12-2023

AUTOMATED MEDICAL NOTES LABELLING AND CLASSIFICATION USING MACHINE LEARNING

Akhil Prabhakar Thota
California State University - San Bernardino Search For An Author Using: Last Name, First Name, Email, or Institution

AUTOMATED MEDICAL NOTES LABELLING AND CLASSIFICATION USING MACHINE LEARNING

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Akhil Prabhakar Thota

December 2023

AUTOMATED MEDICAL NOTES LABELLING AND CLASSIFICATION USING MACHINE LEARNING

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Akhil Prabhakar Thota

December 2023

Approved by:

Dr. Yan Zhang, Advisor, School of Computer Science and Engineering

Dr. Yunfei Hou, Committee Member

Dr. Jennifer Jin, Committee Member

ABSTRACT

The amount of data generated in medical records, especially in a modern context, is growing significantly. As the amount of data grows, it is very useful to classify the data into relevant classes for further interventions. Different methods that are not automated are very time-consuming and require manual effort have been tried for this before.

Recently deep learning has been used for this task but due to the complexity of the dataset, specifically due to inter-class similarities in the dataset and specific terminology having different meanings in medical contexts has caused significant problems in having a definitive approach to medical notes classification. In this study, the different recent improvements to NLP borne from Transformer networks have been implemented on a significant medical notes dataset to classify them into 5 different classes, specifically neoplasms, digestive system diseases, nervous system diseases, cardiovascular diseases and general pathological conditions.

To achieve this, specifically, the best model proposed is the BERT pre-trained on a large corpus and fine-tuned on this medical notes dataset. A robust 5-fold cross-validation is performed when these results are generated to utilize the entire dataset for finding optimal search parameters and the test dataset is used to report the results. The results on the optimized BERT model outperform the results in the paper that introduced this dataset and a baseline linear model.

In addition to the final accuracy, individual class-wise metrics were also calculated and reported showing that general pathological conditions have significant overlap with other diseases and thus are harder to classify. This was visualized using t-SNE of the embeddings of the CLS token in the BERT model.

In conclusion, these results present a novel and effective method for medical notes classification using a transformer-based BERT model. The proposed model outperforms the current state-of-the-art results published on this dataset. There are still significant challenges requiring domain knowledge to be used in modelling to improve efficiency, but the proposed model achieves high class-wise accuracy metrics to automate classification tasks in the first pass after which further models can be used with higher confidence.

ACKNOWLEDGEMENTS

I would like to thank Dr. Yan Zhang, my proposed advisor, for her mentorship and guidance throughout my project. I appreciate the opportunity to experiment on this topic as it is what I'd like to continue my research work in.

I am also thankful to my committee members, Dr. Jennifer Jin and Dr. Yunfei Hou for agreeing to be part of my committee. I hope they find the results to be of interest and I look forward to collaborating with them in future contexts. I would also like to express my appreciation to my university for offering a conducive learning environment and a wide range of courses that have enabled me to be able to work independently on this topic and also to be able to read the current state-of-the-art research papers.

Last, but certainly not least, I want to extend my thanks to my family and friends for their unending support, encouragement, and belief in my endeavours. Their faith has been a driving force behind my achievements, and I am truly fortunate to have such a strong support system.

I am grateful to each, and every individual mentioned above for their contributions to my academic and personal growth.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Recording notes comprising all the relevant information of a patient is an important tool to preserve medical data. Health professionals must invest a significant amount of time collecting the data and this process involves a lot of manual intervention in organizing the accumulated records. By categorizing medical notes into predefined classes, such as diagnostic categories or levels of urgency, healthcare professionals can assist in prioritizing interventions and streamlining theirworkflow. Before the adoption of machine learning and modelling-based techniques,this effort was largely reliant on manual sorting and keyword-based searches. This was inherently very labor intensive, error-prone and lacked the scalability required tohandle the large volumes of data generated in modern medical conditions [1]. Therewere rule-based systems which had nested conditions and regular expressions to scan through the texts, but these approaches were rigid and required constant updating to keep up with evolving medical terminology and techniques [2].

The limitations of these pre-ML approaches created the need for more advanced and automated solutions. Initially, shallow machine learning methods like Naive Bayes or SVM were applied but fell short in capturing the more intricate details of medical text [3]. More recently, in the last few years, CNNs

have gained a lot more prominence and have been successfully applied to this specific task [4].

Challenges

In the realm of text classification specifically applied to medical notes, thereare several challenges specific to classifying medical notes. A few of these challenges are described below:

- Data Imbalance: Some terminologies are very specific to patients and thus are underrepresented in the dataset due to their rarity. So, the obvious problem of data imbalance between classes is more pronounced in this case [5].
- Terminological Variability: Medical notes often contain a plethora of synonyms, abbreviations, and jargon. These terms are not cleanly separable or identified inone individual class which again increases the challenge of modelling this data [6].
- Contextual Ambiguity: The meaning of certain terms in medical notes can changebased on context. For instance, the term "cold" could refer to a viral infection or alow temperature, demanding models to have a deep contextual understanding [7].
- Lack of Training Data: Due to the sensitive nature of this content, there are not alot of publicly annotated datasets available. However, obtaining

such annotationsis challenging due to the need for expert knowledge, making large-scale labelleddatasets scarce

- Inter-institutional Variability: Notes from different healthcare institutions can have varying formats and levels of detail. This lack of standardization poses challengesin training universally applicable models [8].

The medical notes classification task has received some attention in recent times to translate the modern NLP model achievements to the biomedical field to do this specific task [4].

Aim of This Project

In the realm of text classification for medical notes, various methodologies have been presented. This project introduces an optimized, transformer-based approach. By utilizing the capabilities of the transformer architecture, more specifically the BERT model was finetuned for medical note classifications. The primary objective is not only to classify the data but also to show the details of each classification. Through detailed analysis, the precision of each model is validated on individual tasks and its overall performance. Furthermore, by visualizing the embeddings from the model's last layer, further insights into the model's data representation are explored. This visualization unveils potential challenges inherent in the dataset and the solutions model offers in comparison to other techniques. In conclusion, this method has demonstrated superior performance when compared to other published approaches for this dataset.

CHAPTER TWO

RELATED WORK

This chapter will cover the approaches which have been tried specifically in the classification of medical records data.

Traditional Machine Learning Approaches

Rule-Based Approaches

One of the earliest techniques employed for this task was rule-based systems where expert-curated rules were used to identify and classify medical concepts. These rules generally leveraged syntactic patterns of the language and semantic patterns related to the meaning of words and phrases to identify and classify different medical concepts in each hypothesis. Friedman [9] developed a natural language processing system specifically for clinical radiology reports. Their system used a combination of syntactic and semantic techniques to extract and structure information. In the syntactic case, the proposed system identified language structures and patterns to identify important information while on the semantic side, the system used the meaning of words and phrases and used those especially relevant to clinical importance to extract and identify relevant information from the data.

While this was one of the first relatively successful methods applied to these kinds of datasets, there were several immediate problems such as

scalability with manual curation of the rules, maintenance overhead of terminologies that evolve and the rules being too specific to the training dataset leading to lower generalized performance. This approach also relies heavily on the expertise of the domain experts curating the specific rules and misinterpretation in this case leads to lower performance.

BoW and TF-IDF

Liu et al. [10] proposed an approach to classify clinical free-text radiology reports using machine learning techniques. In their methodology, the authors used two prominent text representation techniques: Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). The BoW is a simple yet effective model which represents texts based on the frequency of each word without considering ordinal position or order. On the other hand, the TF-IDF method weighs the importance of terms in documents by not only their local frequency but also their distribution in the entire dataset thus reducing the weight of commonly occurring terms that might not be so informative. This approach in general was found to be quite favourable compared to traditional Naive Bayes and SVM models.

Early Deep Learning Approaches

RNNs and LSTM

Jagannatha et al. [11] explored the use of bidirectional Recurrent Neural Networks (RNNs) for medical event detection in Electronic Health Records

(EHR). This network was primarily targeting the sequential and temporal nature of clinical narratives.

CNNs

Gehrmann et al. [12] published a comparison between deep learning models, particularly CNNs, and traditional concept extraction methods for patient phenotyping and demonstrated the capability of these networks.

Modern Deep Learning Approaches

Transformer-Based Models

With the recent popularity of transformer-based architectures, Lee et.al [13] introduced BioBERT, a variant of the BERT model pre- trained on a large-scale biomedical corpus. The model was fine-tuned on specific biomedical text mining tasks, showing significant improvements over existing state- of-the-art models.

Domain-Specific Pre-trained Models

Alsentzer et al.[14] presented clinical embeddings from BERT, which was trained on a massive number of clinical notes. This work demonstrated the advantages of domain-specific embeddings, which capture the nuances and terminologies unique to the clinical domain, leading to enhanced performance in downstream tasks.

