Natural Language Tasks Natural Language Processing for Clinical Notes

Midhun Shyam
22058122

A report submitted for
INFO7017 Postgraduate Project B
in partial fulfillment of the requirements for the degree of
Master of Data Science

Supervisor: **Dr. Rosalind Wang**
Co-supervisor: **Dr. Jim Basilakis**

School of Computer, Data and Mathematical Sciences,
Western Sydney University

Spring, 2024

Abstract

The research project explores the application of Natural Language Processing (NLP) techniques on healthcare data, primarily utilising transformer architectures. The research also investigate enhancing keyword search through the integration of probability distribution of words in the corpora using Latent Dirichlet Allocation (LDA) method.

The project is carried out with the sole objective of improving the efficiency of the New South Wales (NSW) Healthcare services. The primary objective is building binary classification algorithm to classify kinetic accidents cases in the emergency department using text triage notes.

The key findings of the project include, but are not limited to, the performance evaluation of the Bio Clinical BERT model, which was pre-trained on the Medical Information Mart for Intensive Care (MIMIC) datasets, applied to the NSW Healthcare data. Data mining and data annotation techniques are discussed.

The research project report discusses the successful applications of transformer-based healthcare text classification, along with their identified limitations and the fine-tuning efforts made to address these limitations. The report provides a detailed explanation of the model architecture, training process, fine-tuning techniques, and performance outcomes. The model, which has 110 million parameters, achieved a benchmark performance on classification with an accuracy of 93.75% in the text classification task, utilising NVIDIA GPUs on a High-Performance Computing (HPC) server, with other training and fine tuning details, and recommendations based on findings.

Acknowledgements

I wish to express my profound gratitude and appreciation to
***Dr. Rosalind Wang**, my esteemed supervisor,*
whose unwavering support and guidance have been pivotal
to the realisation of this research project.

*I extend my sincere gratitude to **Dr. Jim Basilakis**, my co-supervisor*
for his invaluable support and guidance throughout this undertaking.

*Furthermore, My sincere gratitude extends to my **professors** & **tutors**,*
*and the **School of Computer, Data and Mathematical Sciences** and the*
Ingham Institute of Applied Medical Research
who supported me with their knowledge and assistance.

*I would like to thank my **parents** and my **partner**, without which none of*
this would have been possible for me.
*I am also thankful to my **friends** and **God**.*

Contents

List of Figures

Nomenclature

`Bio_ClinicalBERT`: This variant of the BERT model is specifically adapted for analysis of biomedical and clinical texts. It plays a crucial role in processing and interpreting clinical notes.

`Fine-Tuning`: This process involves adjusting pre-trained models to specialise them for particular datasets. In this case, the model is fine-tuned using clinical texts that represent kinetic accidents and various clinical conditions, typically focusing on the later layers or solely the classification head to tailor it to specific, domain-related tasks.

`Epochs`: This term refers to the complete cycles through the entire training dataset. The model mentioned is trained over ten epochs to enhance its precision in classifying clinical text.

`F1 Score`: A metric combining precision and recall to evaluate a model's accuracy, crucial for assessing how well the model distinguishes between true and false cases in clinical notes.

`True Positives/Negatives and False Positives/Negatives`: These indicators from the confusion matrix evaluate the classification model's performance by showing the instances where it accurately or inaccurately predicted the classes.

- True Positives (TP): These are the cases where the model correctly predicts the positive class. For example, if the positive class is 'kinetic accident', true positives are the cases where the model correctly identifies the text as describing a kinetic accident.
- True Negatives (TN): These are the cases where the model correctly predicts the negative class. If the negative class is 'non-kinetic accident', true negatives are the cases where the model correctly identifies the text as not describing a kinetic accident.
- False Positives (FP): These occur when the model incorrectly predicts the positive class. That is, the model predicts that a text describes a kinetic accident when it does not.
- False Negatives (FN): These occur when the model incorrectly predicts the negative class. That is, the model predicts that a text does not describe a kinetic accident when it actually does.

`Classification Head`: This part of a neural network is responsible for making predictions specific to the task, often fine-tuned from pre-trained models to better suit particular tasks such as classifying clinical notes.

`Gold-Standard Dataset`: This dataset is considered a benchmark for training and testing machine learning models within a particular field, manually evaluated by surgeon, serving as a crucial reference point for evaluating new models or techniques.

`Layer 11 of Bio_ClinicalBERT`: This specific layer within the Bio_ClinicalBERT model was involved in the fine-tuning process, indicating an extensive adaptation aimed at capturing complex patterns within clinical data more effectively.

`Annotated Dataset`: This refers to clinical texts that have been manually labelled to train or test the machine learning model, essential for verifying the model's capability to accurately identify and classify clinical scenarios.

1. Introduction

Natural language, such as English, is the primary means of communication for humans, while computers operate on binary code, processing data as 0s and 1s. Natural Language Processing (NLP), a branch of computer science, addresses this gap by developing methods for machines to analyse, interpret, and understand human language. Applications of NLP are diverse and widely integrated into everyday technologies, including email platforms, voice assistants, search engines, and translation tools like Google Translate (Vajjala et al., 2020; Kuo, 2023).

