognition in Nepali.
2. XLM-RoBERTa – A more powerful multilingual model trained on a large-scale dataset covering 100 languages, including Nepali. It offers better tokenization and training on diverse internet text rather than just Wikipedia.
3. DistilBERT (Multilingual variant) – A lighter and faster version of BERT designed for efficiency while maintaining good performance. Its multilingual variant can still be useful for Nepali-language applications with limited computational resources.

## Model Training

Model training is an essential process for adapting a pre-trained model to a specific task. It involves tokenization, model training, and hyperparameter optimization. Fine-tuning a pre-trained model like mBERT, XLM-RoBERTa or DistilBERT for Nepali news classification involves several steps:

* Tokenization: Convert news articles text data into tokens using a language-appropriate tokenizer.
* Model Fine-Tuning: Train the model on labeled news, adjusting weights to learn news classification patterns.
* Hyperparameter Optimization: Tune parameters like batch size, learning rate, and epochs for better accuracy.

## Performance Metrics

By evaluating prediction of the fine-tuned model with the test data, performance of the model can be measured in terms of accuracy, precision, recall and f1-score. A confusion matrix is an N x N table designed to evaluate the effectiveness of a classification model, with N denoting the number of target categories. The rows of the matrix represent the true or actual classes, while the columns reflect the predicted classes output by the model. This framework allows for a detailed analysis of the model's performance, including its correct predictions and errors. The confusion matrix corresponds to a 12-class classification task using fine-tuned BERT, where each row represents the true class and each column indicates the model’s predictions [19]. The class labels are mapped to indices as follows: Bank-0, Blog-1, Business-2, Economy-3, Employment-4, Entertainment-5, National News-6, Opinion-7, Sports-8, Technology-9, Tourism–10, and World-11

Table 3.1: 12 x 12 Confusion Matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Actual Classification |  | Class0 | Class1 | Class2 | … | Class11 |
| Class0 | M00 | M01 | M02 | … | M011 |
| Class1 | M10 | M11 | M12 | … | M111 |
| Class2 | M20 | M21 | M22 | … | M211 |
| …. | … | … | … | … | … |
| Class11 | M110 | M111 | M112 | … | M1111 |
| Prediction Classification | | | | | |

Where represents the numbers of instances belongs to actual class and predicted class The diagonal element of the matrix represents the number of instances where the predicated class matches the actual class. By the help of the confusion matrix, we can calculate the accuracy, precision, and f1-score.[20]

In a classification model evaluation, the following terms are used to describe the outcomes:

Accuracy of class is calculated by;

Average accuracy of the model is calculated by:

Precision and recall for each class are calculated using following formulas:

F1-Score for each class is calculated by the harmonic mean of precision and recall.

# Implementation

The study was conducted using a diverse setup involving hardware, software, and libraries for training and evaluating transformer-based models. The development environment was based on Python version 3.12.4, with Pip version 24.0 for managing libraries, and Jupyter Notebook version 7.2.2. For the software side, a range of Python libraries (pandas, NumPy, transformers, torch) was utilized.

## Implementation Configuration

### Hardware

The experiments were conducted on the following hardware:

* Processor: Apple M3 Pro 11-Core CPU and 14-Core MPS GPU
* Memory: 18 GB Unified RAM
* Storage: 512 GB

### Development Environment

The experiments were conducted on the following development environment:

* Python: Version 3.12.4, the primary programming language for implementation.
* Pip: Version 24.0, the package manager for installing and managing Python libraries.
* Jupyter Notebook: Version 7.2.2, an interactive environment for code development, visualization, and documentation, managed via Anaconda-Navigator.

### Python Libraries

This study leverages the following Python libraries, each serving a specific purpose in the research pipeline:

* beautifulsoup: Facilitates web scraping and parsing of HTML/XML documents, useful for collecting text-based datasets from online sources.
* scrapy is used for crawling the structured data from the news portals.
* datasets: A Hugging Face library that simplifies loading, processing, and managing datasets, optimized for natural language processing (NLP) tasks.
* matplotlib: A plotting library used to visualize results through graphs and charts, aiding in data interpretation.
* numpy: Provides efficient array operations and mathematical functions, foundational for numerical computations.
* os: A module for interacting with the operating system, such as managing files and directories.
* pandas is used for manipulation and analysis of structured data.
* re: Supports regular expressions for text pattern matching and preprocessing tasks.
* seaborn: Enhances matplotlib with advanced statistical visualizations, improving insight presentation.
* sklearn: A machine learning library offering tools for preprocessing, model training, and evaluation (e.g., accuracy metrics).
* torch: The PyTorch a deep learning framework that enables efficient construction and training of neural networks. It also supports parallel data processing across GPUs for accelerated computation.
* transformers: A Hugging Face library providing pre-trained NLP models and tokenization utilities for sequence classification.

## Data Collection

### Data Crawling

**Scrapy was used** to crawl news data from various news websites. The scraped data **are organized** in a folder, with subfolders for each category (e.g., business, national news, sports). Each article **are saved** as a text file in the following format:

onlinekhabar\_news/

├── business/

├── national news/

├── sports/

│ └── ...

└── ...

A total of 14,081 news articles were collected from prominent Nepali news portals including Ekantipur, Onlinekhabar, and Ratopati. The gathered news data are categorized into 12 distinct labels.

A graph of a number of people

AI-generated content may be incorrect.

Figure 4.1: Total news articles

### Loading dataset in CSV file

The collected data contains numerous impurities, necessitating a preprocessing step. Initially, all labeled news data is consolidated into a single .csv file. This data is then loaded into the pandas’s dataframe with three columns: file\_id, content and label.

## Data Preprocessing

The collected data has various unwanted characters, noises, and stop words. Filtering those noises helps to speed up the modeling process and simultaneously improves the results. Following steps are carried out for processing the Nepali news articles:

### Label Encoding

The categories of each news item are mapped to their corresponding labels, which are encoded numerically using the LabelEncoder from the sklearn library.

News category and its corresponding label are:

{"Bank": 0, "Blog": 1, "Business": 2, "Economy": 3, "Employment": 4, "Entertainment": 5, "National News": 6, "Opinion": 7, "Sports": 8, "Technology": 9, "Tourism": 10, "World": 11}

### Data Cleaning

The data cleaning process contains various steps which are described below:

#### Removal of Special Characters and Symbols

Python re library is used for removing English characters, symbols, numbers and emojis having following regular expression:

[a-zA-Z0-9!@#$%^&\*()\_+\-=\[\]{};\'‘\\:"|<,./<>?~`।१२३४५६७८९०–÷]

#### Removal of Nepali Stopwords

Stopwords are commonly occurring words in a text that add little value to analysis and are typically removed to enhance NLP model efficiency. In English, examples include “a,” “an,” and “the,” while Nepali stopwords consist of words like “अर्को,” “अरूलाई,” “बीच” and “आदि” . Eliminating these words during preprocessing helps refine datasets, optimize memory usage, and improve text classification accuracy.

Before preprocessing:

"१ नेपाल राष्ट्र बैंकले आजका लागि विदेशी मुद्राको $ विनिमयदर निर्धारण गरेको छ।"

After preprocessing:

"नेपाल राष्ट्र बैंकले आजका विदेशी मुद्राको विनिमयदर निर्धारण"

### Data Splitting

The preprocessed data is split into a 70-30 ratio for training and testing using *train\_test\_split* from *sklearn*. The test set is further divided into test and validation subsets. The training dataset (70%) is used for fine-tuning, the validation dataset (15%) helps tune hyperparameters and assess performance during training, and the test dataset (15%) is reserved for final evaluation. Since the test set remains unseen during training and validation, it provides an unbiased measure of the model’s performance.

A graph with numbers and a bar

AI-generated content may be incorrect.