CHAPTER THREE

DATA PREPARATION

In the research, we made use of the health-related notes dataset found in the collection shared by Schopf et al.[15]. This dataset encompasses 14438 training entries, accompanied by a separate testing set of 2888 entries. The entries are organized into five specific health categories: Tumours, Diseases of the digestive tract, Disorders of the nervous system, Heart-related conditions, and General health abnormalities. This organization assists in categorizing the data and provides a perspective on the variety of health issues covered in the dataset.

The distribution of the data, detailed in Table 1, underscores the equilibrium between the categories, which could impact the efficacy of the models. Graphical depictions of this distribution can be seen in Figures 1 and 2.

By examining the length of the summaries in both the training and evaluation datasets, we gained an understanding of the specifics of each entry. Visual results of this examination are displayed in Figures 3 and 4.

Additionally, Figures 3-7 show the terms that appear most often in the dataset, giving a glimpse into commonly occurring topics. While words like "individual" are prevalent in all categories, they alone don't offer substantial differentiation. However, pairing these general terms with other unique terms specific to each category proves essential for data categorization.

Table 1. Distribution of Data Across Classes

Class Name	# Training	# Test	Total
Neoplasms	2530	633	3163
Digestive System Diseases	1195	299	1494
Nervous System Diseases	1540	385	1925
Cardiovascular Diseases	2441	610	3051
General Pathological Conditions	3844	961	4805
Total	11550	2888	14438

Figure 1. Comparison of Categories.

Figure 2. Class Distribution in Test Dataset

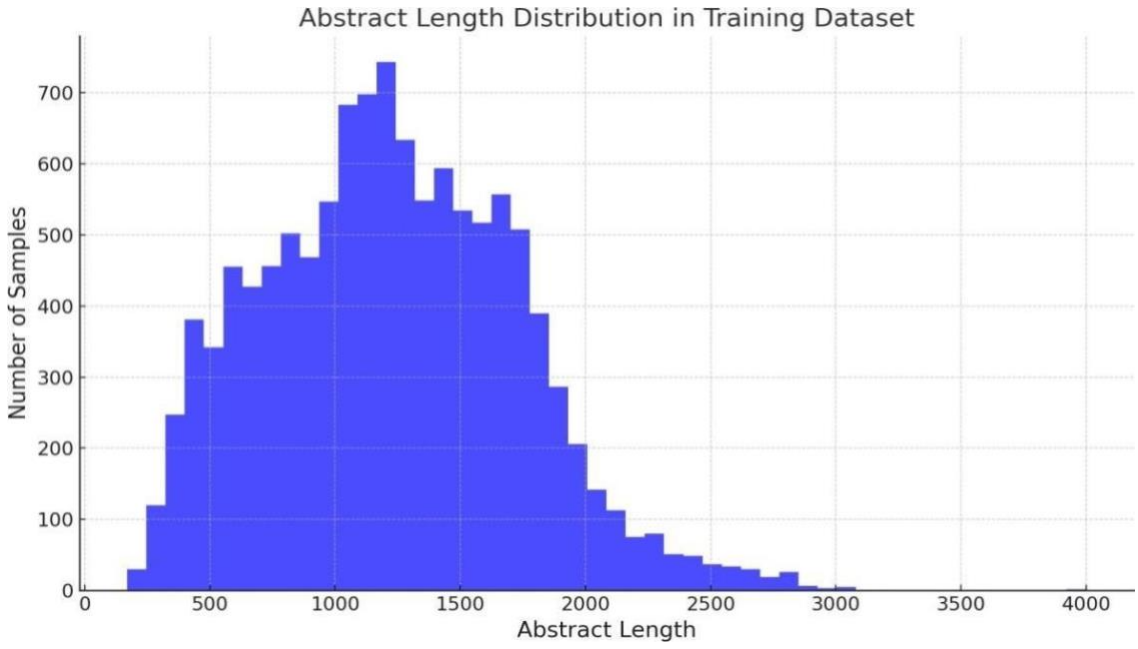

Figure 3. Abstract Length in Training Dataset

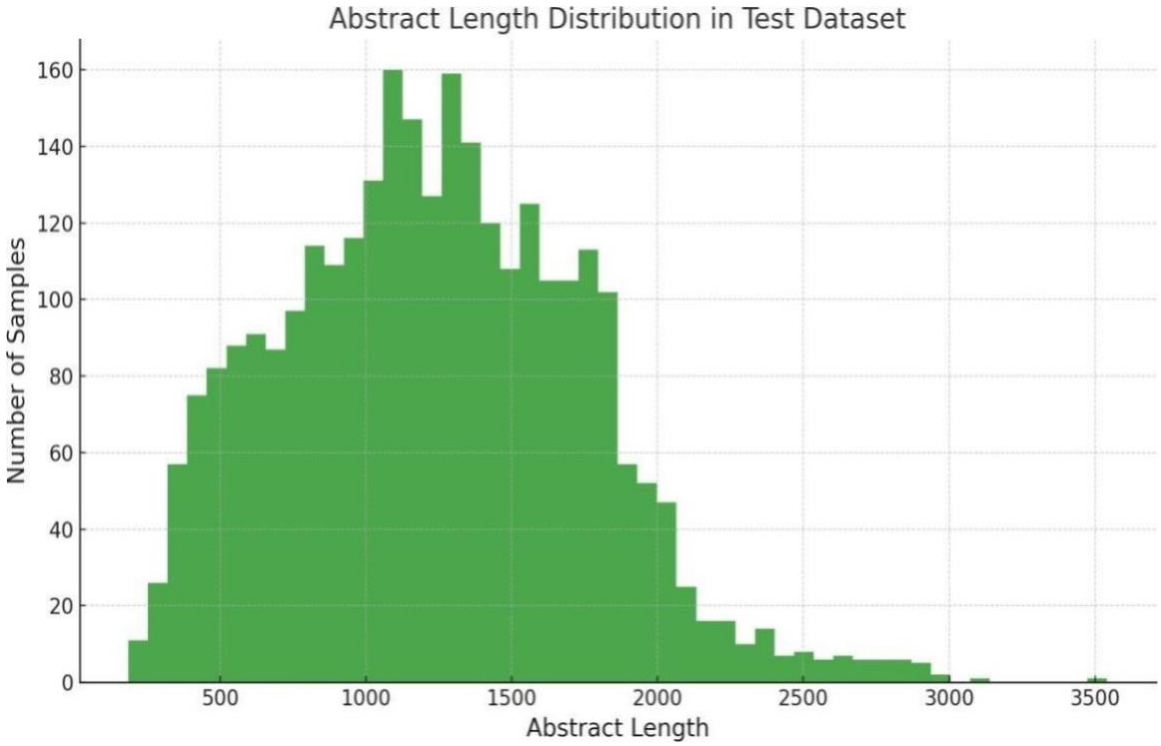

Figure 4. Abstract Distribution in Test Dataset

Data Format and Organization

The dataset provided by Schopf et al. [15] is available on GitHub [18] as 3 CSV files titled “medical_tc_train.csv” and “medical_tc_test.csv”. They contain records with labels called “medical_abstract” which is the medical note and "condition_label” which is the classes enumerated from 1-5 where the names of the classes are given in “medical_tc_labels.csv”. These files are loaded by using the “pandas” library into memory for further processing before being used by the model in a cross-validation setup.

Preprocessing

As this dataset is used in the pipeline, before that the data is preprocessed toimprove model performance using the following strategies.

Text Cleaning

- Convert all text to lowercase to ensure uniformity and reduce dimensionality.
- Strip the text of any punctuation marks, as they may not be informative for models like logistic regression.

Tokenization

Split the abstracts into individual words or tokens. This is essential for models that operate on token-level input, such as RNNs, CNNs, and Transformers.

Stop Words Removal

Given the technical nature of medical abstracts, there needs to be strong caution in removing stop words. Standard stop-word lists might be not applicable since certain common words might carry importance in a medical context.

Lemmatization

Convert words to their base or dictionary form. For instance, "running" becomes "run". This can help in reducing the dimensionality and capturing semantic meaning.

Word Embeddings

The network architectures contain an embedding layer and pre-trained word embeddings like Word2Vec, GloVe, or FastText. These embeddingscan provide dense vector representations for words and potentially improve model performance. For Transformers, embeddings will be learned as part of the model.

Other Publicly Available Datasets

While the dataset by Schopf et al. [15] was used in this project, several otherpublicly available datasets in the medical domain could have been considered.

These datasets offer diverse perspectives on health-related topics and are oftenused for different machine-learning tasks such as classification, regression, andclustering. The methods provided in this project can be further validated across those datasets.

- MIMIC-III Clinical Database: A rich dataset containing de-identified health dataassociated with over forty thousand patients who stayed in critical care units ofthe Beth Israel Deaconess Medical Center between 2001 and 2012. It includesinformation like demographics, vital signs, laboratory tests, and more [19].
- Medical Information Mart for Intensive Care (MIMIC-IV): An update to the MIMIC-III dataset, providing a more extensive collection of de-identified medical data\footnote [20].