Health Informatics merges information science and computer science within healthcare, and is currently experiencing a technological shift driven by Big Data Analytics. This advancement holds immense potential to enhance patient care but also poses significant challenges in managing the extensive data volumes generated in this field (Herland et al., 2014).

Estimates suggest that 80% of healthcare data remains unprocessed. The amount of data in healthcare is expanding by 47% annually, with an average hospital producing about 50 petabytes each year. The integration of Natural Language Processing (NLP) in healthcare offers vast possibilities to improve service quality through techniques like text classification. This NLP method analyses text to assign tags or categories to text based on set criteria, which can, for example, assist healthcare providers in identifying at-risk patients through specific keywords in their records (Vajjala et al., 2020).

New South Wales (NSW) Healthcare services aim to improve the efficiency of patient care through the identification of kinetic accident cases in the emergency department. This project undertakes the goal of classifying patients based on triage notes from the emergency department. The classification algorithm is developed to identify emergency cases associated with kinetic accidents and to facilitate further statistical analysis.

The clinical data is massive and features a large vocabulary associated with varieties such as American or British English, which makes it challenging for NLP applications. This project is a research study on the application of available NLP methods to clinical text processing, aiming to evaluate their efficiency and identify the challenges associated with their implementation.

The integrity and confidentiality of patient data in healthcare are critical, and breaches can lead to severe disruptions. Unauthorised access through cyberattacks can expose sensitive medical records and personal information, endangering patient privacy and disrupting medical services. Stolen data can be used for blackmail, causing psychological and financial harm to patients. Data anonymization or de-identification involves removing patient-identifiable and highly sensitive confidential information from electronic medical records (EMRs), enabling researchers to access clinical records without explicit consent, while ensuring the preservation of patient privacy (Cardinal, 2017).

The research objective is to utilise the potential applications of NLP to enhance the New South Wales (NSW), Australia, healthcare data analysis and the quality of NSW health care services. The study focus on transforming unstructured clinical data into actionable insights, while developing computational algorithms to process the human

language applicable within the context of healthcare.

The algorithms are developed using the Medical Information Mart for Intensive Care (MIMIC) datasets—MIMIC-III and MIMIC-IV—for training and validation. The fine-tuned algorithms are then tested on a NSW Healthcare gold-standard dataset to evaluate their performance.

The classification algorithm is developed using the pre-trained transformer model, Bio ClinicalBERT, from Hugging Face. This model, originally developed by Emily Alsentzer, was trained on the MIMIC-III dataset. A classification head, in the form of a sequential classifier neural network layer, is added to Bio ClinicalBERT for this purpose.

This research utilises a series of NLP techniques to advance towards the primary goal of the study. These steps include using regular expressions to parse text, applying text classification to categorise data, implementing entity recognition to identify specific elements, and utilising Latent Dirichlet Allocation to detect patterns within medical text data.

Latent Dirichlet Allocation (LDA) was utilised as a critical technique for data annotation through topic modeling. LDA's ability to model textual data by identifying latent thematic structures enabled the effective categorisation of large text corpora without prior manual annotation. By assuming that documents are probabilistic mixtures of latent topics, LDA facilitated the extraction and labelling of topics from unstructured data with the integration of regular expression, enhancing the accessibility and interpretability of the information. This method proved instrumental in data annotation, thus providing a solid foundation for training the Bio Clinical BERT transformer model for sequential classification. The methodology for LDA, established by Blei, Ng, and Jordan (Blei et al., 2003), underscores its applicability in automatically discovering and annotating topics within extensive text collections, making it an invaluable tool in this research.

2. Aim and Objectives

The primary aim of this project is to improve the quality of patient care provided in the emergency department by NSW Healthcare. The study will also support further statistical analysis of categorical data to identify potential patterns, such as high-risk zones where accidents occur more frequently, and facilitate root cause analysis, etc.

Objectives:

1. The primary objective of this project is to develop a binary classification algorithm. This algorithm uses triage notes (text data) as input and predicts a label as the output. The true label (1) is assigned to kinetic accidents, such as motor vehicle accidents, bike accidents, etc., while all others are assigned a false label (0).

2. The project aims to explore multi-class classification within the available time frame and extend its objectives to identify and classify based on the specific cause of the accident, such as a bike, e-bike, scooters from other motor vehicles.

3. Research question and hypothesis

Research Question: How effectively can Transformer Neural Network Architectures be applied in healthcare to classify patients from unstructured triage notes using NLP algorithms? Specifically, how accurately can these models categorize patients by emergency types such as kinetic accidents, and minimize misclassification errors? How efficiently can these architectures integrate traditional keyword regular expression algorithms with state-of-the-art transformers?

- Hypothesis: Integrating probability distribution techniques into text mining improves data annotation, essential for training Transformer Neural Network Architectures. This method, supported by the probability distribution of words in annotated corpora, improves the effectiveness of regular expressions and keyword-based methods. This improvement is expected to increase the accuracy of Natural Language Processing (NLP) models in healthcare, allowing for more precise and efficient classification of patients by emergency type, such as kinetic accidents, with fewer misclassification errors compared to traditional searches which could be an integral part of a classification system. The transformers architecture which is the key to large language models, are hypothesised to support effective binomial and multinomial classifications using clinical notes in emergency departments.

