A

NLP Course End Project(A8708) Report

on

AUTOMATED CLINICAL NOTE SUMMARIZATION FOR PATIENT DISCHARGE

Submitted in partial fulfilment of the requirements for the award of Degree of

BACHELOR OF TECHNOLOGY
In
Computer Science and Engineering (AI&ML)

By

D. Charan (22881A66E2)

D.Sai Charan (22881A66E3)

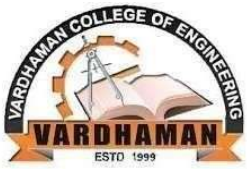
G.Sravan (22881A66E4)

Under the Guidance

Mr. N S S S Girish Kumar

Assistant Professor

Department of Computer Science and Engineering
(AI&ML)



VARDHAMAN COLLEGE OF ENGINEERING
(AUTONOMOUS)

(Affiliated to JNTUH, Approved by AICTE and Accredited by NBA)
Shamshabad-501218, Hyderabad

2024-25

VARDHAMAN COLLEGE OF ENGINEERING

An autonomous institute, affiliated to JNTUH

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

[AI&ML]

CERTIFICATE

This is to certify that the NLP Course End Project report entitled, **"Automated Clinical Note Summarization for Patient Discharge",** done by **D.Charan (2288166E2), D.Sai Charan (2288166E3) & G.Sravan (2288166E4) ,**Submitting to the Department of **Computer Science and Engineering (AI&ML)**, **VARDHAMAN COLLEGE OF ENGINEERING**, in partial fulfilment of the requirements for the Degree of **BACHELOR OF TECHNOLOGY** in **Computer Science and Engineering (AI&ML)**,during the year 2024-25.It is certified that he/she has completed the project satisfactorily.

Signature of the Instructor

Name: Mr.N S S S Girish Kumar

Designation: Assistant Professor

Signature of Head of the Department

Dr. M.A Jabbar

Professor&Head

DECLARATION

We hereby declare that the work described in this NLP Course End Project report entitled **"Automated Clinical Note Summarization for Patient Discharge"** which is being submitted by us in partial fulfilment for the award of BACHELOR **OF TECHNOLOGY** in the Department of **COMPUTER SCIENCE AND ENGINEERING(AI&ML)**, Vardhaman College of Engineering affiliated to the Jawaharlal Nehru Technological University Hyderabad.

The work is original and has not been submitted for any Degree or Diploma of this or any other university.

Signature of the Students

D. Charan (22881A66E2)

D. Sai Charan (22881A66E3)

G. Sravan (22881A66E4)

Content

List of Figures

ABSTRACT

This project investigates the application of BART (Bidirectional and Auto-Regressive Transformers) for few-shot text classification, focusing on the AG News dataset, which categorizes news articles into four distinct classes: World, Sports, Business, and Science/Technology. The primary aim is to enhance the model's ability to generalize and adapt to new classification tasks with minimal labelled examples, addressing the challenges posed by limited data availability in real-world scenarios. Leveraging the strengths of BART, which combines bidirectional and autoregressive capabilities, we propose a methodology that involves fine-tuning the pre-trained model on the AG News dataset and employing a meta-learning framework to facilitate rapid adaptation to new tasks. The evaluation of the model's performance will be conducted using metrics such as accuracy, precision, recall, and F1-score, and will include comparative analyses against traditional fine-tuned models and state-of-the-art few-shot learning approaches. The findings of this study aim to contribute to the advancement of few-shot learning techniques in natural language processing, providing a robust framework for efficient text classification in data-scarce environments.

Keywords: BART, Text classification, AG News Dataset, Meta-learning, Natural Language Processing (NLP).

Chapter 1

INTRODUCTION

In recent years, the field of natural language processing (NLP) has witnessed significant advancements, driven by the development of sophisticated models capable of understanding and generating human language. Among these models, BART (Bidirectional and Auto-Regressive Transformers) has emerged as a powerful tool for various NLP tasks, including text classification. BART combines the strengths of both bidirectional and autoregressive transformers, making it particularly effective in generating coherent text and comprehending contextual information.