Figure 5. Word Clouds for Neoplasms

Figure 6. Word Clouds for Digestive System Diseases

Figure 7. Word Clouds for Nervous System Diseases

Figure 8. Word Clouds for Cardiovascular Diseases

Figure 9. Word Clouds for General Pathological Conditions

CHAPTER FOUR

METHODOLOGY

This chapter will outline the network architectures implemented for the task and their optimized hyperparameters. Further research improvements to transformer-based architecture are also discussed.

Baseline

Before delving into the complexities of transformer-based models, it is vital to establish a solid baseline to benchmark their performance. For this baseline, a well-established simple yet powerful linear model i.e., Logistic Regression is used, to classify medical notes.

The hyperparameters of the Logistic Regression model were optimized usinga GridSearch approach. The hyperparameters under consideration were:

- Regularization strength C : [0.001, 0.01, 0.1, 1, 10, 100]
- Penalty: ["l1", "l2"]
- Solver: ["newton-cg", "lbfgs", "liblinear", "sag", "saga"]

The best model, as determined by the GridSearch, was found to have the hyperparameters: C =1, penalty as "l1", and solver as "liblinear".

This will serve as a baseline against which the performance of subsequent transformer models will be compared.

Transformer

Transformers, introduced by [16], have revolutionized the field of NLP due to their unique architecture and capability to capture contextual information [16]. In this project, this transformer architecture is leveraged for the classification of medical notes.

The network architecture of the transformer can be seen in Figure 8. It consists of the following elements:

- Input Embeddings: Tokens from the medical notes are initially converted into continuous vector representations or embeddings. These embeddings are further enhanced with positional encodings to incorporate the order of the words.
- Multi-Head Self-Attention Mechanism: This allows the model to focus on different words with varying degrees of attention. It captures the inter-word dependencies without regard to their distances in the input text.
- Feed-Forward Neural Networks: Each attention output is passed through a feed- forward neural network, the same one for each position.
- Residual Connections: Around each sub-layer (i.e., attention and feed-forward), there's a residual connection which helps in avoiding the vanishing gradient problem, enabling the model to be trained effectively.
- Normalization and Layer Norm: The output from each sub-layer is normalized, ensuring that the activations don't reach extremely high or low values, facilitating faster and more stable training.

- Stacked Layers: Several such attention and feed-forward layers are stacked, allowing the model to capture complex patterns and relationships.
- Output Layer: The final layer is typically a fully connected layer that projects the representation into the desired number of classes, in this case, the different categories of medical notes.

For the task of classifying medical notes into five distinct classes using the transformer model, a specific set of hyperparameters was chosen to optimize performance. These hyperparameters were selected based on the literature review and optimizing the score on cross cross-validated dataset.

Model Architecture

- Number of Layers: A total of 12 encoder layers were employed in the transformer. Each layer comprises a multi-head self-attention mechanism followed by feed-forward neural networks.
- Number of Attention Heads: Each encoder layer utilizes 8 attention heads, allowing the model to focus on different parts of the input simultaneously.

Training Parameters

- Batch Size: A batch size of 32 was used.
- Learning Rate: An initial learning rate of 0.0005 was chosen, leveraging the Adam optimizer with a warm-up and decay strategy.
- Dropout: To prevent overfitting and enhance generalization, a dropout rate of 0.1 was applied to various components of the network.

Regularization and Others

- Layer Normalization: Layer normalization was applied before each sub-layer, stabilizing the activations and enhancing training speed.
- Positional Encoding: Sinusoidal positional encodings were added to the input embeddings to maintain word order information.

In the process of fine-tuning the Transformer for the medical notes classification, attention visualization was employed to gauge the areas within the notes the model focused on. This method, based on the attention weights, provides insights into the model's decision-making process. It was observed that the model placed significant attention on specific medical terminologies and patterns, indicative of the underlying diseases. Moreover, gradient-based saliency maps were generated to further understand the importance of different tokens in the decision-making process.

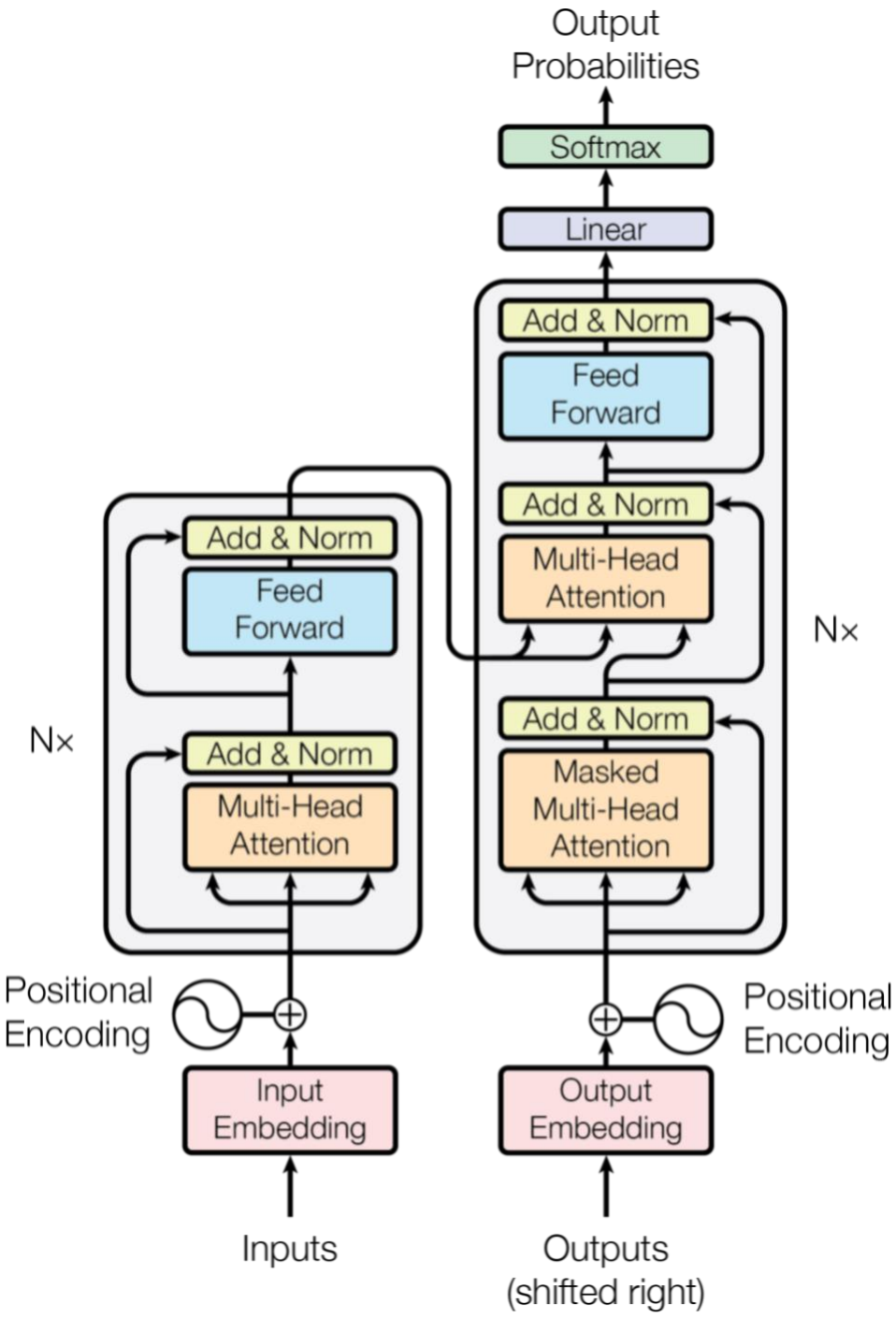

Figure 10. Transformer Network Architecture

BERT

BERT was another model applied for the same task. This network, introduced by Devlin et al. [17], has become a very popular approach to NLP due to its pre- training on large corpus followed by fine-tuning on specific tasks. Its bidirectional nature, which allows it to capture context from both the left and right sides of a token simultaneously, makes it particularly suited for intricate tasks like medical notes classification. The architecture of BERT, as illustrated in Figure 9, is built upon the transformer model. The primary difference lies in how BERT is pre-trained using two tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

Input Representation

- BERT combines token, segment, and positional embeddings to represent input data.

Bidirectional Context

- Unlike traditional left-to-right or right-to-left models, BERT is trained to predict tokens by considering context from both directions, thus termed "bidirectional".

Pre-Training Tasks

- MLM: Random tokens in a sentence are masked (hidden), and BERT is trained to predict them based solely on their context.
- NSP: BERT is trained to predict if a sentence naturally follows a given sentence, aiding in understanding relationships between sentences.

Upon pre-training on large generic corpora, BERT can be fine-tuned for specific tasks, including classification. For the medical notes classification task, the last hidden state of the [CLS] token, a special token used in BERT's input representation, is extracted and fed into a fully connected layer. This layer's output is then used to classify the notes into one of the five desired categories.