4. Literature review

4.1. An overview of Artificial Intelligence (AI)

Artificial Intelligence (AI) encompasses several subfields, including machine learning (ML) and deep learning (DL), each building on the foundations laid by early logic-, heuristics-, and rule-based systems (Vajjala et al., 2020). Machine learning algorithms autonomously perform tasks by learning from large datasets, while deep learning, a subset of ML, utilizes neural network architectures. Natural Language Processing (NLP), another crucial AI domain, translates human language into machine-understandable binary language.

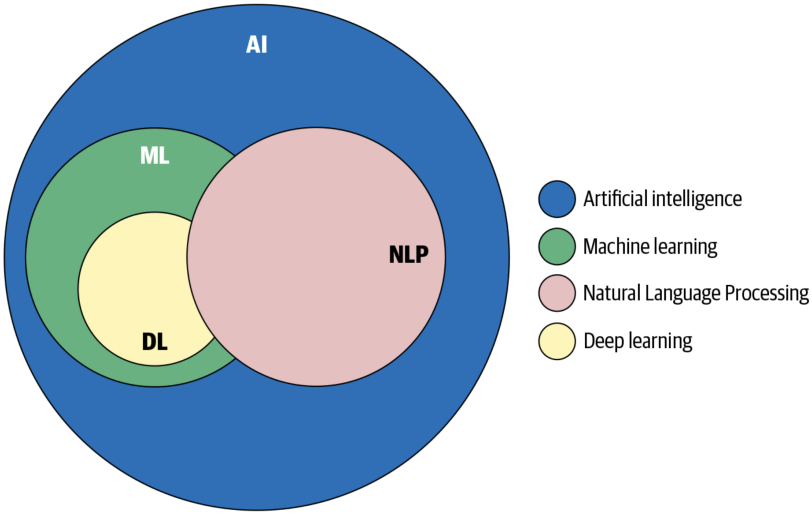

Figure 1: An Overview of AI (Vajjala et al., 2020)

This relationship between AI, ML, DL, and NLP is illustrated in (See Figure 1), highlighting how these areas interconnect and contribute to advancements in understanding and processing human language within the context of AI.

Human language comprises four critical elements—phonemes, morphemes, lexemes, syntax, and context—essential for Natural Language Processing (NLP), as depicted in (See Figure 2)

Phonemes are the smallest sound units crucial for word distinction, such as /b/ in "bat" and /p/ in "pat". Morphemes, the smallest meaning units, include elements like "un-" in "unhappy" and "-s" in "cats". Lexemes represent all inflected forms of a word, with "run," "runs," "ran," and "running" all variations of the lexeme "run". Syntax governs grammatical sentence construction, exemplified by "The cat sat on the mat".

NLP's challenge lies in processing these components amidst the inherent ambiguity and creativity of language, as highlighted by the Winograd Schema Challenge (Levesque

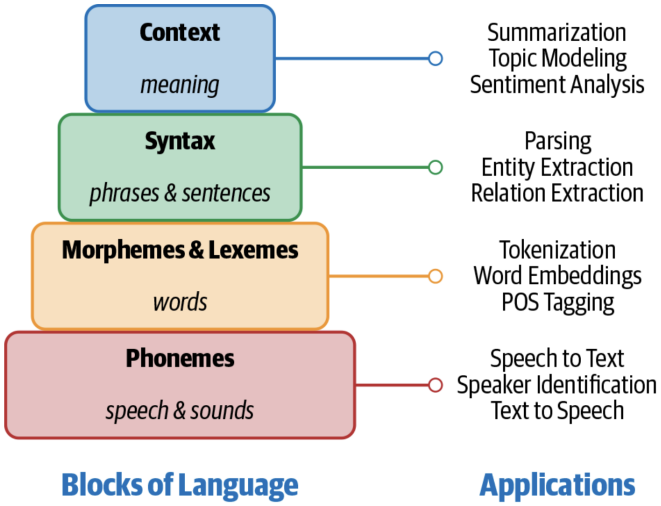

Figure 2: Building blocks of language (Vajjala et al., 2020)

et al., 2012). This complexity is further complicated by the need to encode and interpret the implicit knowledge in human communication, illustrating the sophisticated tasks involved in NLP (Vajjala et al., 2020).

Words must be converted into a format suitable for machine learning algorithms, and in transformers, this involves several key steps. Initially, raw text is tokenised using a model-specific tokenizer that breaks the text into words or subwords. These tokens are then converted to unique integers, or token IDs, which correspond to the model's vocabulary. Additional inputs like attention masks are also created to guide the model's focus on relevant tokens. These token IDs are processed through an embedding layer within the transformer, converting each ID into a dense vector that encapsulates semantic and contextual meanings. This numeric transformation allows the neural network to effectively compute and learn from the textual data.

The embedding layers within transformers are designed not only to convert text to numerical form but also to preserve and interpret the nuanced relationships between different linguistic elements. These embeddings are crucial for capturing deeper linguistic patterns that traditional methods might overlook, allowing the model to handle complex tasks like sentiment analysis and contextual relevance with greater accuracy. This advanced capability significantly enhances the model's ability to derive meaningful insights from vast amounts of text data.