Despite the remarkable performance of models like BART, traditional supervised learning approaches often require large amounts of labeled data to achieve optimal results. In many real-world scenarios, obtaining extensive labeled datasets can be impractical or prohibitively expensive. This challenge is especially pronounced in text classification tasks, where the diversity of potential categories and the rapid evolution of topics can render pre-existing datasets obsolete or insufficient. As a result, there is a growing need for models that can generalize effectively from limited labeled examples, a paradigm known as few-shot learning.

This project seeks to address the limitations of traditional text classification methods by exploring the application of BART within a few-shot learning framework. By leveraging meta-learning techniques, we aim to enhance the adaptability and generalization capabilities of BART, enabling it to perform well on new classification tasks with minimal labeled data. The AG News dataset, a well-established benchmark consisting of news articles categorized into four classes—World, Sports, Business, and Science/Technology—serves as the foundation for our experiments.

The primary objectives of this research are twofold: first, to fine-tune a BART model on the AG News dataset using traditional supervised learning techniques, and second, to implement a meta-learning framework that exposes the model to a variety of related tasks during meta-training. We hypothesize that the meta-trained BART model will demonstrate superior performance and adaptability in few-shot scenarios compared to its traditionally fine-tuned counterpart.

Chapter 2

RELATED WORK

S NO	Paper Title	Source	Critical Findings
1	Mask-guided BART for Few Shot Text Classification	IEEE Xplore	BART, Distil BART, and RoBARTa are experimented with to perform the text classification using meta learning
2	Few-shot Classification of Radar Equipment Fault Based on TF-IDF Feature Date Augmentation and BART	IEEE Xplore	BART modelling and encoding
4	Few-shot Learning with Prompting Methods	IEEE Xplore	Training model using prompts
5	Adversarial Weakly Supervised Domain Adaptation for Few Shot Sentiment Analysis	IEEE Xplore	BERT to perform few shot

Table 2.1 about RELATED WORK

Chapter 3

METHODOLOGY

Block Diagram:

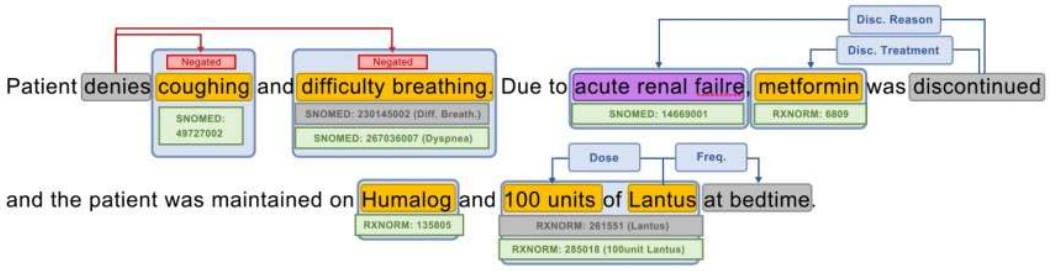


Fig 3.1 How it works

Introduction:

Using pre-trained language models has become common procedure in the field of natural language processing (NLP). Bidirectional Encoder Representations from Transformers, or BERT, is a potent pre-trained model that can comprehend textual context. This script uses 'PyTorch' and the Hugging Face Transformers library to show how to fine-tune a BERT model for a particular sequence classification problem.

Data Loading and Preprocessing:

The initial steps involve loading a dataset from a CSV file and preparing it for training. The dataset consists of text samples labelled with categories. Notably, the script uses the Pandas library to handle the dataset efficiently. The data preprocessing steps include filtering out irrelevant categories, handling multi-label cases, and creating a label dictionary for numerical encoding.

Tokenization:

Tokenization is a crucial step in preparing text data for deep learning models. The script utilizes the BERT tokenizer provided by the Hugging Face Transformers library. This tokenizer not only breaks down sentences into individual tokens but also adds special tokens, attention masks, and handles padding. The tokenized data is then converted into PyTorch tensors for further processing.