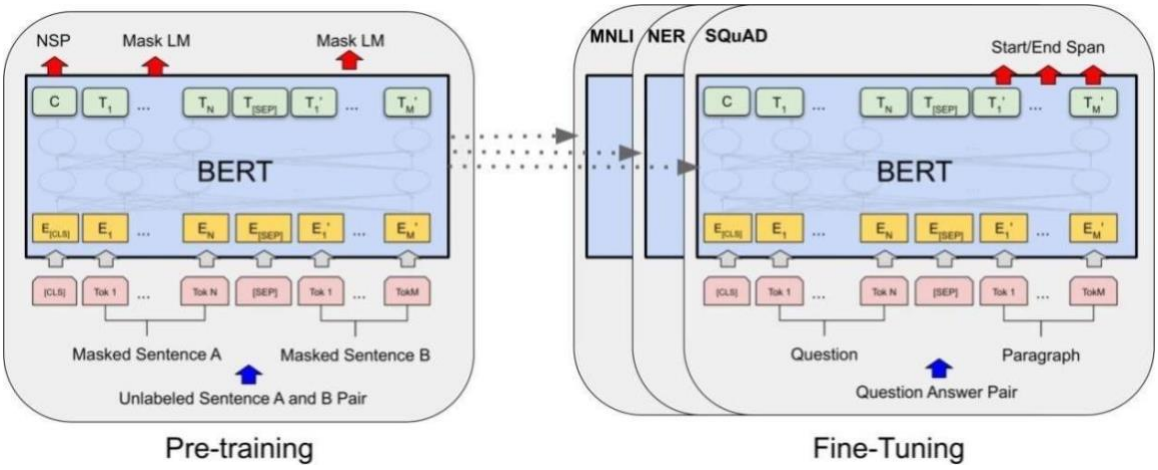

Figure 11. Overall Pre-Training and Fine-Tuning Procedures for BERT

For the task of classifying medical notes into five distinct classes using the BERT model, the following hyperparameters are used after extensive experimentation.

BERT Model Configuration

- Model Variant: “bert-base-uncased".
- Input Sequence Length: 512 tokens. This includes both the content and any special tokens, such as [CLS] and [SEP].

- Number of Output Classes: 5 (corresponding to the five conditions).

Tokenizer Configuration

- Tokenizer: BERT tokenizer (base, uncased variant).
- Maximum Token Length: 512 tokens. Each input sequence is limited to 512 tokens to strike a balance between computational efficiency and information retention.
- Truncation and Padding: Enabled, to ensure consistent input lengths.

Training Configuration

- Batch Size: 16
- Learning Rate: $3 \times 10−5
- Optimizer: AdamW
- Loss Function: Cross-Entropy Loss

Cross-Validation

In the experiments that were conducted, a 5-fold cross-validation strategy was adopted to validate the performance of the model on the medical notes dataset, as illustrated in Figure 10. Cross-validation is a widely used technique in machine learning for assessing how a model generalizes to independent datasets. Specifically, 5-fold cross-validation involves splitting the dataset into 5 equal-sized subsets or 'folds'. The training and evaluation process is repeated five times, with each fold acting as the validation set exactly once, and the remaining four folds collectively are used as the training set.

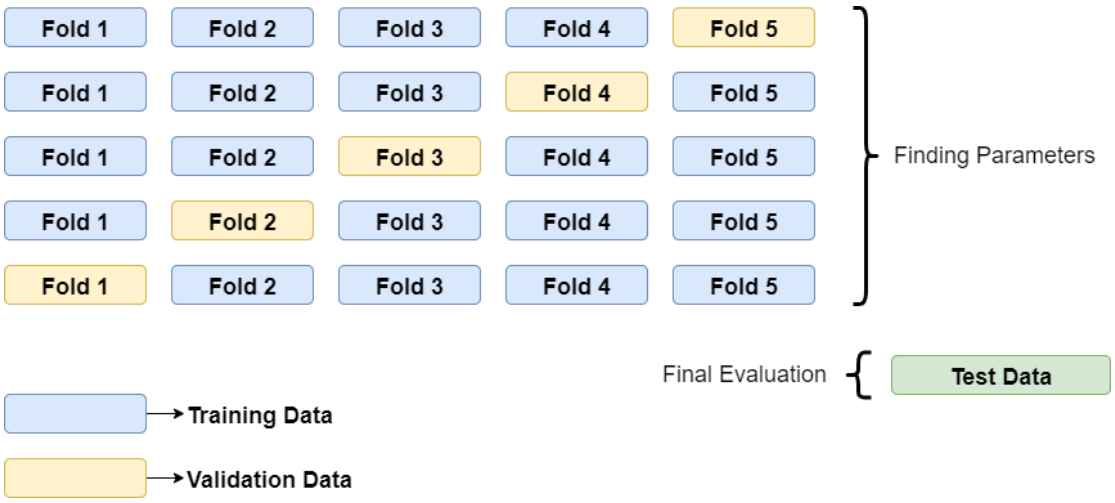

Figure 12. 5-Fold Cross Validation Illustration

The provided illustration in Figure 10 delineates the iterative process of the 5-fold cross-validation:

- In the first iteration, Fold 1 acts as the validation set while Fold 2, Fold 3, Fold 4, and Fold 5 together form the training set.
- In the subsequent iteration, Fold 2 is utilized as the validation set and the other folds constitute the training data.
- This pattern continues iteratively until each fold has been used as the validation data exactly once.

Finally, after the relevant model parameters are chosen the performance is evaluated on the test set.

Memory and Hardware Requirements

Deep learning, especially when dealing with architectures like Transformers and BERT, requires significant computational resources. The hardware and software setup for this project were adjusted to ensure optimal performance without compromising the efficiency of experiments. Here's a closer look at the setup and the versions of the software that's used.

The different models listed in this project were trained and inference on the following hardware:

- RAM: 16GB
- Memory: Checkpoints size 2GB
- GPU: 12GB NVIDIA GTX 1070
- CPU: Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz

Correspondingly, the software utilized for the networks is as follows:

- torch: 2.0.1
- pandas: 2.0.3
- numpy: 1.24.4
- matplotlib: 3.7.3

Training Quirks and Nuances

While training deep neural networks, especially more complex architectures such as Transformers and BERT, certain specific problems need to

be addressed that have a significant impact on model performance. Several important such factors will be discussed below:

Model Initialization

Model initialization is very important in certain classes of models. The way a model is initialized can significantly influence its convergence and final performance. While pre-trained models like BERT which have been implemented here have the advantage of leveraging weights from prior training on large data, the transformer models need very specific initialization. Techniques like He and Xavier initialization were tried to ensure the vanishing or exploding gradients problem doesn't occur.

Learning Rate Scheduling

Here while a constant learning rate is used for simpler tasks, in this project for medical notes classification sometimes a more dynamic learning rate proved more beneficial. Certain strategies like learning rate warm-up and decay were explored to see if the model converged faster. Generally, there were no significant drastic updates initially due to this and the decay phase subsequently reduced the learning rate over time which allowed the model to get a more refined solution.

Tokenization Challenges

As discussed in the previous preprocessing section, tokenization was very important for models like Transformers and BERT. However, in this dataset, standard tokenizers sometimes didn't perform properly and couldn't handle the

dataset due to jargon and abbreviations. Making sure these tokenization processes contained these nuances while not losing semantic meaning was manually observed.

Positional Encoding Limitations

As discussed in Chapter 4, the positional encodings are the way the transformer model imbibes context into the text for the model to process where the words are about other words. However, since medical notes were sometimes significantly long, the sinusoidal positional encoding could sometimes become less effective, especially if sequence lengths exceeded the typical lengths seen during training. Alternate ways were explored but in general, the dataset was able to maintain generalization with the sinusoidal encoding. While it was successful in this dataset, this is an important detail to recognize.

Batch Size and Memory Constraints

Newer architectures sometimes have many attention heads and layers. This can be quite memory-intensive. Balancing the batch size and memory and GPU available was important to train the model efficiently and to avoid out-of-memory errors. Gradient accumulation could be considered in cases where this is not feasible but in this case for a reasonable batch size between 16 and 32, the model was trained successfully.