4.2. Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generative statistical model that categorizes documents into topics that best represent the set of words in them. It assumes documents are generated from a mixture of topics, where each topic is characterized by a distribution over all possible words.

LDA utilizes variational inference methods and an Expectation-Maximization (EM) algorithm for parameter estimation, showing advantages over simpler models such as unigram mixtures and probabilistic latent semantic indexing. Its primary purpose is to provide concise descriptions of collection items, improving processing efficiency while maintaining the essential statistical relationships required for tasks like classification and summarization (Blei et al., 2003).

4.2.1. Mathematical Description

The generative process for each document w in a corpus D unfolds as follows:

1. Choose $N \sim \text{Poisson}(\xi)$.
2. Choose $\theta \sim \text{Dir}(\alpha)$, where θ is the topic distribution, and α is the parameter of the Dirichlet prior on the per-document topic distributions.
3. For each of the N words w_n:
   a) Choose a topic $z_n \sim \text{Multinomial}(\theta)$.
   b) Choose a word w_n from $p(w_n|z_n, \beta)$, where β is the parameter of the Dirichlet prior on the per-topic word distributions.

The joint distribution of the topic mixture θ, topics z, and words w given α and β is:

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta)p(w_n|z_n, \beta)$$

LDA is capable of uncovering hidden thematic structures in large text corpora, facilitating deeper insights into the underlying topics in the data. LDA employs variational inference methods and an Expectation-Maximization (EM) algorithm for parameter estimation, proving superior to simpler models like unigram mixtures and probabilistic latent semantic indexing. The primary aim of LDA is to offer concise item descriptions in a collection, enhancing processing efficiency while preserving the statistical relationships essential for tasks such as classification and summarization (Blei et al., 2003).

4.2.2. Plate Notation

The dependencies among the multiple variables in LDA can be concisely captured using plate notation, which is a standard way to represent the structure of a probabilistic model

with repeating components (See Figure 3). Here, M denotes the number of documents, N the number of words in a document, K the number of topics, and V the vocabulary size. The plate notation for LDA is depicted as follows:

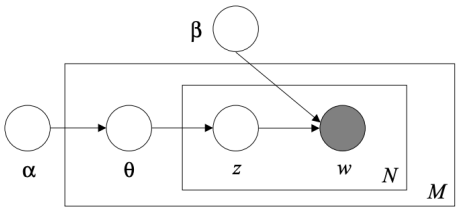

Figure 3: LDA Plate Notation Blei et al. (2003)

- α and β are the hyperparameters for the Dirichlet processes for topics per document and words per topic, respectively.
- θ_d (a vector of length K) represents the topic mixture for document d.
- $z_{d,n}$ is the topic for the n-th word in document d.
- $w_{d,n}$ is the specific word.
- β_k (a vector of length V) represents the word distribution for topic k.

4.2.3. Inference

In practice, the posterior distributions of the hidden variables θ, z, and β are not computationally feasible to compute exactly due to the coupling between θ and β in the marginal likelihood. Various approximation techniques such as Gibbs sampling, Variational Bayes, and Expectation Maximization are used for inference in LDA.

Gibbs sampling is particularly common for its simplicity and effectiveness, where each latent variable is sampled in turn, conditional on the current values of all other latent variables.

4.2.4. Applications

LDA has been successfully applied in various domains including document modeling, information retrieval, and computational biology, demonstrating its versatility and robustness as a tool for unsupervised learning in natural language processing and beyond.

4.3. Transformers

The Transformer model represents a major advancement in natural language processing by introducing a novel architecture that eliminates the need for traditional recurrent or convolutional neural networks. This model relies entirely on self-attention mechanisms, enabling it to capture dependencies between elements in a sequence with greater efficiency and parallelization. The key component, multi-head attention, allows the model to focus on different parts of the sequence simultaneously, providing a more nuanced understanding of context (Vaswani, 2017).

Each layer of the model consists of both self-attention and feed-forward networks, facilitating effective learning of sequence relationships. Positional encodings are integrated to account for the order of tokens, as the self-attention mechanism itself does not inherently consider sequence order. This positional information is essential for accurately interpreting language structure within the model.

In performance, the Transformer achieved state-of-the-art results on machine translation tasks, setting new benchmarks with significantly reduced computational costs compared to earlier models. This architecture has since become foundational in NLP, enabling the development of subsequent models that build upon its efficient handling of large-scale data and long-range dependencies in sequences.

4.3.1. Transformer Architecture

The Transformer architecture, shown in (See Figure 4), follows an encoder-decoder structure, with both components comprising multiple layers of self-attention and feed-forward networks.

- **Encoder**: The encoder stack contains $N = 6$ identical layers, each with:
  - *Multi-Head Self-Attention*: This mechanism allows the model to focus on various parts of the input sequence simultaneously, capturing relationships across different word positions.
  - *Feed-Forward Network (FFN)*: A fully connected network applies non-linear transformations, enhancing the model's representational capacity.

  Residual connections and layer normalization follow each sub-layer, stabilising training. All encoder outputs, including embeddings, are fixed at $d_{\text{model}} = 512$.