Figure 4.2: Training dataset

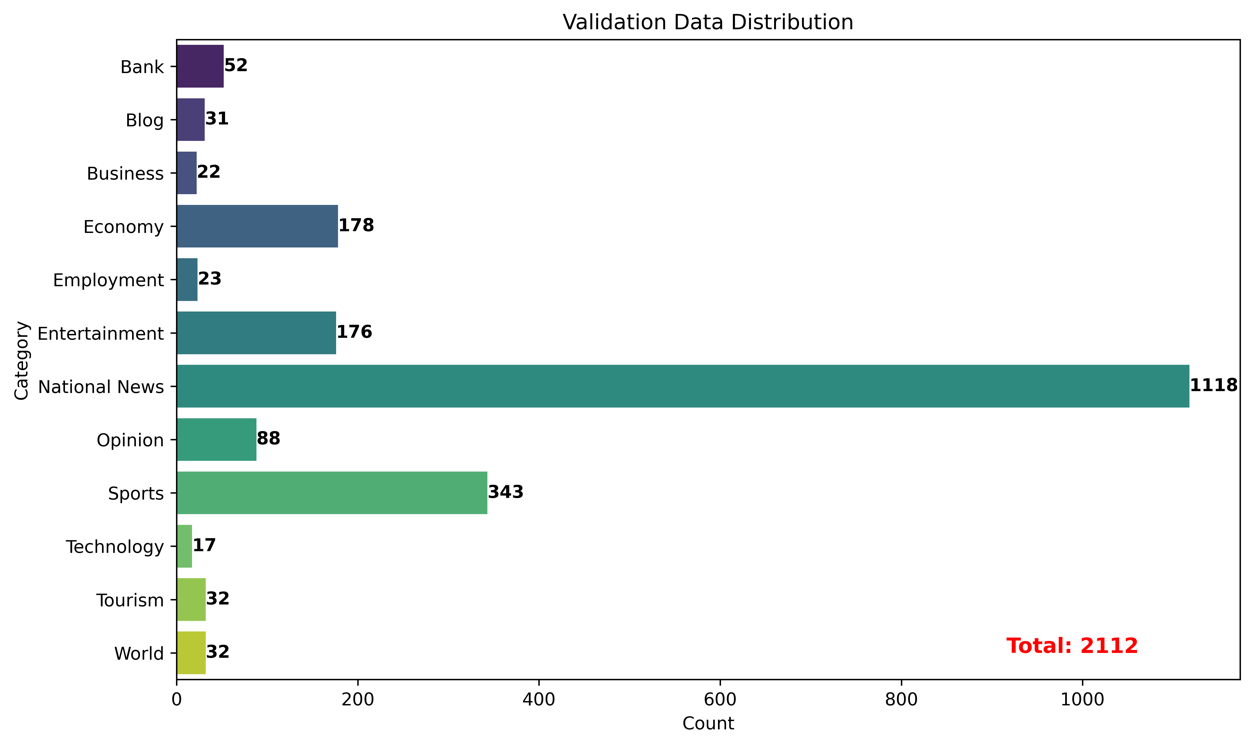


Figure 4.3: Validation dataset

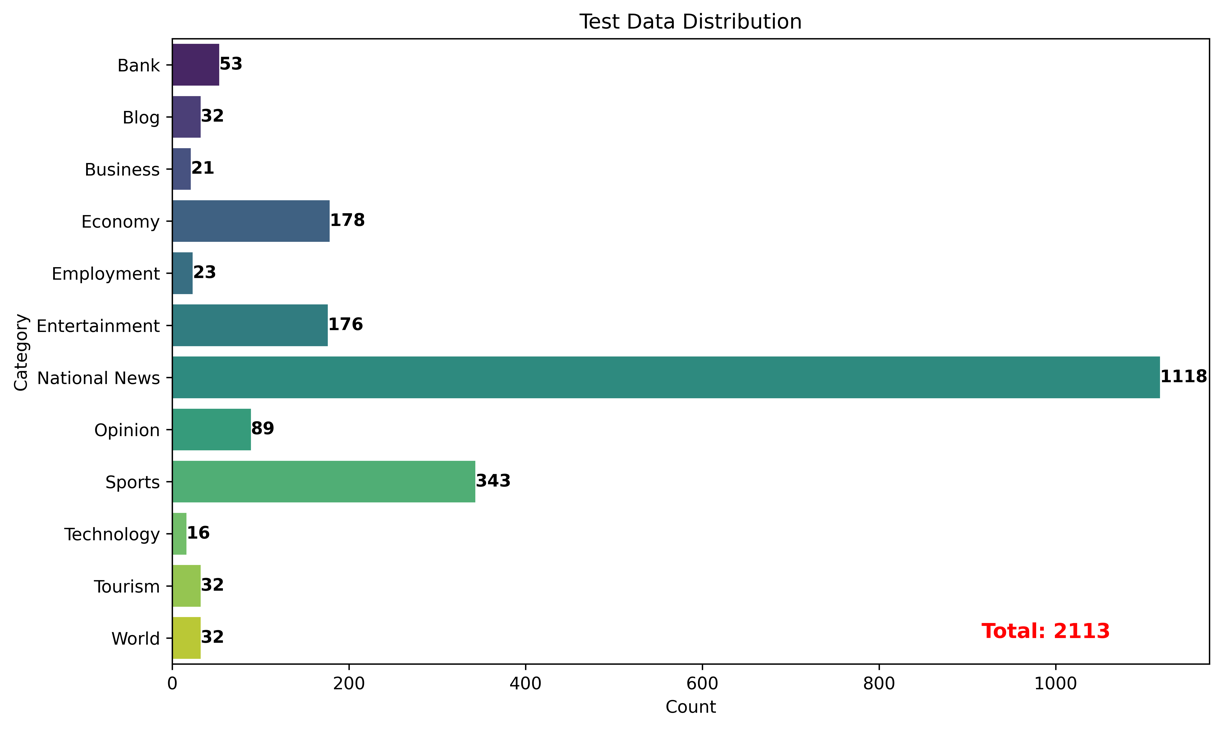


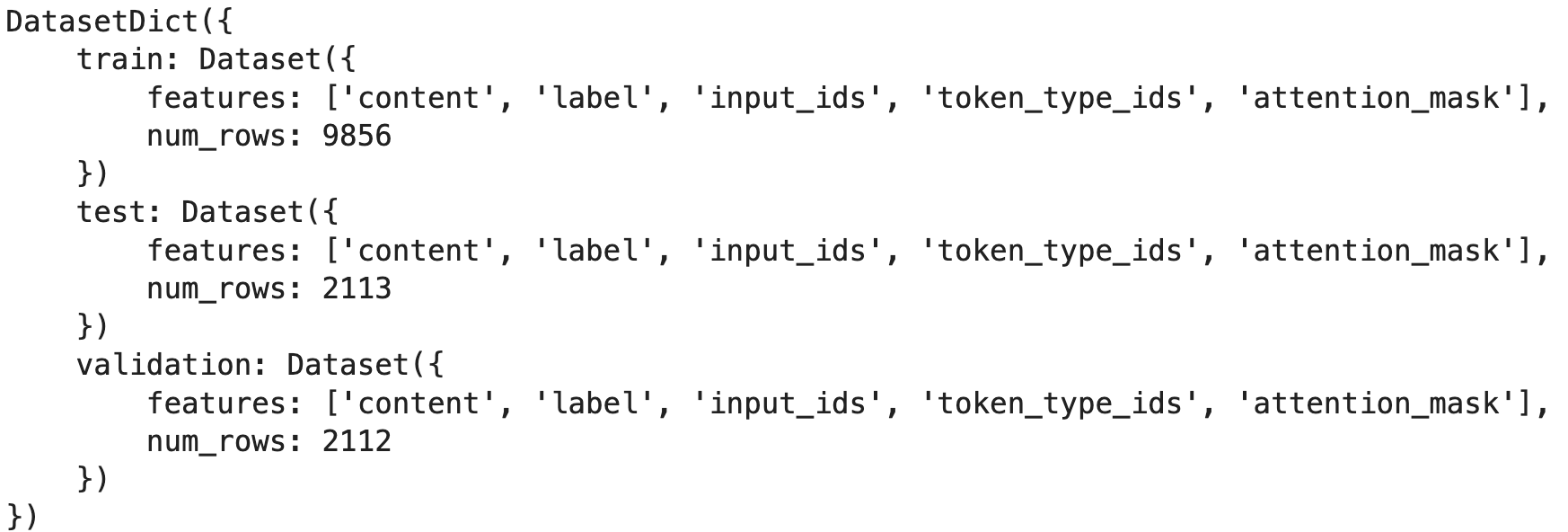
Figure 4.4: Testing dataset

## Tokenization

Tokenization is an essential preprocessing step in transformer-based models. The *transformers* library from Hugging Face is used to tokenize the training, validation, and test datasets. Since BERT, RoBERTa, and DistilBERT were trained using different tokenization strategies, each model requires its own specific tokenizer. The tokenization process converts raw text data into a numerical format suitable for input into transformer models. BERT, RoBERTa, and DistilBERT models use similar tokenization methods but have some differences in their preprocessing and input formats.

### BERT Tokenization

The bert-base-multilingual-uncased tokenizer from Hugging Face is a WordPiece-based tokenizer designed to handle text in 104 languages, including Nepali. It uses a vocabulary of 119,547 subword tokens to efficiently process multilingual inputs by breaking down rare words into smaller components, reducing unknown ([UNK]) tokens. The tokenizer is case-insensitive, converting all text to lowercase, which simplifies processing but may lose case-sensitive information. Key special tokens include [CLS] (classification token), [SEP] (separator), [PAD] (padding), and [UNK] (unknown words) [21]. It supports a maximum sequence length of 512 tokens, with longer texts truncated automatically.



### RoBERTa Tokenization

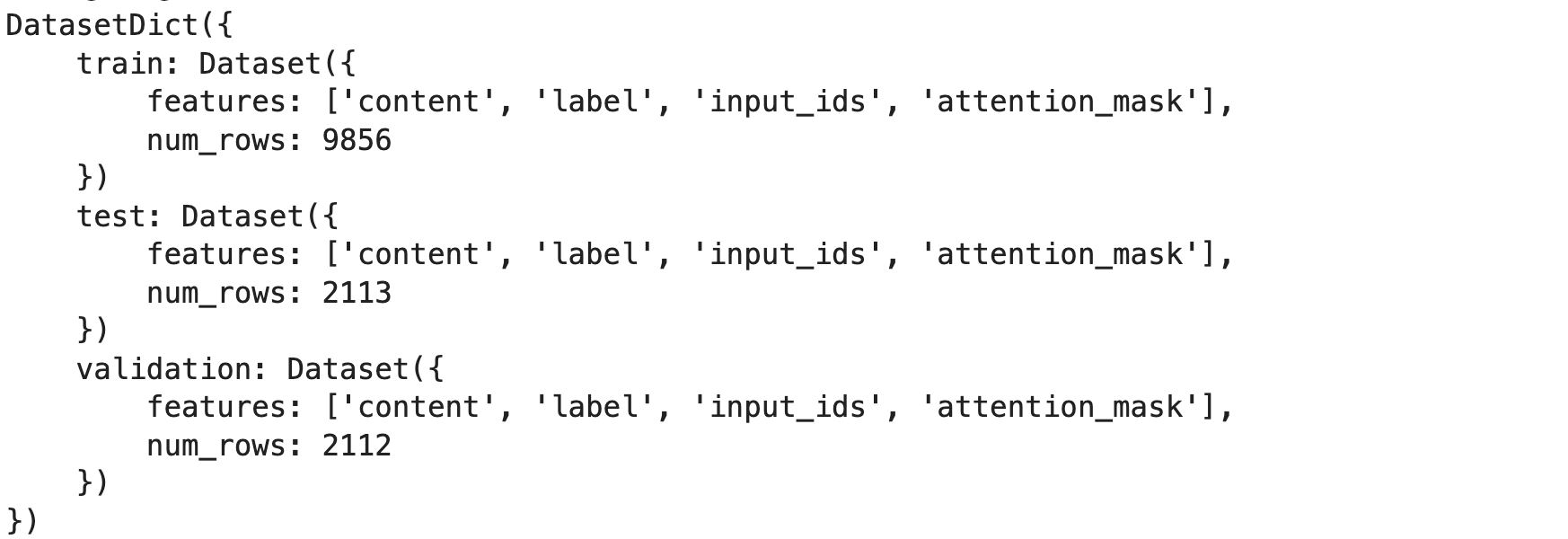
The xlm-roberta-base tokenizer from Hugging Face is a SentencePiece-based tokenizer designed to handle text in 100+ languages, including Nepali. It uses a larger vocabulary of 250,000 subword tokens to efficiently process multilingual inputs. It supports sequences up to 512 tokens and automatically handles longer texts through truncation. The tokenizer's design is particularly effective for low-resource languages and cross-lingual transfer learning tasks.