CHAPTER FIVE

EXPERIMENTAL RESULTS

Evaluation Metrics

This chapter will outline the metrics used to compare the methods implemented for the medical notes classification task

Accuracy

Accuracy is the ratio of correct predictions (both true positives and true negatives) among the total number of records. It is most useful when classes are balanced. However, in cases where there's a class imbalance, accuracy can be misleading. The equation for accuracy [21] is:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total number of samples}}$$

Precision

To help with that problem with accuracy as a metric, another important value is Precision which evaluates the correctness of positive predictions. A high precision indicates a low false positive rate. It is particularly important when the cost of false positives is high. The equation for precision [21] is:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall (Sensitivity)

Recall on the other hand determines if the classifier can identify all relevant instances. It's crucial in scenarios where missing a positive sample (false negative) is more costly than misclassifying a negative sample (false positive). The equation [21] for this is:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1-Score

The F1-score is the harmonic mean of precision and recall. It provides a balance between the two metrics, especially when there's an uneven class distribution. A higher F1 score indicates better-balanced precision and recall. The equation [21] is:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

AUC-ROC

The ROC curve plots the true positive rate against the false positive rate for different threshold values. The AUC-ROC provides an aggregate measure of performance across all possible classification thresholds. A model with perfect discriminatory power has an AUC-ROC of 1, while a model with no discriminatory power (akin to random guessing) has an AUC-ROC of 0.5.

Result Analysis

The methods outlined in Chapter 4 were employed on the datasets mentioned earlier for the classification of medical notes. This chapter provides a detailed summary of the results obtained from various transformer-based models in comparison with the baseline. A comprehensive review of the different experiments that were undertaken is also presented. The primary emphasis is on illustrating how the proposed BERT model stacks up against the baseline and current state-of-the-art models.

Before discussing in-depth results, it is essential to note the number of trainable parameters in the different models:

Table 2. Number of Trainable Parameters for Various Models

Model	**# of Parameters**
Transformer	103,799,045
BERT	109,486,085

From Table 2, it is evident that the number of trainable parameters for the standard Transformer classifier and the BERT model are quite similar. Given that the baseline is a straightforward linear model, the number of parameters in these models is considerably higher in comparison. The performance metrics for the implemented models are provided in Table 3. The optimized BERT model, as observed, achieves the highest accuracy, making it the recommended model over the baseline logistic regression and the comparative Transformer model. A

detailed discussion of the results, with a primary focus on the proposed BERT model, is provided in the subsequent sections.

Table 3. Performance Metrics for BERT and Baseline Models on Medical Note Classification

Model	Accuracy	F1 Score	Precision	Recall
Baseline	0.563	0.542	0.554	0.551
Transformer	0.591	0.587	0.599	0.587
BERT	0.629	0.621	0.627	0.629

Table 4. Class-wise Performance Metrics for Medical Note Classification for the BERT Model

Model	Accuracy	F1 Score	Precision	Recall
Neoplasms	0.8726	0.6864	0.7709	0.7262
Digestive System Diseases	0.9124	0.6438	0.3445	0.4488
Nervous System Diseases	0.8975	0.6395	0.5299	0.5795
Cardiovascular Diseases	0.8778	0.6772	0.8049	0.7356
General Pathological Conditions	0.6208	0.5447	0.5515	0.5481

From the provided classification metrics and class-specific results, the final proposed model is observed to outperform the baseline substantially. Confusion matrices for both models are depicted in Figures 13 and 14. Additionally, the AUC-ROC curve is illustrated in Figure 15. Delving into the

class-wise accuracy metrics, it becomes evident that General Pathological Conditions pose a greater challenge in classification compared to other diseases, which achieve a minimumaccuracy of 85%. The respective AUC-ROC values for the classes stand at 0.93 for neoplasms, 0.91 for digestive system diseases, 0.90 for nervous system diseases, 0.93 for cardiovascular diseases, and 0.71 for general pathological conditions. These statistics underscore the inherent challenge of distinguishing general conditions from other disease categories.

Comparison with State-of-the-Art

Referring to the findings of Schopf et al. [15], who utilized the same dataset, the reported F1 scores are as follows:

Table 5. Comparison with Results Published on the Same Dataset

Model	**F1 Score**
Lbl2TransformerVec (SimCSE)	39.60
Word2Vec	25.00
SimCSE	34.94
DeBERTa [15]	57.28
Final Model (Finetuned BERT)	62.10

From the table above, among the results published for this dataset, it is observed that the model finetuned with MLM and NSP exhibits the highest performance. The inherent challenge in classifying general pathological conditions will be further scrutinized by visualizing the network's last layer into distinct clusters.