- **Decoder**: The decoder stack, also with $N = 6$ identical layers, includes an additional sub-layer:
  - *Masked Multi-Head Self-Attention*: This layer ensures each token only attends to previous tokens, preserving autoregressive properties.
  - *Encoder-Decoder Attention*: This layer attends to encoder outputs, focusing the decoder on relevant input sequence parts.
  - *Feed-Forward Network (FFN)*: Similar to the encoder, this network applies non-linear transformations.

Each sub-layer in the decoder is also followed by residual connections and layer normalization.

- **Positional Encoding**: Since Transformers lack inherent sequence order, positional encodings are added to input embeddings, providing necessary information on token order.

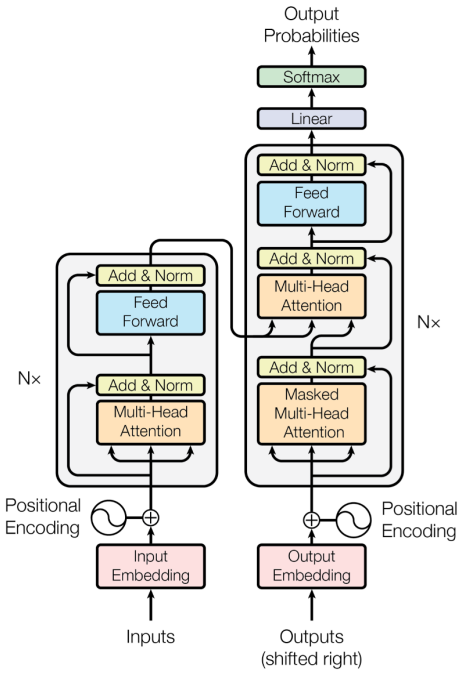

Figure 4: The Transformer architecture (Vaswani, 2017)

The Figure (See Figure 4) illustrates this structure, showing how each component contributes to sequential data processing.

4.3.2. Attention Mechanism

The Transformer utilises *Scaled Dot-Product Attention* and *Multi-Head Attention*:

- *Scaled Dot-Product Attention*: Given queries Q, keys K, and values V, the attention computes:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where d_k scales the dot products, controlling for large values.

- *Multi-Head Attention*: The model projects queries, keys, and values $h = 8$ times with different linear projections, enabling each head to capture diverse information. The output is then:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

This multi-head setup allows the model to attend to multiple representation subspaces, keeping computational costs efficient.

4.3.3. Scaled Dot-Product Attention

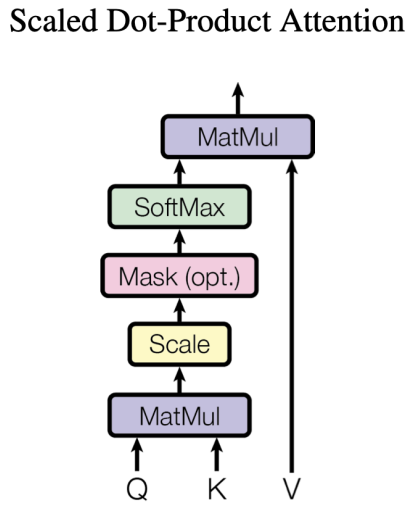

Figure 5: Scaled dot product attention (Vaswani, 2017)

The figure (See Figure 5) depicts *Scaled Dot-Product Attention* function maps a set of queries Q, keys K, and values V to an output. Each output is a weighted sum of values, where the weights depend on the compatibility of each query with its respective key:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

The scaling factor, $\frac{1}{\sqrt{d_k}}$, mitigates the risk of extremely small gradients that occur when the dimensionality of the keys d_k is large, thus enhancing training stability.

4.3.4. Multi-Head Attention

The *Multi-Head Attention* mechanism, illustrated in (See Figure 6), is a cornerstone of the Transformer architecture, designed to enhance the model's ability to process different

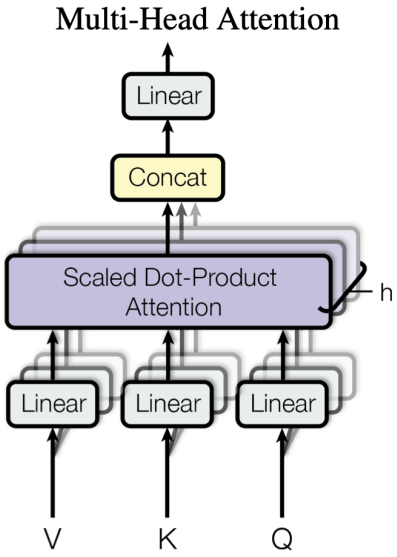

Figure 6: Multi-head attention Vaswani (2017)

subspaces of information simultaneously. Unlike traditional attention mechanisms that focus on a single set of relationships at a time, multi-head attention allows the model to attend to information from different representation subspaces at different positions.

Each head has unique projections for the queries, keys, and values:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

where:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

and $W_i^Q, W_i^K, W_i^V \in R^{d_{\text{model}} \times d_k}$ and $W^O \in R^{h \cdot d_v \times d_{\text{model}}}$ are learned projections. With $h = 8$ heads, each with $d_k = d_v = d_{\text{model}}/h = 64$, the Transformer achieves efficient computation while maintaining dimensional diversity across attention heads.