A close-up of a computer code

AI-generated content may be incorrect.

### DistilBERT Tokenization

The distilbert-base-multilingual-cased tokenizer from Hugging Face is a WordPiece-based tokenizer optimized for 104 languages, including Nepali. As a distilled version of BERT, it retains multilingual capabilities while being 40% faster and lighter.



## Model Building

We conduct fine-tuning of three transformer model: BERT, RoBERTa, and DistilBERT for 4 epochs. During each epoch, validation loss and training loss as well as accuracy, precision, recall and f1-score are monitored. The tokenized training dataset is utilized for fine-tune of the models, while the validation dataset is used for hyperparameter optimization. The following table summarizes the training parameters and computational characteristics observed during the experiments:

Table 4.1: Parameters settings for BERT, RoBERTa and DistilBERT

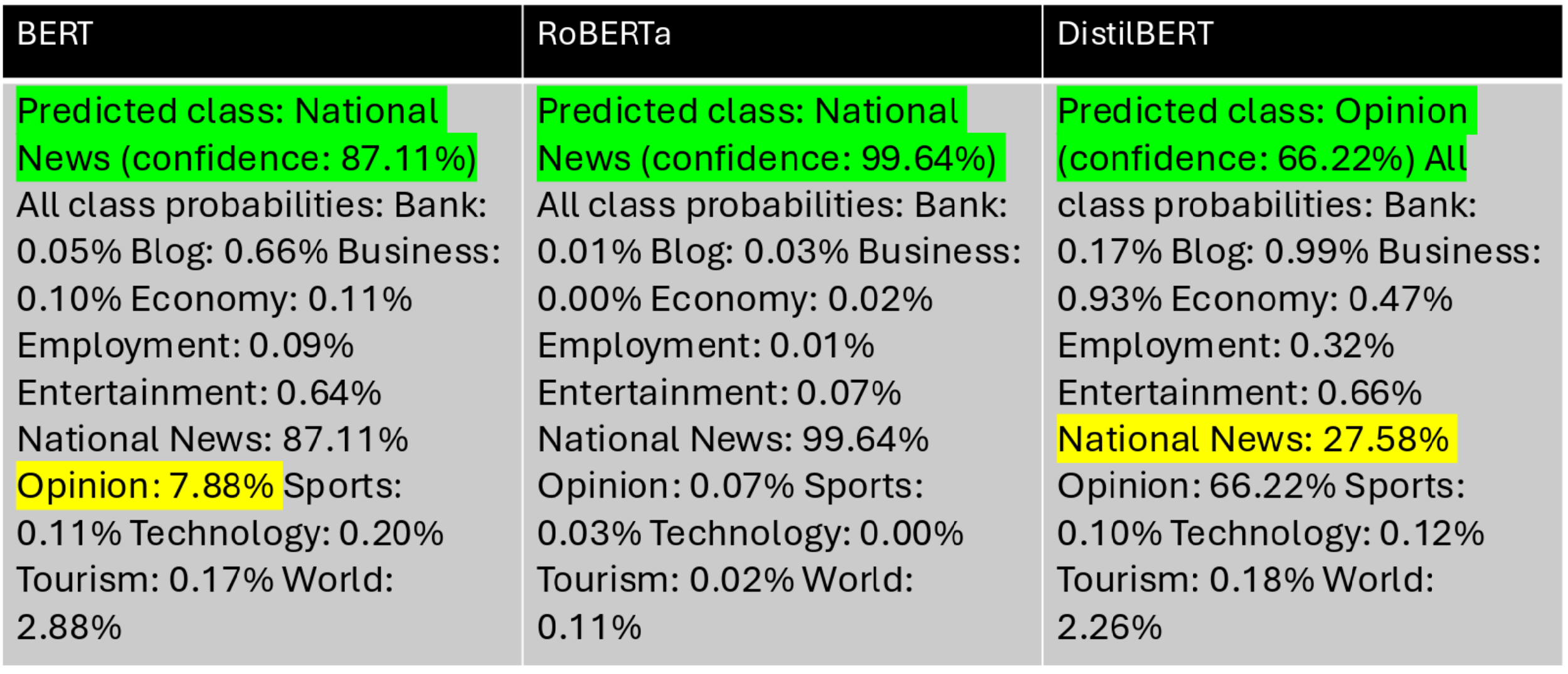
|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **BERT** | **RoBERTa** | **DistilBERT** |
| Batch Size | 10 | 10 | 10 |
| Number of Epoch | 4 | 4 | 4 |
| Max. Length | 512 | 512 | 512 |
| Learning Rate | 3e-5 | 3e-5 | 3e-5 |
| Weight Decay | 0.05 | 0.05 | 0.05 |
| Warmup Steps | 198 | 198 | 198 |
| Total Training Time | ~ 6h 25m | ~ 6h | ~ 1h 9m |
| Memory Consumption | ~ 15.05 GB | ~ 16.72 GB | ~ 8.06 GB |

## Inference

This below process demonstrates how to use a fine-tuned model to classify Nepali news text into one of 12 predefined categories.

Input text: "गत साता राजसंस्था पक्षधरहरूलाई ‘शान्तिपूर्ण’ विरोध प्रदर्शनका लागि स्थानीय प्रशासनले काठमाडौंको तीनकुने क्षेत्र प्रदान गरेको थियो । तर प्रज्वलनशील इन्धनको डिपो र अन्तर्राष्ट्रिय विमानस्थल नजिकै रहेको स्थानमा आन्दोलन अशान्त मात्र भएन, विध्वंसात्मक देखियो । आन्दोलनका क्रममा एक टेलिभिजन पत्रकारसहित दुई जनाको ज्यान गयो । विमानस्थलभित्र आगो फ्याँकियो । जडीबुटी उत्पादन तथा प्रशोधन कम्पनी लिमिटेडका कर्मचारीमाथि धरपकड गरियो र आगो लगाइयो । कान्तिपुर टीभी र अन्नपूर्ण पोस्टजस्ता सञ्चारगृहका कार्यालयमा आक्रमण गरियो । एकीकृत समाजवादीको पार्टी कार्यालयमा तोडफोड तथा आगजनी गरियो । निजी घर, गाडी तथा सार्वजनिक सम्पत्ति जलाइयो । भाटभटेनी सुपरमार्केटमा लुटपाट गरियो । आगो निभाउन गएको दमकलसमेत रोकियो, तोडफोड गरियो र फर्काइयो । प्रहरी तथ्यांकअनुसार प्रदर्शनकारीले १२ स्थानमा आगजनी गरेका र निजी तथा सरकारी गरी १८ भवन तोडफोड गरेका छन् । हिंसात्मक प्रदर्शनको जिम्मेवारी कसैले लिएको छैन । प्रदर्शनका कमान्डर दुर्गा प्रसाईं भागेका छन् । शान्तिपूर्ण प्रदर्शनलाई हिंसाका रूपमा भड्काउन आह्वान गर्ने र उक्साउने रवीन्द्र मिश्र र धवलशमशेर राणा तथा चोरी, तोडफोड, लुटपाट र आगजनीमा संलग्न केही व्यक्ति प्रहरी हिरासतमा छन् । संघीय तथा स्थानीय सरकार र गणतन्त्रवादी दलहरूले हिंसात्मक प्रदर्शनका लागि पूर्वराजा ज्ञानेन्द्र शाह मुख्य जिम्मेवार भएको दाबी गरेका छन् । यस घटनाको पृष्ठभूमि र प्रयोजन तथा राजनीति र सुरक्षा दृष्टिकोणलाई विश्लेषण गर्ने प्रयासस्वरूप यो लेख तयार गरिएको छ ।"

Output:



# Result and Discussion

Each model was trained on a dataset comprising 9,856 news articles distributed across 12 distinct categories. To ensure robust evaluation, the data was carefully partitioned into three subsets: a training set of 9,856 articles for model development, a validation set of 2,112 articles for hyperparameter tuning and model selection, and an independent test set of 2,113 articles reserved exclusively for final evaluation on completely unseen data. The evaluation methodology employed four key metrics - precision, recall, F1-score, and accuracy to comprehensively measure model performance across all categories.