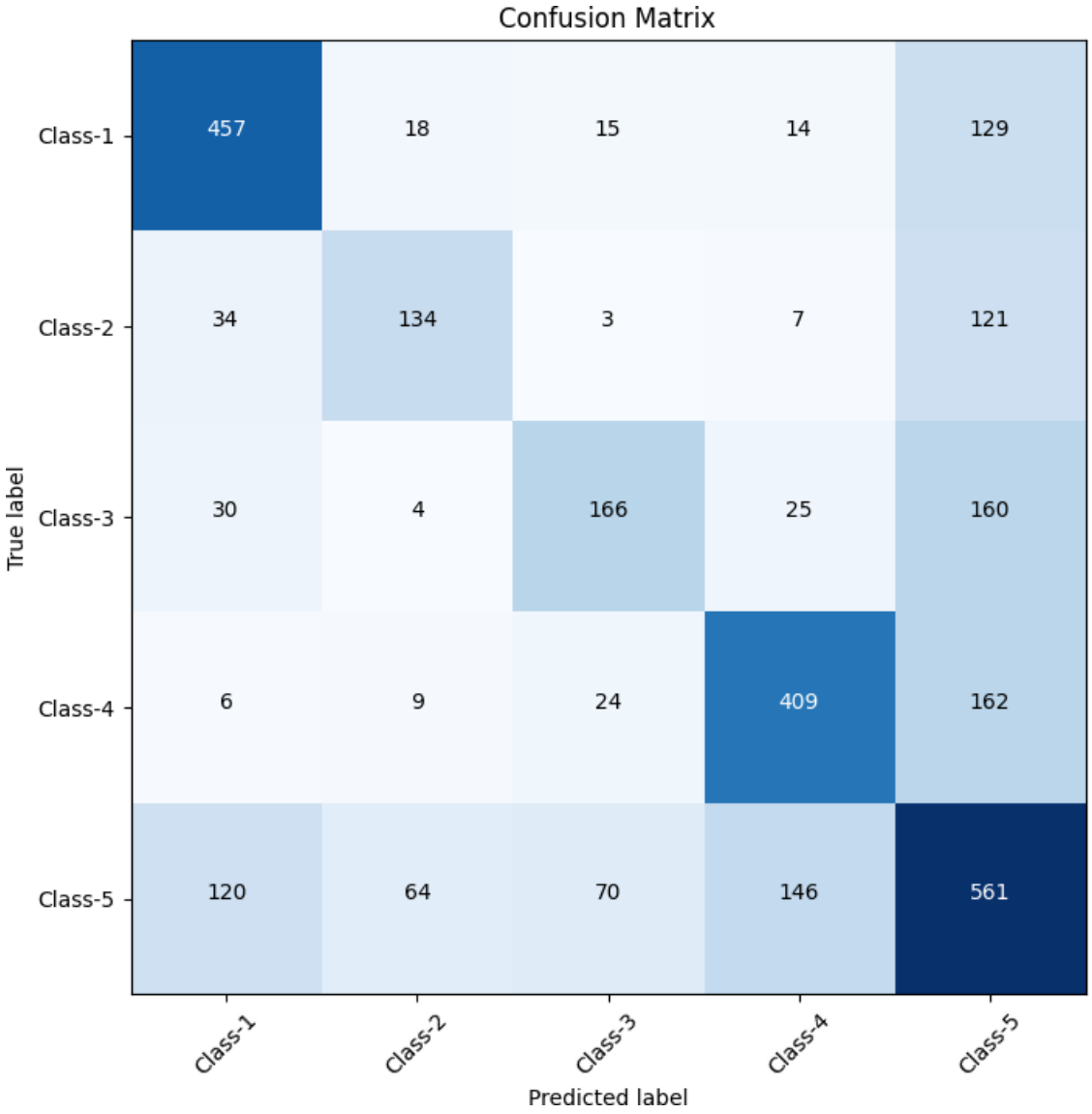

Figure 13. Confusion Matrix of Baseline Linear Model

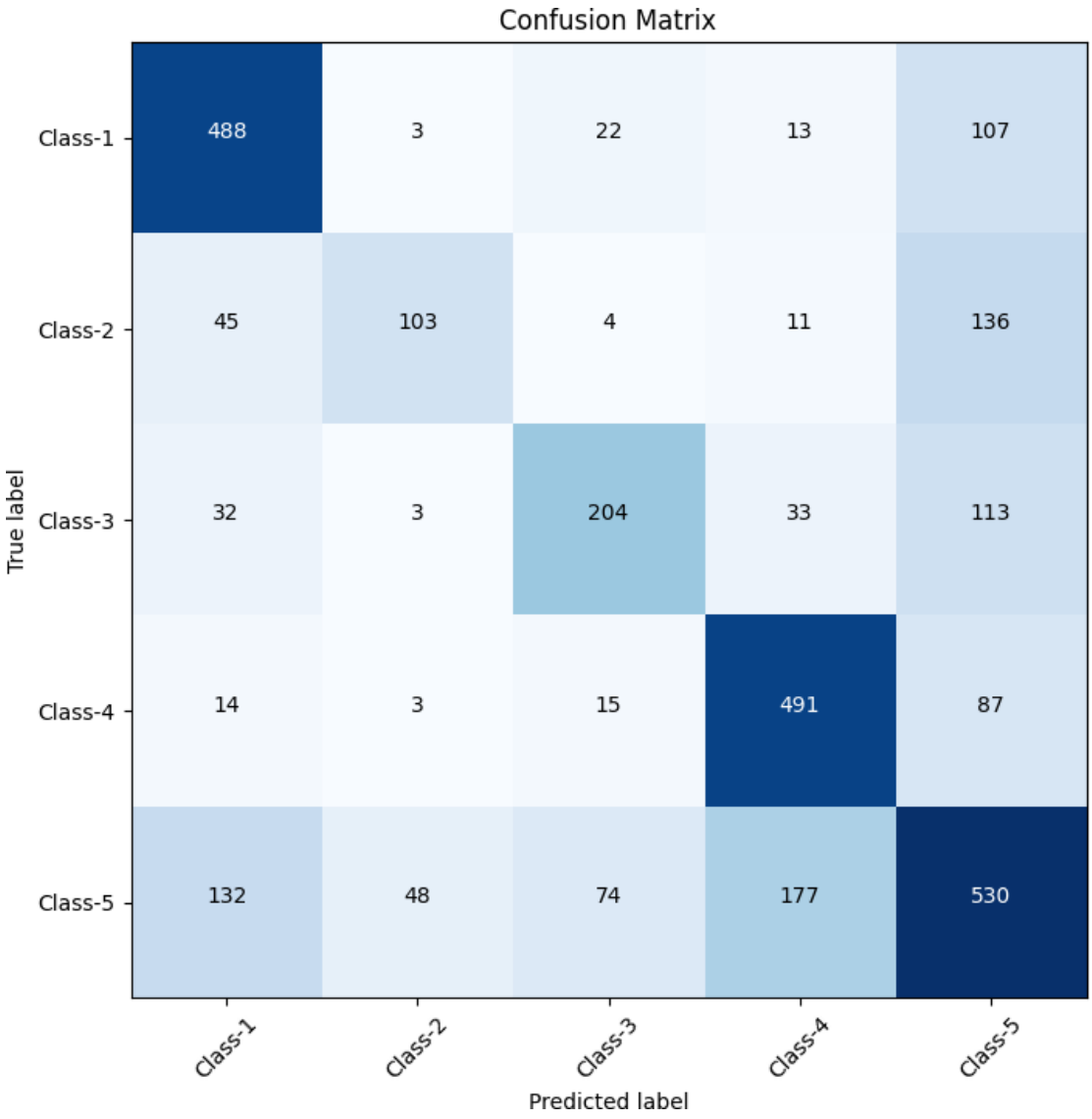

Figure 14. Confusion Matrix of BERT Model

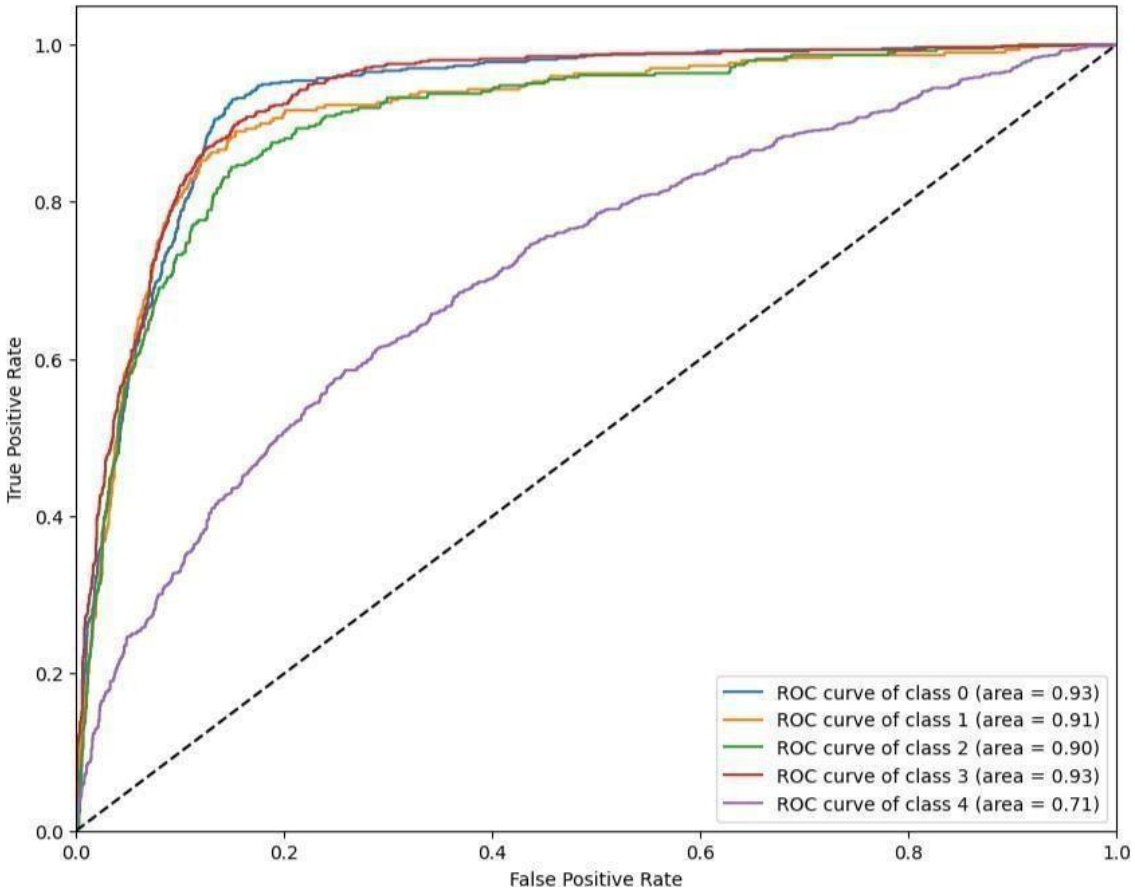

Figure 15. ROC Curve and AUC-ROC Values Class-wise of BERT Model

The t-SNE visualizations presented in Figure 14 offer an enhanced perspective of the embeddings derived from the BERT model. This visualization captures the embeddings of the Classification (CLS) token in the last layer. As can be seen, this visualization confirms the findings in Table 4. While it is not a linear model, there are clear separable clusters for every class except general pathological conditions which might explain the low accuracy.