4.3.5. Applications of Attention in the Model

The Transformer model utilises attention in three main ways:

- *Encoder-Decoder Attention*: This layer allows the decoder to focus on relevant input features by computing attention scores over the encoder's output.
- *Self-Attention in Encoder*: Each encoder layer contains self-attention, allowing each position to attend to other positions in the input sequence.
- *Masked Self-Attention in Decoder*: By masking future positions, the decoder ensures that predictions for a given position depend only on known outputs up to that point.

4.3.6. Position-wise Feed-Forward Networks

Each encoder and decoder layer includes a *Position-wise Feed-Forward Network (FFN)* applied independently at each position:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

where $W_1 \in R^{d_{\text{model}} \times d_{\text{ff}}}$ and $W_2 \in R^{d_{\text{ff}} \times d_{\text{model}}}$. With $d_{\text{model}} = 512$ and $d_{\text{ff}} = 2048$, this layer enhances the model's representational power by adding non-linearity.

4.3.7. Positional Encoding

The Transformer uses *Positional Encodings* to retain sequence order, as it lacks recurrence. The encoding for position *pos* is defined as:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

These sinusoidal functions provide information on the relative and absolute positions of tokens, facilitating the model's ability to process sequence order effectively.

4.3.8. Residual Connections and Layer Normalization

Each sub-layer within the Transformer model's encoder and decoder stacks employs *Residual Connections* and *Layer Normalization*. The residual connection ensures stable gradient flow by allowing the original input to bypass each sub-layer, which mitigates issues of vanishing gradients, especially in deep architectures. Formally, for an input x and a sub-layer function $\text{Sublayer}(x)$, the output is computed as:

$$\text{LayerNorm}(x + \text{Sublayer}(x))$$

where LayerNorm represents layer normalization. This structure stabilises training and improves model performance by reducing internal covariate shift, which is essential in deep networks like the Transformer.

4.3.9. Overall Transformer Architecture

The Transformer model comprises an encoder-decoder setup where both the encoder and decoder include multiple identical layers with unique processing tasks:

- The **encoder stack** processes input sequences, producing context-rich representations. The encoder does not rely on future information and applies self-attention and feed-forward networks across each layer.

- The **decoder stack** generates output sequences one token at a time, using masked self-attention to prevent positions from attending to future tokens, preserving the model's autoregressive property.

Each encoder and decoder layer is identical in structure but uses separate learned weights. The multi-head attention, positional encodings, and the feed-forward networks work jointly to enable the model to process large and complex input sequences efficiently.

4.3.10. Efficiency and Scalability of Transformer Architecture

The Transformer's architecture is optimised for parallel processing due to the absence of recurrence. By replacing traditional recurrence with self-attention, the Transformer achieves higher training efficiency on long sequences, as each token can be processed simultaneously. The computational complexity of self-attention is $O(n^2 \cdot d_{\text{model}})$, where n is the sequence length, but the Transformer compensates for this with efficient parallelisation, making it highly scalable on modern hardware.

4.3.11. Application of Transformers in Natural Language Processing (NLP)

The introduction of the Transformer model has led to significant advances in NLP, enabling state-of-the-art performance in tasks such as machine translation, summarization, and text generation. Its capacity to process context across entire sequences simultaneously enhances its ability to capture complex language patterns and dependencies. The model's adaptability has made it foundational for developing large language models, such as BERT and GPT, which further refine and build upon the core Transformer principles for specific language understanding and generation tasks.

4.4. BERT for Text Classification

Bidirectional Encoder Representations from Transformers (BERT) is a groundbreaking model introduced in 2018 that transforms the approach to text classification and several other natural language processing tasks (Lee and Toutanova, 2018). BERT's architecture is based on a multi-layer bidirectional Transformer encoder. By pre-training on a large corpus of text and then fine-tuning on specific tasks, BERT achieves state-of-the-art results across a diverse range of tasks and benchmarks.

4.4.1. Architecture

BERT's architecture consists of multiple layers of bidirectional transformers, a type of attention mechanism that learns contextual relations between words (or sub-words) in a text. Unlike directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once.

Hence, it is considered bidirectional, though it would be more accurate to say it is non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

The mathematical model of BERT is represented as follows:

$$\mathbf{H} = \text{Transformer}(\mathbf{X})$$

where $\mathbf{X}$ is the input token embeddings, and $\mathbf{H}$ is the output of the transformer network which is used as the encoded feature representation for classification tasks.

4.4.2. Pre-training and Fine-tuning

BERT is pre-trained using two unsupervised tasks:

1. Masked Language Model (MLM): It randomly masks a percentage of the input tokens and predicts the masked words based on their context.

2. Next Sentence Prediction (NSP): It predicts whether a sentence follows another in a given text, encouraging the model to learn relationships between sentences.

For text classification, BERT is fine-tuned with additional layers:

$$\mathbf{P} = \text{Softmax}(\mathbf{W}_\text{t} \cdot \mathbf{H}_\text{CLS})$$

where $\mathbf{W}_\text{t}$ is the task-specific parameter matrix, $\mathbf{H}_\text{CLS}$ is the final hidden state corresponding to the first input token (CLS token), and $\mathbf{P}$ represents the probability distribution over labels.