## Training Loss and Validation Loss

In this study, we fine-tuned bert-base-multilingual-uncased, xlm-roberta-base and distilbert-base-multilingual-cased models separately on a training dataset and evaluated their performance on a validation dataset over four epochs. For each epoch, we calculated training loss and validation loss to monitor the model’s learning progress and generalization ability. Furthermore, we evaluated classification performance using the metrics of Accuracy, Precision, Recall, and F1-score. The results are presented in below:

Table 5.1: Training Loss and Validation Loss for BERT model

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Epoch | Training Loss | Validation Loss | Accuracy | Precision | Recall | F1 |
| 1 | 0.953900 | 0.625896 | 0.796875 | 0.772595 | 0.796875 | 0.771229 |
| 2 | 0.544600 | 0.523362 | 0.833333 | 0.838252 | 0.833333 | 0.831964 |
| **3** | **0.400700** | **0.516584** | **0.844697** | **0.846153** | **0.844697** | **0.842006** |
| 4 | 0.280800 | 0.562933 | 0.842330 | 0.843717 | 0.842330 | 0.839649 |

The BERT model’s training loss decreased steadily from 0.9539 in Epoch 1 to 0.2808 in Epoch 4, indicating successful optimization and learning on the training data. However, validation loss dropped from 0.625896 (epoch 1) to a minimum of 0.516584 by Epoch 3 but increased to 0.562933 in Epoch 4, reflecting overfitting as the model began to memorize training patterns rather than generalize. Epoch 3 is the optimal checkpoint; training beyond this point degraded validation performance. Classification metrics improved consistently, with Accuracy rising from 0.796875 to 0.842330 and F1-score from 0.771229 to 0.839649.

Table 5.2: Training Loss and Validation Loss for RoBERTa model

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Epoch | Training Loss | Validation Loss | Accuracy | Precision | Recall | F1 |
| 1 | 1.0166 | 0.64075 | 0.79072 | 0.74904 | 0.79072 | 0.763606 |
| 2 | 0.547 | 0.515009 | 0.824811 | 0.836337 | 0.824811 | 0.825657 |
| 3 | 0.4132 | 0.50274 | 0.839489 | 0.842012 | 0.839489 | 0.838204 |
| **4** | **0.3137** | **0.497674** | **0.851326** | **0.853343** | **0.851326** | **0.850509** |

The RoBERTa model demonstrates a consistent improvement in performance over the course of four epochs. In the first epoch, the model begins with a high training loss of 1.0166 and a validation loss of 0.64075, achieving an accuracy of 0.79072 and an F1-score of 0.7636. As training progresses, the model shows significant improvements. In the second epoch, the training loss drops to 0.547 and the validation loss reduces to 0.5150, with the accuracy improving to 0.8248 and the F1-score increasing to 0.8257. In the third epoch, the training loss further decreases to 0.4132 and the validation loss to 0.5027, leading to a higher accuracy of 0.8394 and an F1-score of 0.8382. By the fourth epoch, the training loss reaches its lowest value of 0.3137, and the validation loss continues to decline to 0.4977. At this stage, the model achieves its highest accuracy of 0.8513 and its best F1-score of 0.8505.

Table 5.3: Training Loss and Validation Loss for DistilBERT model

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Epoch | Training Loss | Validation Loss | Accuracy | Precision | Recall | F1 |
| 1 | 1.023400 | 0.700712 | 0.772254 | 0.729926 | 0.772254 | 0.740100 |
| 2 | 0.627200 | 0.562694 | 0.810133 | 0.808448 | 0.810133 | 0.804043 |
| **3** | **0.478200** | **0.544509** | **0.816761** | **0.822563** | **0.816761** | **0.815005** |
| 4 | 0.362900 | 0.557560 | 0.820549 | 0.816472 | 0.820549 | 0.816498 |

DistilBERT, a distilled version of BERT, reduced its training loss from 1.0234 in Epoch 1 to 0.3629 in Epoch 4, indicating efficient learning despite its smaller size. Validation loss decreased from 0. 700712 to a low of 0. 544509 in Epoch 3 but rose slightly to 0. 55756 in Epoch 4, suggesting mild overfitting similar to BERT. Classification metrics improved steadily, with Accuracy rising from 0. 772254 to 0.820549 and F1-score from 0. 7401 to 0.816498. Epoch 3 marked the best balance of generalization and performance, with the subsequent validation loss increase indicating that further training may not enhance unseen data performance.

A graph with lines and numbers

AI-generated content may be incorrect.

Figure 5.1: BERT, RoBERTa and DistilBERT training performance over epochs

RoBERTa demonstrates the best overall performance, with the lowest validation loss and highest accuracy by Epoch 4. BERT and DistilBERT, while effective in reducing training loss and improving accuracy, exhibit mild overfitting in the final epoch, suggesting that stopping at Epoch 3 might be optimal for these models. DistilBERT, as a lightweight model, offers a good balance of efficiency and performance but falls slightly short of BERT and RoBERTa in accuracy.

## Model Performance on Testing Dataset

Performance of each model is evaluated on the testing dataset. Since the testing dataset remains unseen during training and validation, it offers an unbiased assessment of the model's performance.

### Performance of BERT

In BERT model, the optimal model checkpoint was Epoch 3, as further training led to declining validation performance. To evaluate the model's effectiveness, we selected this checkpoint and assessed its performance using the test dataset to generate the final evaluation metrics.

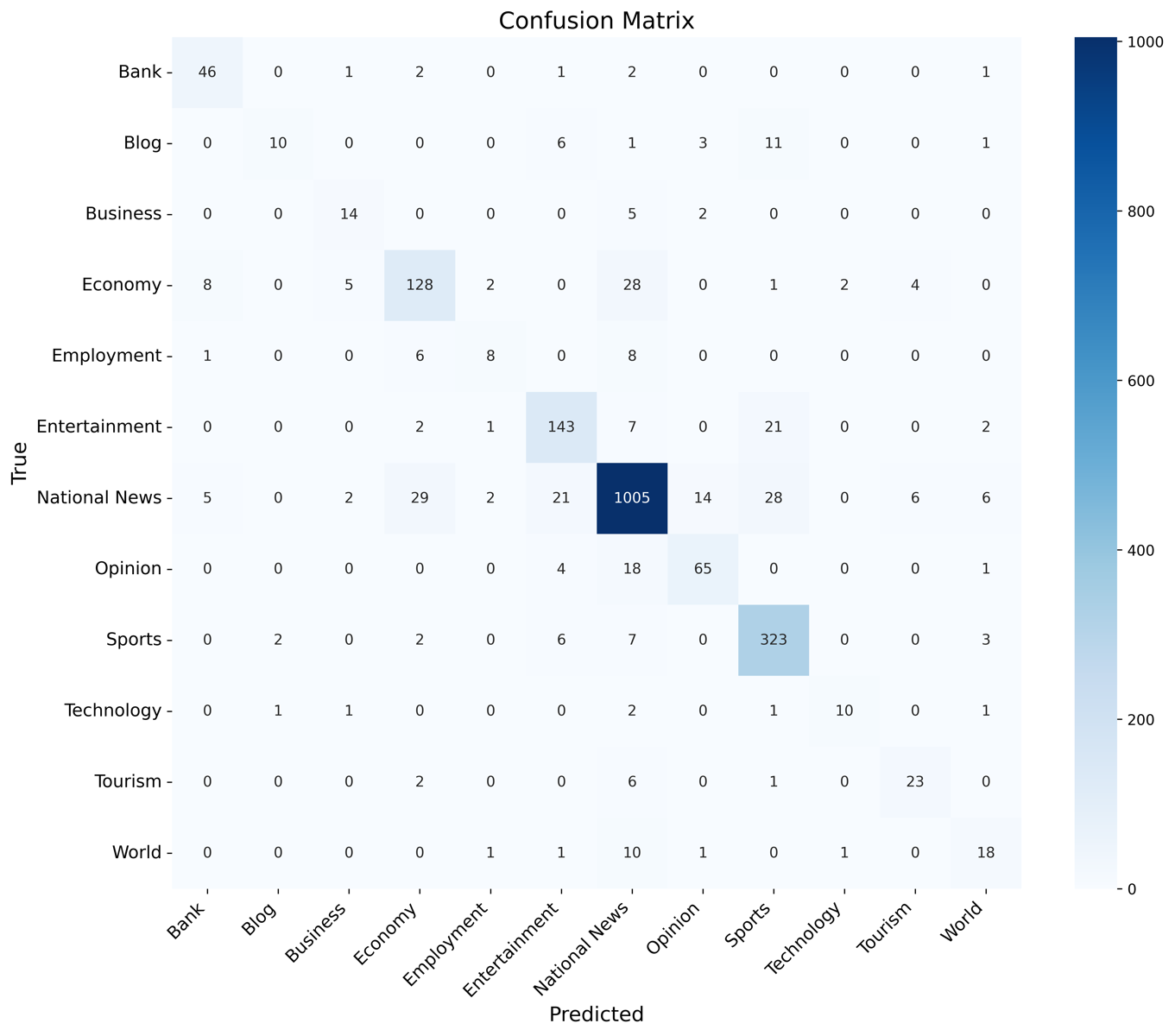


Figure 5.2: 12 x12 News Categories Confusion Matrix for BERT

The model performs exceptionally well for National News (1005 correct predictions), Sports (323 correct), and Entertainment (143 correct), demonstrating strong classification accuracy for these dominant classes. However, significant challenges emerge with minority classes: Blog achieves only 10 correct predictions and it frequently confused with Sports and Business, World has just 18 correct predictions which often misclassified as Entertainment, and Employment shows only 8 correct predictions which commonly confused with Economy.

Table 5.4: Per-Class Precision, Recall, F1-score and Accuracy for BERT Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Precision** | **Recall** | **F1-Score** | **Accuracy** |
| National News | 0.914468 | 0.898927 | 0.906631 | 0.898927 |
| Sports | 0.836788 | 0.941691 | 0.886145 | 0.941691 |
| Bank | 0.766667 | 0.867925 | 0.814159 | 0.867925 |
| Entertainment | 0.785714 | 0.8125 | 0.798883 | 0.8125 |
| Opinion | 0.764706 | 0.738636 | 0.751445 | 0.738636 |
| Economy | 0.748538 | 0.719101 | 0.733524 | 0.719101 |
| Tourism | 0.69697 | 0.71875 | 0.707692 | 0.71875 |
| Technology | 0.769231 | 0.625 | 0.689655 | 0.625 |
| Business | 0.608696 | 0.666667 | 0.636364 | 0.666667 |
| World | 0.545455 | 0.5625 | 0.553846 | 0.5625 |
| Blog | 0.769231 | 0.3125 | 0.444444 | 0.3125 |
| Employment | 0.571429 | 0.347826 | 0.432432 | 0.347826 |

Among the categories, National News performs the best, with an F1-Score of 0.9066, indicating high reliability. The Sports category also performs well, particularly in recall (0.9417), meaning the model captures most relevant cases. However, some categories, such as Blog and Employment, have significantly lower recall and F1-scores, suggesting difficulty in correct classification. Similarly, Technology and World categories also show lower performance compared to others.