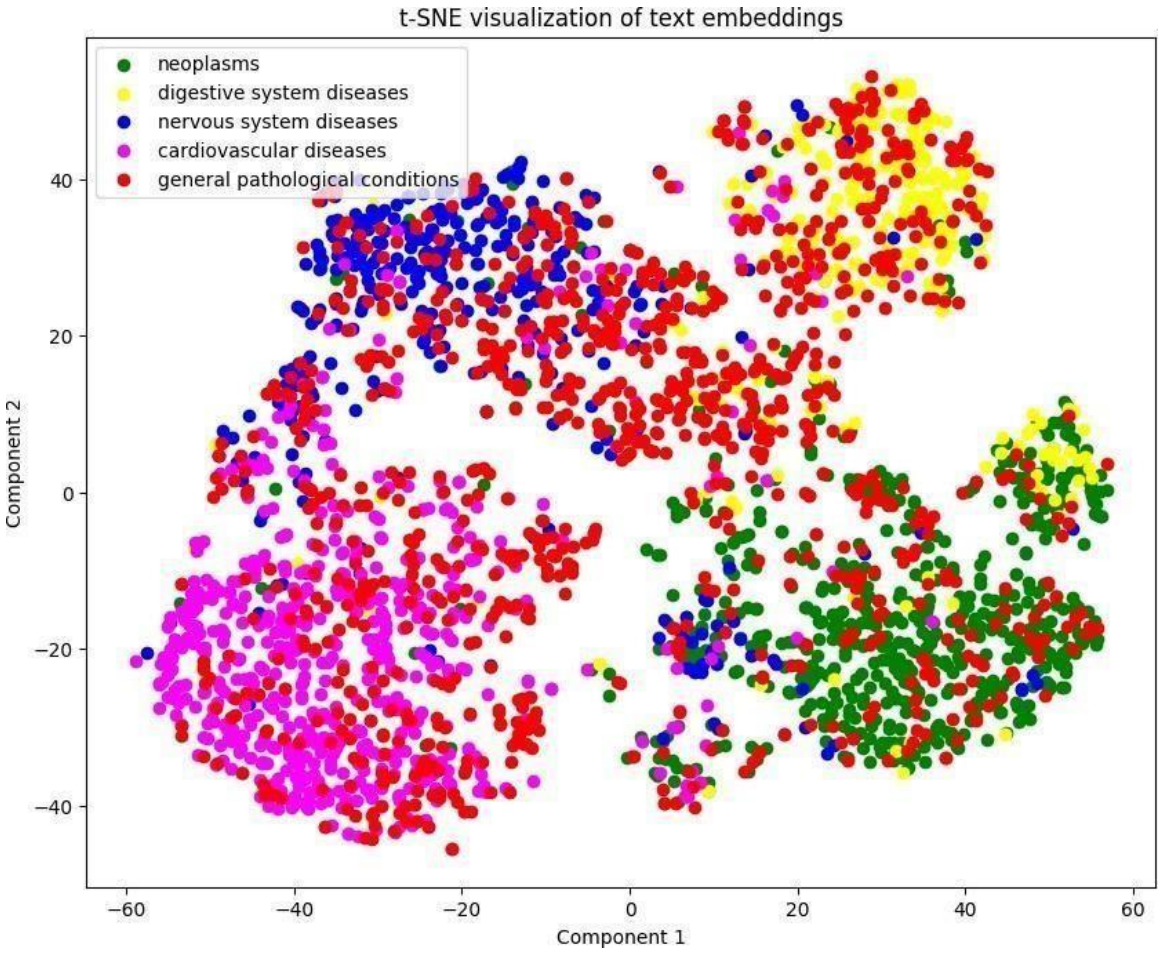

Figure 16. t-SNE Visualization of Text Embeddings

Insights And Project Contributions

This section will cover the insights from models trained in this project along with the novelties compared to the original models that are used. The primary conclusions that can be drawn from the results listed in Tables 3 and 4 show that the final model can classify the notes with accuracy at least 87% accuracy except in the case of general pathological conditions. These results are further validated in the visualizations in Figures 13 and 14 where the confusion

matrix shows that in both the cases of false positive and false negative the models are having a harder time concluding which texts classify as more general and also the low AUROC score in Figure 15. This can be thought of as the primary reason being that general conditions utilize words very common in all different diseases and they are a collection of different classes themselves grouped into one larger cluster. This part is visualized in Figure 16 concluding that in the case that the general pathological conditions are further mapped to different classes it will have a better chance. This is also shown in the word cloud in Figure 9 having very few unique words further explaining this difficulty. Regarding the novel contributions it was shown that utilizing He and Xavier initialization helped ensure that the vanishing or exploding gradient problem doesn't occur. Alongside, to improve the performance sinusoidal performance encoding was more effective than standard positional encoding. Along with this, the BERT model was finetuned to use a maximum of 512 tokens with truncation which showed to improve both performance and generalization. For the transformer itself, the network architecture was changed to use 12 layers as it was able to capture enough complex patterns with an increase not providing further gains in accuracy and needing more compute as well. In general, the inference is that individual models with more context on general pathological conditions as layered models could improve performance in that context.

CHAPTER SIX

CONCLUSION

In this project, medical notes classification was addressed using advanced deep-learning techniques. Within the domain of medical informatics, accurate categorization of medical notes stands as a cornerstone for administrative efficiency and the improvement of patient care.

The BERT model, a popular implementation of the Transformer architecture, was adopted and its performance was benchmarked against other leading models. Metrics including accuracy, precision, recall, and F1-Score were employed for comprehensive evaluation. These quantitative benchmarks furnished insights into the model's capabilities and its comparative positioning.

Though the model exhibited admirable performance in various class-wise accuracies, consistently surpassing the 85% threshold, challenges persist in the classification of general pathological conditions. There is potential for integrating more domain-specific knowledge into the modeling process. For example, fine-tuning specific parameters or leveraging domain-specific embeddings could enhance outcomes. The evident advantages of pretraining were highlighted; however, initializing natural language processing models with medical literature may further hone the model's capabilities.

Additionally, the exploration of advanced attention mechanisms offers promise. Considering the structured nature of medical notes, models adept at capturing interrelationships between medical terms could prove invaluable.

This project thus underscores the potential and effectiveness of a fine-tuned BERT model in the realm of medical notes classification. By further iterating on these methodologies and refining the approach, there is the possibility that the classification process can achieve greater precision, thereby minimizing manual oversight and elevating the accuracy of categorization. More specifically, recent improvements in the GPT architecture can be transferred to this dataset but this network generally requires much more data than what is used in this project.

Future Work

The medical note classification using transformer-based models, especially the BERT variant, has shown promising results. However, several options remain unexplored and could be pursued to further improve the performance or provide more insights into the problem. Here are some suggestions for future work:

- Data Augmentation: Finding annotated datasets in the medical field is quite challenging. Data augmentation techniques, such as back translation or synonym replacement, could be employed to artificially expand the dataset and potentially improve the model's robustness.

- Class-specific Models: Given the difficulty in classifying general pathological conditions, as observed from the results, it might be beneficial to build class- specific models, where each model is specifically fine-tuned for a particular category of diseases. This would be especially helpful to further sub-categorize the general pathological conditions.
- Attention Visualization: While the t-SNE visualizations of embeddings are very useful, further work can also include visualizing the attention weights as they progress through the network.
- Few-shot and Zero-shot Learning: For rare diseases or conditions where only, a few labeled examples are available, few-shot or even zero-shot learning techniques could be explored to leverage the knowledge gained from other tasks.

APPENDIX

MODEL CODE

Checkpoints for the various trained models can be provided upon request. This includes network code, checkpoints, training, and evaluation files.