4.4.3. Fine-Tuning for Specific Tasks

Fine-tuning BERT for text classification involves modifying the pre-trained model by adding a classification layer on top, and then training all the parameters end-to-end on a labeled dataset specific to the task. This approach has shown significant improvements over traditional models that rely on feature engineering and other machine learning algorithms.

The effectiveness of BERT in text classification is due to its ability to model complex patterns in data and generalize well to new, unseen examples. It has been successfully applied to tasks such as sentiment analysis, intent detection, and spam detection, demonstrating the versatility and robustness of the model in handling various types of text classification challenges.

4.4.4. Computational Considerations

Despite its success, fine-tuning BERT is computationally expensive and requires significant amounts of memory and processing power, typically necessitating the use of

GPUs or TPUs. This can be a limiting factor for deployment in resource-constrained environments.

BERT marks a major improvement in text classification through deep learning, serving as a versatile and robust tool for diverse natural language processing tasks. Its compatibility with existing technologies and scalability with large datasets render it invaluable for both academic research and practical applications.

4.5. Bio_ClinicalBERT

Bio_ClinicalBERT is a variant of BERT specifically adapted and pre-trained for clinical text applications. Developed by Emily Alsentzer it builds upon the original BERT architecture by integrating domain-specific knowledge from biomedical and clinical texts to enhance performance on healthcare-related NLP tasks.

4.5.1. Development and Pre-training

Bio_ClinicalBERT is initialized with weights from BioBERT, which itself is pre-trained on biomedical corpora including PubMed abstracts and PMC full-text articles. This initialization allows Bio_ClinicalBERT to inherit a rich understanding of biomedical language. Alsentzer et al. then further trained Bio_ClinicalBERT on the MIMIC-III dataset, a comprehensive database of clinical notes from intensive care units, which includes a broad spectrum of medical language and terminology used in clinical practice.

The model was pre-trained using the masked language modeling (MLM) objective, similar to the original BERT, but tailored to accommodate the nuances of clinical narratives. The pre-training process also involved segmenting the clinical notes into meaningful sections using rule-based parsers before applying sentence tokenization with SciSpacy, ensuring that the linguistic structure of clinical documentation was preserved.

4.5.2. Model Architecture

Bio_ClinicalBERT retains the architectural framework of BERT, consisting of multiple layers of transformer blocks that process tokens in parallel, capturing contextual relationships between words in clinical notes. The model utilizes a 12-layer transformer architecture with an attention mechanism that is particularly suited for handling long sequences typical in medical records.

4.5.3. Applications and Performance

This model has shown superior performance on several clinical natural language processing tasks, including named entity recognition (NER), relation extraction, and clinical concept normalization. The adaptation to clinical text allows Bio_ClinicalBERT to outperform general-domain BERT models and even BioBERT on tasks where deep domain knowledge is crucial.

4.5.4. Implementation and Usage

Bio_ClinicalBERT is available through the Hugging Face Transformers library, facilitating easy integration and deployment in clinical NLP pipelines. The model can be loaded with predefined weights and used in conjunction with a tokenizer specifically designed for clinical text, allowing researchers and practitioners to apply state-of-the-art transformer-based models directly to electronic health records (EHRs) and other clinical documentation.

4.5.5. Impact and Future Directions

The introduction of Bio_ClinicalBERT represents a significant step forward in the application of deep learning to the medical domain. By leveraging both the broad language understanding of BERT and the specialized knowledge from biomedical literature, Bio_ClinicalBERT provides a powerful tool for extracting insights from clinical texts, potentially supporting clinical decision-making and patient care optimization. Future work involves expanding the training corpora to include more diverse clinical settings and updating the model to handle multi-language EHRs, broadening the scope of its applicability in global health informatics.

5. Methodology

5.1. Data

5.1.1. MIMIC-III Clinical Database

MIMIC-III Clinical Dataset Description The MIMIC-III (Medical Information Mart for Intensive Care III) dataset is a large, freely accessible database containing deidentified health-related data associated with over forty thousand patients who were admitted to critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. The data is rich with information such as demographics, vital sign measurements, laboratory test results, medications, imaging reports, and mortality data, including post-hospital discharge outcomes.

Data Description: MIMIC-III is structured into 26 relational tables, linked by identifiers like SUBJECT_ID (unique patient), HADM_ID (hospital admission), and ICUSTAY_ID (ICU stay). This research project focused on three key tables:

- NOTEEVENTS.csv: Contains detailed notes including nursing and physician observations, discharge summaries, and other free-text forms of data related to patient care. Text data in this file was rigorously deidentified to remove personal health information.
- DIAGNOSES_ICD.csv: Lists the ICD-9 diagnosis codes for each hospital stay, linking specific conditions and diagnoses to patient admissions. It helps in understanding the medical issues a patient was treated for during their hospital stay.
- D_ICD_DIAGNOSES.csv: This dictionary table provides descriptions for the ICD-9 codes found in DIAGNOSES_ICD.csv, including both short and long titles for each code, which aids in interpreting the clinical codes without needing to reference external resources.