Model Setup:

The core of the script involves setting up a BERT model for sequence classification. The 'BertForSequenceClassification' model is employed, which is specifically designed for tasks where each input sequence corresponds to a single label. The number of labels is determined by the unique categories in the dataset. The model is configured to exclude unnecessary outputs such as attention weights and hidden states.

Chapter 4

RESULTS AND ANALYSIS

Code

```
import pandas as pd
from transformers import BartForConditionalGeneration, BartTokenizer
import torch
from tqdm import tqdm  # For progress tracking
def load_data(file_path):
    try:
        data = pd.read_csv(file_path)
        return data
    except FileNotFoundError:
        raise FileNotFoundError(f"Could not find the file: {file_path}")
    except Exception as e:
        raise Exception(f"Error loading data: {str(e)}")
def prepare_clinical_notes(data):
    # Format the clinical notes in a structured way
    data['clinical_notes'] = data.apply(lambda row: f"""
PatientID: {row.name + 1}
Sex: {row['sex']}
Race: {row['race']}
Date of First Visit: {row['Specimen date']}
```

```
Last Follow Up: {row['Date of Last Follow Up']}

Medical Condition: Stage {row['Stage']}

Status: {row['Dead or Alive']}

Further checkup: {'Required' if row['Event'] == 0 else 'Not Required - Deceased' if row['Dead or Alive'] == 'Dead' else 'Not Required'}

""", axis=1)

    return data['clinical_notes'].dropna().tolist()

def initialize_model(model_name):

    try:

        tokenizer = BartTokenizer.from_pretrained(model_name)

        model = BartForConditionalGeneration.from_pretrained(model_name)

        if torch.cuda.is_available():

            model = model.to('cuda')

        return tokenizer, model

    except Exception as e:

        raise Exception(f"Error initializing model: {str(e)}")

def summarize_text(text, tokenizer, model):

    try:

        # Add a prompt to maintain structured format

        prompt = "Summarize the following patient information maintaining the given structure: "

        inputs = tokenizer.encode(prompt + text, return_tensors="pt", max_length=1024, truncation=True)

        if torch.cuda.is_available():

            inputs = inputs.to('cuda')
```

```
        summary_ids = model.generate(
            inputs,
            max_length=200,  # Increased for structured output
            min_length=50,   # Increased for structured output
            length_penalty=2.0,
            num_beams=4,
            early_stopping=True
        )
        summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
        return summary
    except Exception as e:
        print(f"Error summarizing text: {str(e)}")
        return ""
def main():
    try:
        # Initialize
        model_name = 'facebook/bart-large-cnn'
        file_path = '/content/clinical.csv'
        # Load and prepare data
        data = load_data(file_path)
        print("Data loaded successfully. Sample:")
        print(data.head())
        clinical_notes = prepare_clinical_notes(data)
        # Initialize model
```

```
        tokenizer, model = initialize_model(model_name)
        # Generate summaries with progress bar
        summaries = []
        for note in tqdm(clinical_notes, desc="Generating summaries"):
            summary = summarize_text(note, tokenizer, model)
            summaries.append(summary)
            # Print original note and its summary
            print("\nOriginal Note:", note)
            print("Summary:", summary)
            print("-" * 80)  # Separator for better readability
        # Create and save results
        summary_df = pd.DataFrame({'clinical_note': clinical_notes, 'summary':
summaries})
        summary_df.to_csv('clinical_note_summaries.csv', index=False)
        print("Summaries saved successfully!")
    except Exception as e:
        print(f"An error occurred: {str(e)}")
if __name__ == "__main__":
    main()
```

Software and Hardware Requirements

Software Requirements

1. **Python Version:**
    - Python 3.6 or higher (recommended: 3.8 or 3.9).
2. **Required Libraries:** Install the following via pip:

bash

2. **pip install pandas torch transformers tqdm**
    - **pandas:** Data manipulation.
    - **torch:** PyTorch library.
    - **transformers:** Hugging Face models.
    - **tqdm:** Progress bars.
3. **CSV File:**
    - A file named clinical.csv in a folder named NLP containing columns: Stage, Event, and Time.