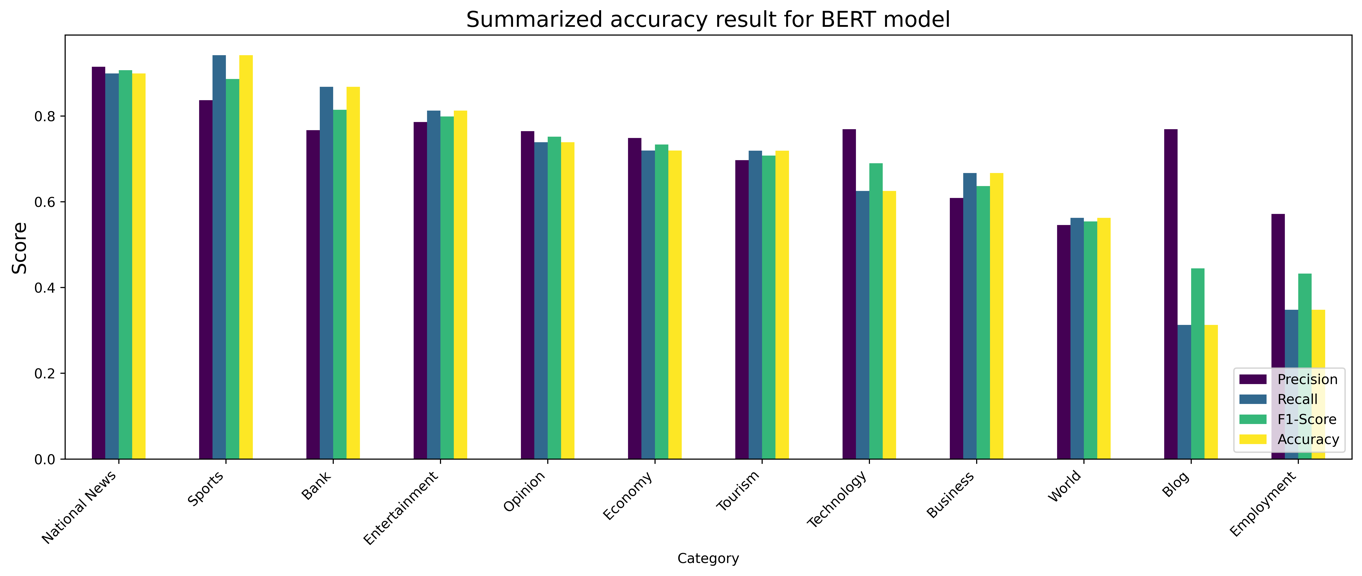


Figure 5.3: Summarized result for BERT model

### Performance of RoBERTa

The confusion matrix for RoBERTa is given below:

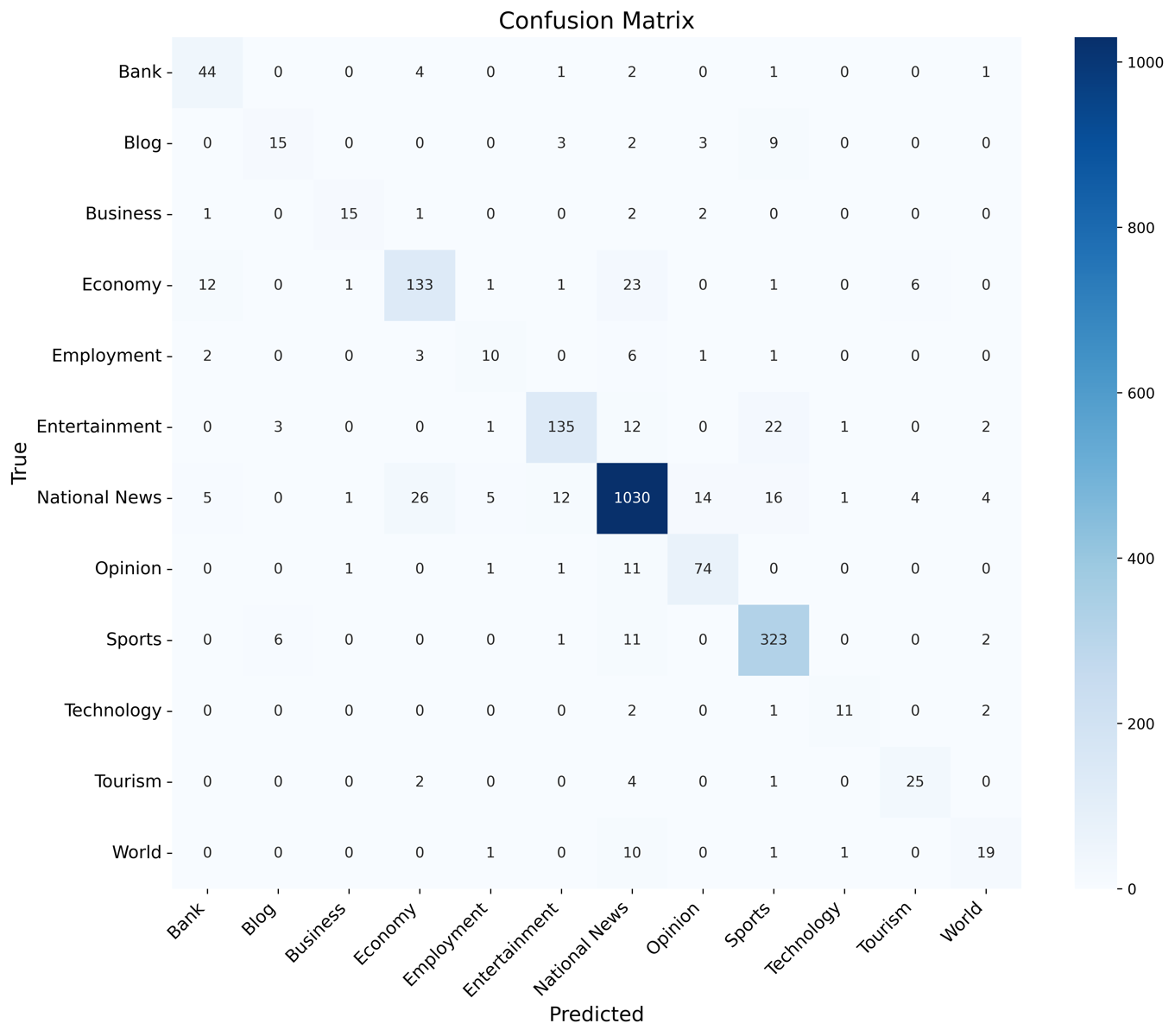


Figure 5.4: 12 x12 News Categories Confusion Matrix for RoBERTa

The RoBERTa model excels at classifying dominant categories like National News, Sports, and Entertainment, demonstrating high accuracy with 1030, 323, and 135 correct predictions, respectively. However, it struggles with minority classes such as Blog, World, and Employment, often misclassifying them due to class imbalance and contextual overlap (e.g., Blog confused with Sports, World with National News).

Table 5.5: Per-Class Precision, Recall, F1-score and Accuracy for RoBERTa Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Precision | Recall | F1-Score | Accuracy |
| National News | 0.923767 | 0.921288 | 0.922526 | 0.921288 |
| Sports | 0.859043 | 0.941691 | 0.89847 | 0.941691 |
| Entertainment | 0.876623 | 0.767045 | 0.818182 | 0.767045 |
| Opinion | 0.787234 | 0.840909 | 0.813187 | 0.840909 |
| Business | 0.833333 | 0.714286 | 0.769231 | 0.714286 |
| Economy | 0.786982 | 0.747191 | 0.766571 | 0.747191 |
| Bank | 0.6875 | 0.830189 | 0.752137 | 0.830189 |
| Tourism | 0.714286 | 0.78125 | 0.746269 | 0.78125 |
| Technology | 0.785714 | 0.6875 | 0.733333 | 0.6875 |
| World | 0.633333 | 0.59375 | 0.612903 | 0.59375 |
| Blog | 0.625 | 0.46875 | 0.535714 | 0.46875 |
| Employment | 0.526316 | 0.434783 | 0.47619 | 0.434783 |

National News performs the best, with an F1-Score of 0.9110, indicating high reliability and balanced precision and recall. The Sports category also achieves strong performance, especially in recall (0.9417), meaning the model is highly effective at identifying sports-related content. On the other hand, the Blog and Employment categories have considerably lower recall (0.3438 and 0.3043, respectively) and F1-scores (0.4231 and 0.4000), suggesting the model struggles to correctly identify instances from these categories. Similarly, the Technology, Tourism, and World categories show relatively lower F1-scores (0.6897, 0.5970, and 0.5517, respectively), pointing to potential areas for improvement.