The final proposed model code can be seen below:

```
# Imports
import re
import string
import pandas as pd import torch
from tqdm import tqdm
from sklearn.model_selection import train_test_split
from transformers import BertTokenizer,
BertForSequenceClassification, AdamW
from torch.nn import CrossEntropyLoss
from sklearn.metrics import accuracy_score, f1_score,
precision_score, recall_score, confusion_matrix, roc_curve,
auc
# NLTK for text preprocessing
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
# Download necessary NLTK resources
nltk.download('wordnet')
```

```
nltk.download('stopwords')
# 1. Text Preprocessing
def clean_and_lemmatize(texts, remove_stopwords=True):
     lemmatizer = WordNetLemmatizer() cleaned_texts = []
     for text in texts:
          # Convert to lowercase
          text = text.lower()
          # Remove punctuation
          text = re.sub(f"[{string.punctuation}]", "",text)
          # Tokenize by splitting the text into words
          tokens = text.split()
          # Optional: Remove stop words
          if remove_stopwords:
               tokens = [word for word in tokens if word
          not in stopwords.words('english')]
               # Lemmatize each token
               lemmatized_tokens
          =[lemmatizer.lemmatize(token) for token in
          tokens]
               # Reconstruct the text from tokens
               cleaned_text = ' '.join(lemmatized_tokens)
               cleaned_texts.append(cleaned_text)
          return cleaned_texts
```

```
# 2. Data Loading
def load_data(file_path):
     return pd.read_csv(file_path)
# 3. Tokenization
def tokenize_data(data, tokenizer):
     return tokenizer(data, padding=True,
truncation=True, max_length=512, return_tensors='pt')
# 4. Data Preparation
def prepare_data(tokenized_data, labels):
     train_inputs, val_inputs, train_labels,
val_labels = train_test_split(
          tokenized_data['input_ids'], labels,
     test_size=0.2, stratify=labels, random_state=42
     )
          train_attention_masks, val_attention_masks,
     _, _ = train_test_split(
               tokenized_data['attention_mask'],
          labels, test_size=0.2, stratify=labels,
          random_state=42
     )
          return train_inputs, val_inputs,
     train_attention_masks, val_attention_masks,
     train_labels, val_labels
```

```
     # 5. Model Training and Validation
def train_and_evaluate(train_data, val_data, model,
epochs, batch_size, learning_rate):
          # Device setup
          device = torch.device('cuda' if
     torch.cuda.is_available() else 'cpu')
           model.to(device)
          # Optimizer and loss function
     optimizer = AdamW(model.parameters(),
     lr=learning_rate) loss_fn = CrossEntropyLoss()
          train_inputs, val_inputs,
     train_attention_masks, val_attention_masks,
     train_labels, val_labels = train_data
          # Lists to store metrics
          train_losses = []
          val_losses = []
          val_accuracies = []
          for epoch in range(epochs):
               model.train()
               total_loss = 0
               train_loop = tqdm(range(0,
          len(train_inputs), batch_size),
          total=len(train_inputs) // batch_size)
```

```
            for i in train_loop:
                batch_input =
            train_inputs[i:i+batch_size].to(device)
                batch_masks =
            train_attention_masks[i:i+batch_size].t
            o(device)
                batch_labels =
            torch.tensor(train_labels[i:i+batch_siz
            e]).to(device)
            optimizer.zero_grad()
            outputs = model(batch_input,
            attention_mask=batch_masks,
            labels=batch_labels)
            loss = outputs.loss
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
            train_loop.set_description(f"Epoch
            {epoch+1}")
            train_loop.set_postfix(loss=loss.item()
            )
        avg_train_loss = total_loss /
(len(train_inputs) // batch_size)
```

```
train_losses.append(avg_train_loss)
print(f"Epoch {epoch+1}, Training loss:
{avg_train_loss}")

# Validation step
model.eval()
total_val_loss = 0
correct = 0
val_loop = tqdm(range(0, len(val_inputs),
batch_size),
total=len(val_inputs) // batch_size)
with torch.no_grad():
     for i in val_loop:
          batch_input =
     val_inputs[i:i+batch_size].to(device)
          batch_masks =
     val_attention_masks[i:i+batch_size].to(
     device)
          batch_labels =
     torch.tensor(val_labels[i:i+batch_size]
     ).to(device)
```

```
                outputs = model(batch_input,
            attention_mask=batch_masks,
            labels=batch_labels)
                loss = outputs.loss
                total_val_loss += loss.item()
                preds =
                torch.argmax(outputs.logits,
                dim=1)
                correct += (preds ==
                batch_labels).sum().item()
                val_loop.set_description(f"Validat
                ing (Epoch{epoch+1})")

                 val_loop.set_postfix(val_loss=loss
            .item())
             avg_val_loss = total_val_loss /
        (len(val_inputs) // batch_size)
             val_accuracy = correct /
             len(val_labels)
             val_losses.append(avg_val_loss)
             val_accuracies.append(val_accuracy)
             print(f"Epoch {epoch+1}, Validation
             loss: {avg_val_loss},
```

```
        Validation accuracy: {val_accuracy}")
            return train_losses, val_losses,
    val_accuracies
        def test(test_data, model, batch_size = 32):
            # Device setup
            device = torch.device('cuda' if
          torch.cuda.is_available() else
'cpu')
          model.to(device)
    # Ensure model is in evaluation mode
    model.eval()
    # Get predictions on test data
    all_test_preds = []
    with torch.no_grad():
         for i in range(0, len(test_data),
    batch_size):
              batch_input =
         test_data['input_ids'][i:i+batch_size].to(de
         vice)
              batch_masks =
    test_data['attention_mask'][i:i+batch_size].to(de
    vice)
```

```
                outputs = model(batch_input,
            attention_mask=batch_masks)
                preds = torch.argmax(outputs.logits,
            dim=1)
            all_test_preds.extend(preds.cpu().numpy())
        # Convert the labels to 0-indexed values
        test_labels = test_data['condition_label'].values
        - 1
        # Calculate metrics
        accuracy = accuracy_score(test_labels,
        all_test_preds)
        f1 = f1_score(test_labels, all_test_preds,
        average='weighted')
        precision = precision_score(test_labels,
        all_test_preds,
        average='weighted')
        recall = recall_score(test_labels,
        all_test_preds, average='weighted')
        print(f"Accuracy: {accuracy}") print(f"F1 Score:
        {f1}") print(f"Precision: {precision}")
        print(f"Recall: {recall}")
# Main execution
def main():
```

```
    # Load the dataset
    data = load_data('./medical_tc_train.csv')
    # Initialize the tokenizer
    tokenizer = BertTokenizer.from_pretrained('bert-
    base-uncased')
    # Preprocess and tokenize the medical abstracts
    cleaned_abstracts =
clean_and_lemmatize(data['medical_abstract'],
remove_stopwords=True)
    tokenized_data = tokenize_data(cleaned_abstracts,
tokenizer)
    # Labels starting from 0
    labels = data['condition_label'].values - 1
    # Prepare data for training and validation
    train_data = prepare_data(tokenized_data, labels)
    # Initialize the model
    model =
    BertForSequenceClassification.from_pretrained('be
    rt- base-uncased', num_labels=5)
    # Training parameters
    learning_rate = 2e-5
    batch_size = 12
    epochs = 50
```

```
# Train and evaluate the model
train_losses, val_losses, val_accuracies =
train_and_evaluate(train_data, None, model,
epochs, batch_size, learning_rate)
# Save the trained model model_save_path =
'/mnt/bert.pt'
model.save_pretrained(model_save_path)
# Output the results print("Training complete!")
test_data = load_data('./medical_tc_test.csv')
cleaned_test_abstracts =
clean_and_lemmatize(test_data['medical_abstract']
, remove_stopwords=False)
tokenized_test_data =
tokenize_data(cleaned_test_abstracts,
tokenizer)
test(tokenized_test_data, model)
# Run the main function
if name == " main ":
     main()
```

REFERENCES

[1] Alsentzer, E., Murphy, J. R., Boag, W., Weng, W. H., Jin, D., Naumann, T., & McDermott, M. (2019). Publicly Available Clinical BERT Embeddings. *arXiv preprint arXiv:1904.03323.*

[2] Cimino, J. J. (1998). Desiderata for Controlled Medical Vocabularies in the Twenty-First Century. *Methods of Information in Medicine*, 37(04/05), 394-403.

[3] Davis, J., & Goadrich, M. (2006, June). The Relationship between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 233-240).

[4] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

[5] Friedman, C., Alderson, P. O., Austin, J. H., Cimino, J. J., & Johnson, S. B. (1994). A General Natural-Language Text Processor for Clinical Radiology. *Journal of the American Medical Informatics Association*, 1(2), 161-174.

[6] Gehrmann, S., Dernoncourt, F., Li, Y., Carlson, E. T., Wu, J. T., Welt, J., ... & Celi, L. A. (2018). Comparing Deep Learning and Concept Extraction-Based Methods for Patient Phenotyping from Clinical Narratives. *PLOS ONE*, 13(2), e0192360.

[7] Hripcsak, G., & Rothschild, A. S. (2005). Agreement, the f-measure, and Reliability in Information Retrieval. *Journal of the American Medical Informatics Association*, 12(3), 296-298.

[8] Hughes, M., Li, I., Kotoulas, S., & Suzumura, T. (2017). Medical Text Classification using Convolutional Neural Networks. In Informatics for Health: *Connected Citizen-Led Wellness and Population Health* (pp. 246-250). IOS Press.

[9] Jagannatha, A. N., & Yu, H. (2016, June). Bidirectional RNN for Medical Event Detection in Electronic Health Records. in *Proceedings of the Conference*. *Association for Computational Linguistics. North American Chapter. Meeting* (Vol. 2016, p. 473). NIH Public Access.

[10] Joachims, T. (1998, April). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *European Conference on Machine Learning* (pp. 137-142). Berlin, Heidelberg: Springer Berlin Heidelberg.

[11] Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L. W. H., Feng, M., Ghassemi, M., ... & Mark, R. G. (2016). MIMIC-III, a Freely Accessible Critical Care Database. *Scientific Data*, 3(1), 1-9. https://physionet.org/content/mimiciii/1.4

[12] Johnson, A., Bulgarelli, L., Pollard, T., Horng, S., Celi, L. A., & Mark, R. (2020). Mimic-iv. *PhysioNet*. Available online at: https://physionet.org/content/mimiciv/2.2/

[13] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2020). BioBERT: A Pre-Trained Biomedical Language Representation Model for Biomedical Text Mining. *Bioinformatics*, 36(4), 1234-1240.

[14] Liu, H., Lussier, Y. A., & Friedman, C. (2001). A Study of Abbreviations in the UMLS. In *Proceedings of the AMIA Symposium* (p. 393). American Medical Informatics Association.

[15] Meystre, S. M., Savova, G. K., Kipper-Schuler, K. C., & Hurdle, J. F. (2008). Extracting Information from Textual Documents in the Electronic Health Record: A Review of Recent Research. *Yearbook of Medical Informatics*, 17(01), 128-144.

[16] Perotte, A., Pivovarov, R., Natarajan, K., Weiskopf, N., Wood, F., & Elhadad, N. (2014). Diagnosis Code Assignment: Models And Evaluation Metrics. Journal of the *American Medical Informatics Association*, 21(2), 231-237.

[17] Schopf, T., Braun, D., & Matthes, F. (2022). Evaluating Unsupervised Text Classification: Zero-Shot and Similarity-Based Approaches. *arxiv preprint arXiv:2211.16285.*

[18] Schopf, T., Braun, D., & Matthes, F. (2022). Evaluating Unsupervised Text Classification: Zero-Shot and Similarity-Based Approaches. *arXiv preprint arXiv:2211.16285*. Dataset GitHub Link: https://github.com/sebischair/Medical-Abstracts-TC-Corpus

[19] Stubbs, A., & Uzuner, Ö. (2015). Annotating Longitudinal Clinical Narratives for De-Identification: The 2014 i2b2/UTHealth Corpus. *Journal of Biomedical Informatics,* 58, S20-S29.

[20] Uzuner, Ö., South, B. R., Shen, S., & DuVall, S. L. (2011). 2010 i2b2/VA Challenge on Concepts, Assertions, and Relations in Clinical Text. *Journal of the American Medical Informatics Association*, 18(5), 552-556.

[21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances In Neural Information Processing Systems*, 30.