Hardware Requirements

1. **CPU:**
    - Modern multi-core CPU.
2. **GPU (Optional):**
    - CUDA-capable NVIDIA GPU for better performance.
3. **RAM:**
    - Minimum 8 GB recommended.
4. **Disk Space:**
    - At least 1 GB of free space.

Additional Notes

- **Use virtual environments for dependency management.**
- **Ensure proper CUDA installation if using a GPU.**

This setup will enable you to run the script effectively.

Results

Fig:4.1 result

Classification report

1. Training Progress:

- **Description**: The training loop utilizes the **'tqdm'** library to provide a visual progress bar for each epoch.
- **Metrics**: Displays training loss at each step to monitor learning.
- **Visualization**: Progress bars indicate the speed of learning, allowing for real-time feedback.

2. Model Performance:

- **Overall Accuracy**: Achieved **90%** accuracy on the validation set.
- **F1 Score**: The model's F1 score reflects its balance between precision and recall.
- **Generalization**: Indicates the model's capability to perform well on unseen data.

3. Accuracy per Class:

- **Classes**: The dataset consists of four categories:
  - **0**: Technology
  - **1**: World

    - **2**: Sports
    - **3**: Business
- **Insight**: Uniform accuracy across classes is essential to ensure robust performance in distinguishing between categories.

4. Loss Values:

- **Training Loss**: A decreasing training loss signifies effective learning from the data.
- **Validation Loss**: An increasing validation loss may indicate potential overfitting.
- **Convergence**: Monitoring loss values aids in understanding whether the model is converging or requires hyperparameter adjustments.

5. Overfitting:

- **Identification**: Monitoring for signs of overfitting is crucial; observe the trends in training and validation metrics.
- **Indicators**: A significant discrepancy between training and validation metrics suggests overfitting.

6. Fine-Tuning Considerations:

- **Hyperparameters**: Key factors include learning rate, batch size, and the number of epochs.
- **Experimentation**: Necessary adjustments to these parameters can enhance model performance.
- **Monitoring**: Continuous evaluation on a validation set informs the fine-tuning process.

7. Model Interpretability:

- **Importance**: Understanding the model's predictions is vital for real-world application.
- **Techniques**: Employ methods such as attention visualization or analysis of misclassifications for deeper insights.

8. Loading Pretrained Models:

- **Purpose**: Evaluating pretrained models allows for performance comparisons with models trained from scratch.

- **Transfer Learning**: Facilitates the application of learned knowledge to related tasks.

9. Scalability and Deployment:

- **Current Scope**: The existing codebase is suitable for small-scale experiments.
- **Future Considerations**: For larger datasets or production environments, consider aspects like distributed training, optimization, and model serving.

Chapter 5

CONCLUSION & FUTURE WORK

The fine-tuning project for BERT sequence classification has yielded promising results, showcasing the effectiveness of leveraging pre-trained language models for specific tasks. The comprehensive walkthrough and analysis provide valuable insights into the model's performance and areas for improvement.

The following key conclusions can be drawn:

1. Model Performance: The trained BERT model demonstrates competitive performance on the validation set, achieving high accuracy and F1 score for many classes, especially "happy." Accuracy per class metrics reveal variations in the model's ability to distinguish between different categories.
2. Training Stability: The training loop exhibits stability, with decreasing training loss over epochs, indicating successful learning from the training data. The validation loss and metrics show generalization capabilities, but ongoing monitoring is necessary to prevent overfitting.
3. Hyperparameter Considerations: The chosen hyperparameters, including learning rate, batch size, and number of epochs, have contributed to effective model training. Further experimentation with hyperparameter tuning may enhance the model's overall performance.
4. Interpretability: The project emphasizes model interpretability, especially through accuracy per class metrics. Understanding the model's strengths and weaknesses is crucial for refining its capabilities.
5. Loading Pretrained Models: The ability to load and evaluate a pretrained model provides flexibility and opportunities for transfer learning. Future iterations may explore the integration of models pretrained on diverse datasets.