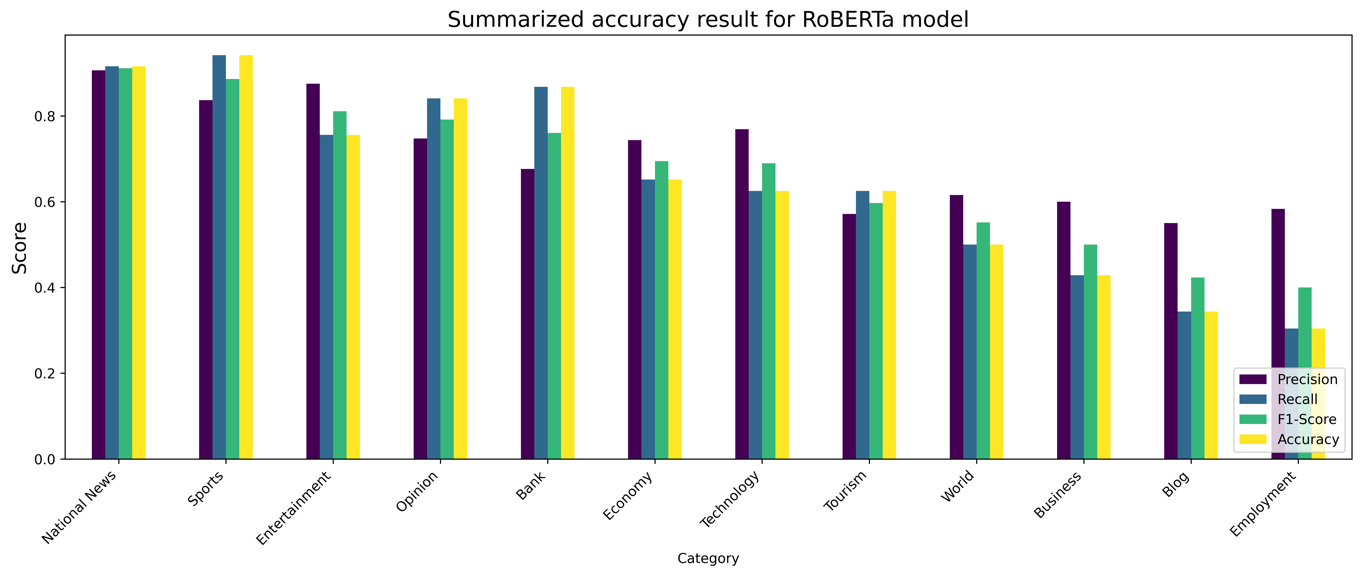


Figure 5.5: Summarized result for RoBERTa model

### Performance of DistilBERT

Since the validation loss began increasing after Epoch 3 during DistilBERT training, we chose the Epoch 3 checkpoint as the optimal model for evaluation. The confusion matrix for DistilBERT is given below:

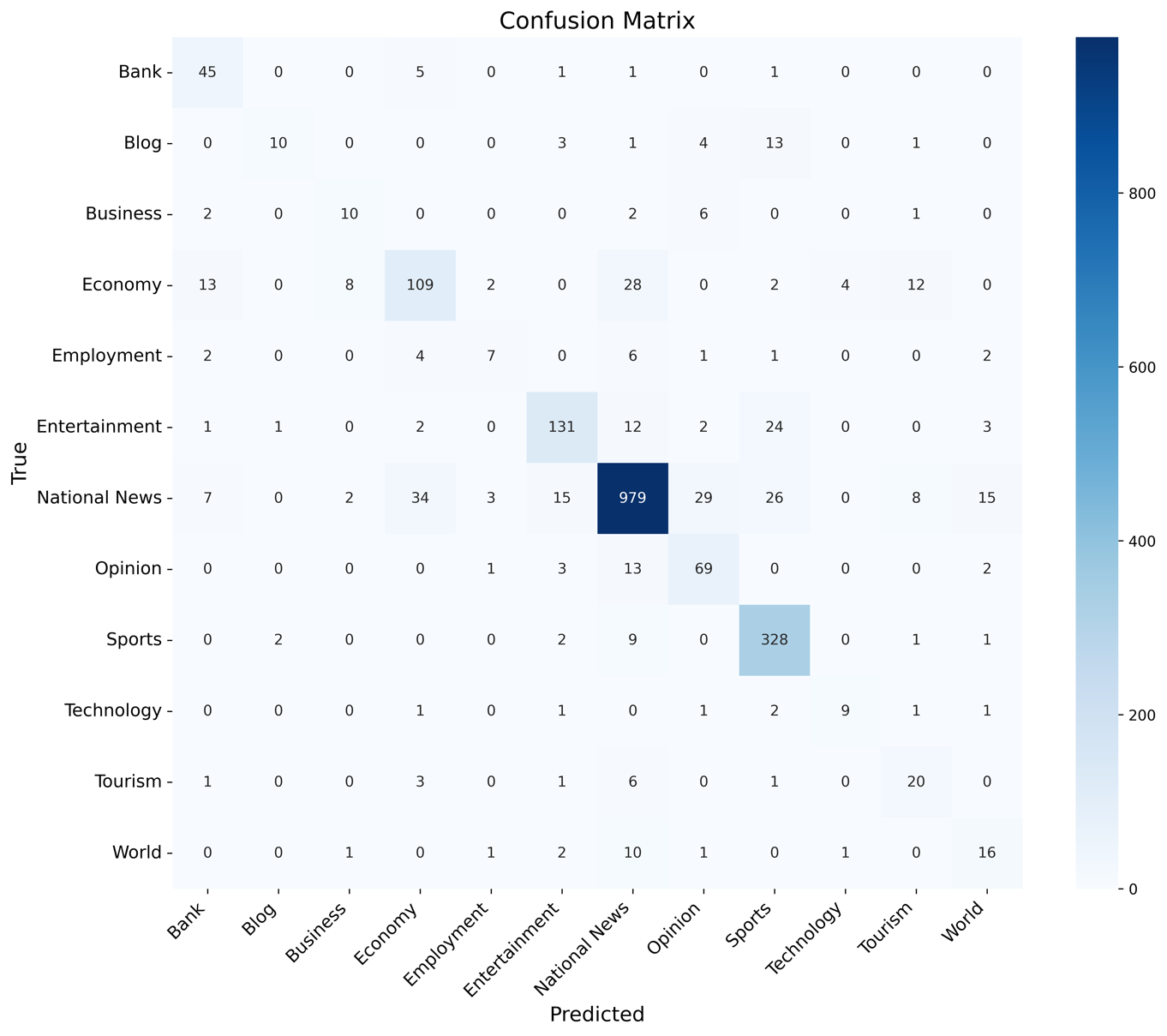


Figure 5.6: 12 x12 News Categories Confusion Matrix for DistilBERT

The confusion matrix analysis reveals that the DistilBERT model performs exceptionally well for dominant categories like National News (979 correct predictions), Sports (328 correct), and Entertainment (131 correct), demonstrating strong precision and recall for these high-volume classes. However, significant performance gaps exist for minority categories such as Blog (only 10 correct predictions), Employment (7 correct), and World (16 correct), which suffer from frequent misclassifications due to both class imbalance and semantic overlap with larger categories. The model particularly struggles with distinguishing between conceptually similar classes like Economy vs. Employment and World vs. National News.

Table 5.6: Per-Class Precision, Recall, F1-score and Accuracy for BERT Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Precision** | **Recall** | **F1-Score** | **Accuracy** |
| National News | 0.917526 | 0.875671 | 0.89611 | 0.875671 |
| Sports | 0.824121 | 0.956268 | 0.88529 | 0.956268 |
| Entertainment | 0.823899 | 0.744318 | 0.78209 | 0.744318 |
| Bank | 0.633803 | 0.849057 | 0.725806 | 0.849057 |
| Opinion | 0.610619 | 0.784091 | 0.686567 | 0.784091 |
| Economy | 0.689873 | 0.61236 | 0.64881 | 0.61236 |
| Technology | 0.642857 | 0.5625 | 0.6 | 0.5625 |
| Tourism | 0.454545 | 0.625 | 0.526316 | 0.625 |
| Business | 0.47619 | 0.47619 | 0.47619 | 0.47619 |
| Blog | 0.769231 | 0.3125 | 0.444444 | 0.3125 |
| World | 0.4 | 0.5 | 0.444444 | 0.5 |
| Employment | 0.5 | 0.304348 | 0.378378 | 0.304348 |

Among the categories, National News showed the best results with an F1-score of 0.896, along with strong precision (0.918) and recall (0.876). Sports also performed well with an F1-score of 0.885, driven by a high recall (0.956). Entertainment and Bank had moderate performance, with F1-scores of 0.782 and 0.726, respectively. Opinion had a good recall (0.784) but lower precision (0.611), leading to an F1-score of 0.687. Economy and Technology categories had lower scores, with F1-scores of 0.649 and 0.600. Tourism, Business, Blog, World, and Employment had weaker performance, with F1-scores below 0.53, meaning the model struggled to classify these accurately. Blog and Employment had the lowest recall (0.312 and 0.304), leading to many misclassifications. Overall, DistilBERT worked well for some categories but needs improvement for others.

A graph of different colored bars

AI-generated content may be incorrect.

Figure 5.7: Summarized result for DistilBERT model

## Comparison of BERT, RoBERTa and DistilBERT models

Based on the confusion matrices, we calculated four key evaluation metrics: overall accuracy, weighted precision, weighted recall, and weighted F1-score. The table below summarizes the comparative performance of BERT, RoBERTa, and DistilBERT:

Table 5.7: Comparative performance result of BERT, RoBERTa and DistilBERT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Overall Accuracy** | **Weighted Precision** | **Weighted Recall** | **Weighted F1-Score** |
| BERT | 0.849 | 0.8482 | 0.849 | 0.8461 |
| **RoBERTa** | **0.8684** | **0.8678** | **0.8684** | **0.8669** |
| DistilBERT | 0.8205 | 0.8273 | 0.8205 | 0.8197 |

The comparison of BERT, RoBERTa, and DistilBERT reveals clear performance differences across all evaluated metrics. RoBERTa emerges as the top-performing model, achieving the highest scores in overall accuracy (0.8684), weighted precision (0.8678), weighted recall (0.8684), and weighted F1-score (0.8669). BERT follows closely behind, with consistent scores around 0.848-0.849 across metrics, demonstrating its reliability as a baseline model. DistilBERT shows slightly lower performance (0.8205 accuracy, 0.8197 F1-score).

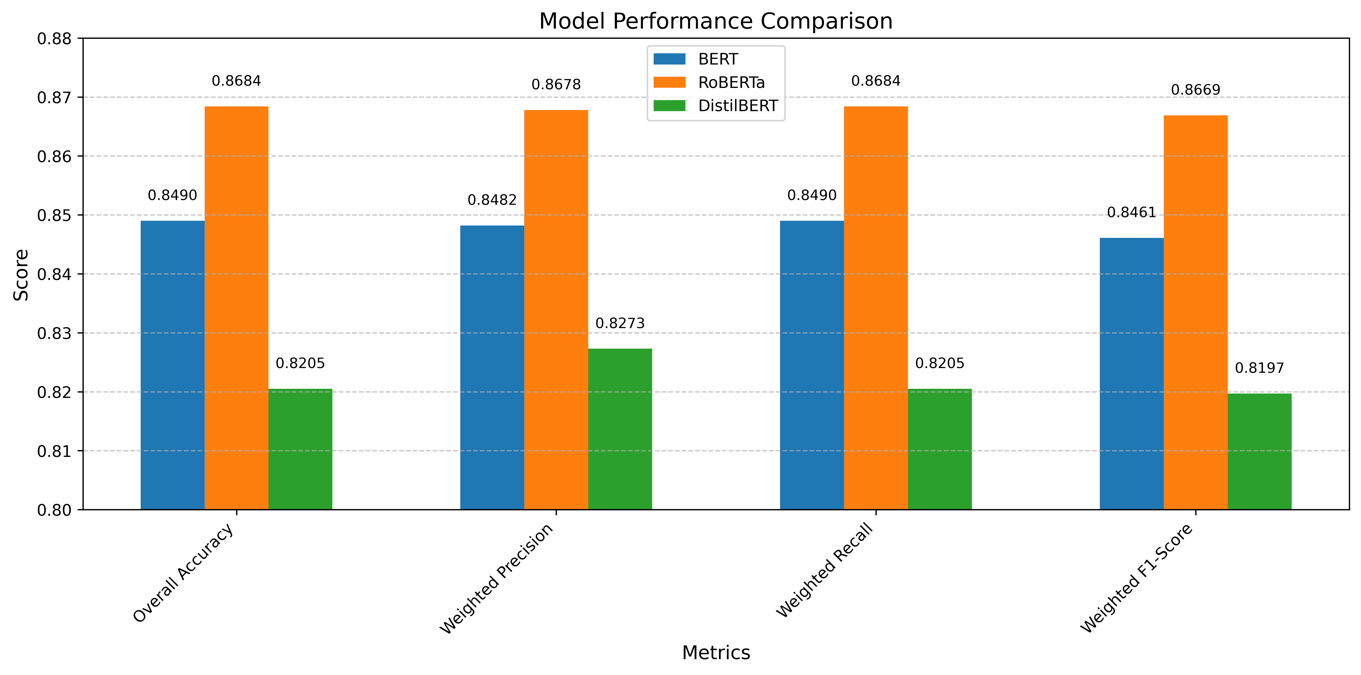


Figure 5.8: BERT, RoBERTa and DistilBERT performance comparison

# Conclusion and Future Recommendation

## Conclusion

The volume of Nepali news content continues to grow rapidly across diverse categories. For this study, news articles were collected from multiple Nepali news portals and systematically labeled according to 12 predefined categories for classification purposes. Transformer-based models - BERT, RoBERTa, and DistilBERT - were implemented and rigorously evaluated for their classification capabilities.

The dataset was partitioned into training, validation, and testing subsets to ensure robust evaluation. Models were fine-tuned with a learning rate of 3e-5 and analyzed across multiple performance metrics. Comparative analysis revealed that RoBERTa achieved superior performance with 86.84% accuracy, along with strong precision (0.8678) and recall (0.8684) scores, yielding an F1-score of 0.8669. BERT demonstrated competitive results (84.9% accuracy), while DistilBERT offered a balance between performance (82.05% accuracy) and computational efficiency. DistilBERT offers faster and lighter training and inference with low consumption of memory.

The results validate the viability of fine-tuned transformers for low-resource languages, though challenges persist in handling imbalanced data. While the study successfully classified dominant categories like National News and Sports, performance limitations were observed for minority classes such as Blog and Employment, primarily due to class imbalance and contextual similarities between categories.

## Limitations and Future Recommendation

The study utilized 14,081 Nepali news articles across 12 categories, encountering several challenges:

1. Data scarcity for minority classes (Blog at 1.06%, Employment at 1.41%) impaired model generalization
2. Semantic overlap between similar categories (Economy/Employment, National News/World) caused classification errors
3. Nepali's morphological complexity and lack of standardized tokenizers adversely affected processing

Future work could expand both the dataset size and category breadth, while exploring additional machine learning architectures for potentially improved results.

References

[1] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, “Deep learning–based text classification: a comprehensive review,” *ACM computing surveys (CSUR)*, vol. 54, no. 3, pp. 1–40, 2021, doi: https://doi.org/10.1145/3439726.

[2] T. B. Shahi and A. K. Pant, “Nepali news classification using Naive Bayes, support vector machines and neural networks,” in *2018 international conference on communication information and computing technology (iccict)*, 2018, pp. 1–5.

[3] H. Wang, J. Li, H. Wu, E. Hovy, and Y. Sun, “Pre-trained language models and their applications,” *Engineering*, 2022.

[4] S. Subba, N. Paudel, and T. B. Shahi, “Nepali Text Document Classification Using Deep Neural Network,” *Tribhuvan University Journal*, vol. 33, no. 1, pp. 11–22, Jun. 2019, doi: 10.3126/tuj.v33i1.28677.

[5] Q. Li *et al.*, “A survey on text classification: From traditional to deep learning,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 2, pp. 1–41, 2022.

[6] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, “Text classification algorithms: A survey,” *Information*, vol. 10, no. 4, p. 150, 2019.

[7] A. Gasparetto, M. Marcuzzo, A. Zangari, and A. Albarelli, “A survey on text classification algorithms: From text to predictions,” *Information*, vol. 13, no. 2, p. 83, 2022.

[8] L. Yao, C. Mao, and Y. Luo, “Graph convolutional networks for text classification,” in *Proceedings of the AAAI conference on artificial intelligence*, 2019, pp. 7370–7377.

[9] A. Vaswani *et al.*, “Attention is all you need,” *Adv Neural Inf Process Syst*, vol. 30, 2017.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[11] Y. Liu *et al.*, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.

[12] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” *arXiv preprint arXiv:1910.01108*, 2019.

[13] S. V *et al.*, “The DistilBERT Model: A Promising Approach to Improve Machine Reading Comprehension Models,” *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 11, no. 8, pp. 293–309, Sep. 2023, doi: 10.17762/ijritcc.v11i8.7957.

[14] K. Salehin, M. K. Alam, Md. A. Nabi, F. Ahmed, and F. Bin Ashraf, “A Comparative Study of Different Text Classification Approaches for Bangla News Classification,” in *2021 24th International Conference on Computer and Information Technology (ICCIT)*, 2021, pp. 1–6. doi: 10.1109/ICCIT54785.2021.9689843.

[15] F. Rollo, G. Bonisoli, and L. Po, “A Comparative Analysis of Word Embeddings Techniques for Italian News Categorization,” *IEEE Access*, vol. 12, pp. 25536–25552, 2024.

[16] S. Timilsina, M. Gautam, and B. Bhattarai, “NepBERTa: Nepali language model trained in a large corpus,” in *Proceedings of the 2nd conference of the Asia-pacific chapter of the association for computational linguistics and the 12th international joint conference on natural language processing*, 2022.

[17] S. R. Tamrakar and C. Silpasuwanchai, “Comparative evaluation of transformer-based Nepali language models,” 2022.

[18] 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, 2024.

[19] G. Manias, A. Mavrogiorgou, A. Kiourtis, C. Symvoulidis, and D. Kyriazis, “Multilingual text categorization and sentiment analysis: a comparative analysis of the utilization of multilingual approaches for classifying twitter data,” *Neural Comput Appl*, vol. 35, no. 29, pp. 21415–21431, May 2023, doi: 10.1007/s00521-023-08629-3.

[20] M. Grandini, E. Bagli, and G. Visani, “Metrics for multi-class classification: an overview,” *arXiv preprint arXiv:2008.05756*, 2020.