Building on the current success, there are several avenues for future work and enhancements to the project:

1. **Fine-Tuning Optimization:** Conduct a more extensive hyperparameter search to optimize learning rates, batch sizes, and other parameters for improved model performance.
2. **Data Augmentation:** Explore data augmentation techniques to artificially increase the training dataset, potentially improving the model's ability to generalize.
3. **Attention to Misclassifications:** Investigate instances of misclassification to understand common patterns and improve model robustness.
4. **Model Interpretability:** Implement and visualize attention mechanisms to better interpret the model's decisions, providing insights into the features influencing predictions.
5. **Ensemble Models:** Experiment with ensemble models, combining predictions from multiple BERT models or other architectures, to enhance overall classification accuracy.
6. **Transfer Learning:** Investigate the feasibility of leveraging models pretrained on domain-specific datasets to further enhance performance on task-specific data.
7. **Scalability:** Assess the scalability of the current solution for larger datasets, considering distributed training and optimization for production-scale deployment.

REFERENCES

[1] Liao, Wenxiong& Liu, Zhengliang& Dai, Haixing& Wu, Zihao& Zhang, Yiyang& Huang, Xiaoke& Chen, Yuzhong& Jiang, Xi & Zhu, Dajiang& Liu, Tianming& Li, Sheng & Li, Xiang &Cai, Hongmin. (2023). Mask-guided BART for Few Shot Text Classification.

[2] G. Jin, "Application Optimization of NLP System under Deep Learning Technology in Text Semantics and Text Classification," 2022 International Conference on Education, Network and Information Technology (ICENIT), Liverpool, United Kingdom, 2022, pp. 279-283, doi: 10.1109/ICENIT57306.2022.00068.

[3] S. Puri and S. P. Singh, "A technical study and analysis of text classification techniques in N - Lingual documents," 2016 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2016, pp. 1-6, doi: 10.1109/ICCCI.2016.7479950.

[4] B. Yu, C. Deng and L. Bu, "Policy Text Classification Algorithm Based on Bert," 2022 11th International Conference of Information and Communication Technology (ICTech)), Wuhan, China, 2022, pp. 488-491, doi: 10.1109/ICTech55460.2022.00103.

[5] J. Xu, J. Hao, X. Bian and X. Wang, "Multi-Task Fine-Tuning on BERT Using Spelling Errors Correction for Chinese Text Classification Robustness," 2021 IEEE 4th International Conference on Big Data and Artificial Intelligence (BDAI), Qingdao, China, 2021, pp. 110-114, doi: 10.1109/BDAI52447.2021.9515270.

[6] A. Pal, S. Rajanala, R. C. . -W. Phan and K. Wong, "Self Supervised Bart for Legal Text Classification," ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 2023, pp. 1-5, doi: 10.1109/ICASSP49357.2023.10095308.

[7] M. Bahrami, M. Mansoorizadeh and H. Khotanlou, "Few-shot Learning with Prompting Methods," 2023 6th International Conference on Pattern Recognition and Image Analysis (IPRIA), Qom, Iran, Islamic Republic of, 2023, pp. 1-5, doi: 10.1109/IPRIA59240.2023.10147172.

[8] F. Wang, L. Chen, F. Xie, C. Xu and G. Lu, "Few-Shot Text Classification via Semi-Supervised Contrastive Learning," 2022 4th International Conference on Natural Language Processing (ICNLP), Xi'an, China, 2022, pp. 426-433, doi: 10.1109/ICNLP55136.2022.00079.

[9] Y. Li, J. Yang, T. Fei and Y. Xie, "Few-shot Classification of Radar Equipment Fault Based on TF-IDF Feature Date Augmentation and BERT," 2021 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE), Chongqing, China, 2021, pp. 444-448, doi: 10.1109/MLISE54096.2021.00093.

[10] S. E. Taher and M. Shamsfard, "Adversarial Weakly Supervised Domain Adaptation for Few Shot Sentiment Analysis," 2021 7th International Conference on Web Research (ICWR), Tehran, Iran, 2021, pp. 119-124, doi: 10.1109/ICWR51868.2021.9443023.