[21] “BERT.” Accessed: Apr. 01, 2025. [Online]. Available: https://huggingface.co/docs/transformers/en/model\_doc/bert#transformers.BertTokenizer

Appendix

1. Snippet of code for data crawling from ekantipur

import requests

from bs4 import BeautifulSoup

# Target categories

categories = {

"business": "https://ekantipur.com/business/",

"world": "https://ekantipur.com/world/",

"sports": "https://ekantipur.com/sports/"

}

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)'

}

for category, url in categories.items():

res = requests.get(url, headers=headers)

soup = BeautifulSoup(res.text, 'html.parser')

print(f"\n--- {category.upper()} ---\n")

for item in soup.select('.normal h2'):

title = item.text.strip()

link = item.a['href'] if item.a else None

full\_link = link if link.startswith("https") else url.split(f"/{category}")[0] + link

print(f"Title: {title}")

print(f"Link: {full\_link}\n")

1. Snippet of code for data loading into csv file

import os

import pandas as pd

#raw data directory

dataset\_root\_dir = '0\_raw\_dataset\_large'

# Initialize empty lists to store file names, labels, and contents

file\_names = []

labels = []

contents = []

# Traverse through each folder

for folder in os.listdir(dataset\_root\_dir):

folder\_path = os.path.join(dataset\_root\_dir, folder)

if os.path.isdir(folder\_path):

for file in os.listdir(folder\_path):

if file.endswith('.txt'):

file\_path = os.path.join(folder\_path, file)

with open(file\_path, 'r', encoding='utf-8', errors='ignore') as txt\_file:

file\_contents = txt\_file.read().strip()

file\_names.append(file)

labels.append(folder) # Store the folder name directly as the label

contents.append(file\_contents)

# Create a DataFrame from the lists with labels as folder names directly

df = pd.DataFrame({

'file\_id': file\_names,

'content': contents,

'label': labels

})

label\_counts = df['label'].value\_counts()

print(label\_counts)

# Sort the DataFrame by the 'label' column

df = df.sort\_values(by='label', ascending=True)

# Display the DataFrame with labels as folder names

print(df)

# Specify the path where you want to save the CSV file

combined\_data\_path = '1\_combined\_dataset/combined.csv'

df.to\_csv(combined\_data\_path, index=False)

print(f"Data saved to {combined\_data\_path}")

1. Snippet of code for label encoding

import pandas as pd

import json

from sklearn.preprocessing import LabelEncoder

# Load the data

data = pd.read\_csv("1\_combined\_dataset/combined.csv")

# Encode labels and save mapping

label\_encoder = LabelEncoder()

data['label'] = label\_encoder.fit\_transform(data['label'])

label\_mapping = {label: int(code) for label, code in zip(label\_encoder.classes\_, label\_encoder.transform(label\_encoder.classes\_))}

# Save label mapping to JSON

with open("1\_combined\_dataset/label\_mapping.json", "w") as file:

json.dump(label\_mapping, file)

# Define your preprocess function if not already defined

def preprocess(text):

return text.lower() # Example: convert to lowercase

# Preprocess the 'content' column

data['content'] = data['content'].apply(preprocess)

# Save the encoded and preprocessed data back to CSV

data.to\_csv("1\_combined\_dataset/encoded\_combined.csv", index=False)

print(data)

1. Snippet of code for data cleaning

import re

import pandas as pd

# Load Nepali stopwords from file

nepali\_stopwords = []

with open("0\_nepali\_corpus/nepali\_stopwords.txt", "r", encoding='utf-8', errors='ignore') as file:

nepali\_stopwords = [line.strip() for line in file]

stopwords = set(nepali\_stopwords

# Read the dataset

data = pd.read\_csv("1\_combined\_dataset/encoded\_combined.csv")

# Preprocess the dataset, including stopword removal

for i in range(len(data)):

text = data.at[i, 'content']

text = re.sub(r'[a-zA-Z0-9!@#$%^&\*()\_+\-=\[\]{};\'‘\\:"|<,./<>?~`।१२३४५६७८९०–÷]', '', text)

text = re.sub(r'।', '', text)

text = re.sub(r'\ufeff', '', text)

text = re.sub(r'\\u200d', '', text)

text = text.replace('\n', ' ')

# Split text into words and remove stopwords

words = text.split()

text = ' '.join(word for word in words if word not in stopwords)

# Update the DataFrame

data.at[i, 'content'] = text

# Save preprocessed data

data.to\_csv("2\_preprocessed\_dataset/preproccessed\_data.csv", index=False)

print("Preprocessed data saved to '2\_preprocessed\_dataset/preproccessed\_data.csv'")

1. Snippet of code for data splitting

# Stratified split into train and test sets and validation sets

train\_data, test\_data = train\_test\_split(data, test\_size=0.3, stratify=labels, random\_state=42)

validation\_data, test\_data = train\_test\_split(test\_data, test\_size=0.5, stratify=test\_data['label'], random\_state=42)

# Convert splits back to Hugging Face Datasets

train\_dataset = Dataset.from\_pandas(train\_data)

validation\_dataset = Dataset.from\_pandas(validation\_data)

test\_dataset = Dataset.from\_pandas(test\_data)

# Save train, validation, and test datasets

# Convert Hugging Face Datasets to Pandas DataFrames before saving

train\_df = train\_dataset.to\_pandas()

validation\_df = validation\_dataset.to\_pandas()

test\_df = test\_dataset.to\_pandas()

# Save as CSV files

train\_df.to\_csv("3\_split\_data/train.csv", index=False)

validation\_df.to\_csv("3\_split\_data/validation.csv", index=False)

test\_df.to\_csv("3\_split\_data/test.csv", index=False)

1. Snippet of code for data loading into hugging face dataset

from datasets import load\_dataset, DatasetDict

train\_data = load\_dataset('csv', data\_files='3\_split\_data/train.csv')

test\_data = load\_dataset('csv', data\_files='3\_split\_data/test.csv')

validation\_data = load\_dataset('csv', data\_files='3\_split\_data/validation.csv')

dataset = DatasetDict({

'train': train\_data['train'],

'test': test\_data['train'],

'validation':validation\_data['train']

})

print(dataset)

1. Snippet of code for BERT Tokenization

# Load BERT tokenizer

tokenizer = BertTokenizer.from\_pretrained('bert-base-multilingual-uncased')

def tokenize\_function(examples):

contents = [str(content) for content in examples['content']]

return tokenizer(contents, padding="max\_length", truncation=True, max\_length=512)

tokenized\_dataset = dataset.map(tokenize\_function, batched=True)

print(tokenized\_dataset)

1. Snippet of code for RoBERTa Tokenization

# Load RoBERTa tokenizer

tokenizer = AutoTokenizer.from\_pretrained('xlm-roberta-base')

def tokenize\_function(examples):

contents = [str(content) for content in examples['content']]

return tokenizer(contents, padding="max\_length", truncation=True, max\_length=512)

tokenized\_dataset = dataset.map(tokenize\_function, batched=True)

print(tokenized\_dataset)

1. Snippet of code for DistilBERT Tokenization

# Load DistilBERT tokenizer

from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from\_pretrained("distilbert/distilbert-base-multilingual-cased")

def tokenize\_function(examples):

contents = [str(content) for content in examples['content']]

return tokenizer(contents, padding="max\_length", truncation=True, max\_length=512)

tokenized\_dataset = dataset.map(tokenize\_function, batched=True)

print(tokenized\_dataset)

1. Sample Snippet of code for BERT Training

from transformers import TrainingArguments, Trainer

from torch.optim import AdamW

from sklearn.metrics import accuracy\_score, precision\_recall\_fscore\_support

import numpy as np

def compute\_metrics(p):

preds = p.predictions.argmax(-1)

labels = p.label\_ids

accuracy = accuracy\_score(labels, preds)

precision, recall, f1, \_ = precision\_recall\_fscore\_support(labels, preds, average='weighted')

return {

'accuracy': accuracy,

'precision': precision,

'recall': recall,

'f1': f1

}

batch\_size = 10

no\_epochs = 4

train\_len = len(dataset['train'])

steps = (train\_len \* no\_epochs) / batch\_size

training\_args = TrainingArguments(

eval\_strategy="epoch",

logging\_strategy="steps",

save\_strategy="epoch",

output\_dir='./nepali\_news\_mbert\_fine\_tuned\_model',

logging\_dir='./logs',

logging\_steps=train\_len // batch\_size,

per\_device\_train\_batch\_size=batch\_size,

per\_device\_eval\_batch\_size=batch\_size,

num\_train\_epochs=no\_epochs,

weight\_decay=0.05,

learning\_rate=3e-5,

warmup\_steps=int(0.05 \* steps),

load\_best\_model\_at\_end=True,

metric\_for\_best\_model="eval\_loss",

greater\_is\_better=False,

save\_total\_limit=None,

bf16=True,

gradient\_checkpointing=False

)

trainer = Trainer(

model=model,

args=training\_args,

train\_dataset=tokenized\_dataset['train'],

eval\_dataset=tokenized\_dataset['validation'],

optimizers=(AdamW(model.parameters(), lr=training\_args.learning\_rate), None),

compute\_metrics=compute\_metrics

)

trainer.train()

1. Figure 4.4 a: Snippet of code for prediction using BERT

from transformers import AutoModelForSequenceClassification, AutoTokenizer

import torch

import json

# Load model and tokenizer

model\_name = 'bert-base-multilingual-uncased'

path\_to\_model = './nepali\_news\_mbert\_fine\_tuned\_model'

model = AutoModelForSequenceClassification.from\_pretrained(path\_to\_model, num\_labels=12)

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

# Text to predict

text\_to\_predict = "डिफेन्डिङ च्याम्पियन अर्जेन्टिना बुधबार विश्वकप २०२६ मा छनोट हुने दोस्रो टिम भएको छ।"

# Tokenize

inputs = tokenizer(text\_to\_predict, return\_tensors="pt", truncation=True, padding=True, max\_length=512)

with torch.no\_grad():

outputs = model(\*\*inputs)

# Load label encode mapping

label\_mapping\_path = '1\_combined\_dataset/label\_mapping.json'

with open(label\_mapping\_path, 'r') as json\_file:

label\_mapping = json.load(json\_file)

id\_to\_label = {v: k for k, v in label\_mapping.items()}

# Apply softmax to get probabilities

probs = torch.softmax(outputs.logits, dim=1)

predicted\_prob, predicted\_class\_id = torch.max(probs, dim=1)

# Convert to Python values

predicted\_prob = predicted\_prob.item() # Confidence score

predicted\_class\_id = predicted\_class\_id.item() # Class ID

predicted\_label = id\_to\_label[predicted\_class\_id] # Class name

# Get all class probabilities (optional)

all\_probs = probs.squeeze().tolist() # Convert to list

class\_probabilities = {id\_to\_label[i]: prob for i, prob in enumerate(all\_probs)}

print(f"Predicted class: {predicted\_label} (confidence: {predicted\_prob:.2%})")

print("\nAll class probabilities:")

for cls, prob in class\_probabilities.items():

print(f"{cls}: {prob:.2%}")