The MIMICIII datasets are primarily used in preliminary modeling (classification head parameter optimizing) and fine tuning of `Bio_ClinicalBERT` which includes optimising the parameters of the sequential classification neural network head layer and the last layer (11th layer).

Preprocessing Steps:

- Loading the data using pandas. Merging text descriptions from D_ICD_DIAGNOSES.csv to enhance the interpretability of diagnosis codes.
- Filtering and cleaning text data in NOTEEVENTS.csv to remove non-alphanumeric characters and stopwords for NLP tasks.

- Using regular expressions to further refine and extract specific data subsets for analysis, like identifying mentions of kinetic accidents or specific medical conditions.

5.1.2. Ingham Applied Medical Research - NSW healthcare data

The gold-standard, human labeled dataset with 1000 observations is used for evaluating the model's performance on unseen data. This dataset has 559 false labels (non kinetic accidents) and 439 true labels (kinetic accident cases).

This gold-standard dataset was used to set a benchmark for classification using keyword based regular expression, achieving a 92% classification accuracy.

The Ingham gold standard dataset is used to further fine-tune and optimize the parameters of the Bio_ClinicalBERT (final model) sequential classifier, employing an 80-20 train-test split ratio. This optimization includes adjustments to the sequential classification neural network head layer and the 11th layer.

5.2. Bidirectional Encoder Representations from Transformers (BERT)

Architecture: BERT (Bidirectional Encoder Representations from Transformers) utilises the Transformer's encoder mechanism. Depending on the model size, it features multiple layers of multi-head self-attention mechanisms and feed-forward networks (Rothman, 2022).

- **BERT-base:** Consists of 12 encoder layers, each with 12 attention heads, and a hidden layer size of 768.
- **BERT-large:** Comprises 24 encoder layers, with 16 attention heads each, and a hidden layer size of 1024.

Pretraining: BERT is pretrained on a large corpus using two main tasks:

1. **Masked Language Modeling (MLM)**: Randomly masks words in the input sequences and predicts them based on the context provided by the other non-masked words in the sequence.
2. **Next Sentence Prediction (NSP)**: Given pairs of sentences, predicts whether the second sentence is the subsequent sentence in the original document, facilitating an understanding of the relationship between consecutive sentences.

Fine-tuning for Binary Classification: For binary classification tasks, BERT can be fine-tuned with a simple output layer:

$$\begin{aligned} \text{Input Sequence} &: [\text{CLS}]\, x_1\, x_2\, \ldots\, x_n\, [\text{SEP}] \\ \text{BERT Output} &: \mathbf{h} = \text{BERT}(\text{Input Sequence}) \\ \text{Classification Output} &: y = \sigma(\mathbf{w}^\top \mathbf{h}_{\text{CLS}} + b) \end{aligned}$$

where:

- [CLS] is a special token added at the beginning of each sequence, whose final hidden state $\mathbf{h}_{\text{CLS}}$ is used as the aggregate sequence representation for classification tasks.
- σ denotes the sigmoid function, mapping the output to a $(0, 1)$ range, suitable for binary classification.
- $\mathbf{w}$ and b are trainable parameters of the output layer.

5.3. Hardware and Software

5.3.1. Hardware

High-Performance Computing (HPC) The High-Performance Computing (HPC) system at Western Sydney University was utilized to facilitate the training of the Bio_ClinicalBERT model, which is designed to handle large-scale computational tasks. This sophisticated infrastructure provided the necessary computational resources that enabled efficient management of the model's extensive requirements. The system includes multiple CPUs and high-end NVIDIA GPUs that support parallel processing and significantly accelerate machine learning computations. The system is equipped with substantial RAM and high-speed storage solutions, crucial for managing large datasets and model checkpoints. High-bandwidth networking is also integral, ensuring rapid data transfer rates that are essential during the model's training phase.

- `RAM: 20GHz - ampere20`
- `RAM: 40GHz - ampere40`
- `RAM: 80GHz - ampere80`

`Ampere 80` **with a100 chip GPU was utilised for this project**

Environment Setup Access to the HPC cluster was secured with specific user permissions to ensure data confidentiality and integrity. The software environment was standardized using containerization tools, which guaranteed a consistent and reproducible setup across various computing nodes.

HPC Modules used Modules available in the HPC is loaded using `module load` command followed by the module name

- `anaconda/conda3` for setting up conda environment for PPB project
- `PyTorch/Python3.10` for utilising the GPU

Data Management

Biomedical datasets were preprocessed to format and clean the data, ensuring compatibility with the BioClinicalBERT model's input requirements. The data was then split into training and testing sets using an 80-20 ratio, with stratified sampling to maintain consistent distribution across the sets.

5.3.1. Software

Development Environment The project utilized Jupyter Notebooks, python script, and SLURM scheduler.

Programming Language

- Python, version 3.10, served as the programming language due to its extensive support for scientific computing and machine learning libraries.
- Bash language was used in the HPC environment to navigate the HPC from the terminal.

Libraries

- **pandas** - Used for data manipulation and analysis. Provides data structures and operations for manipulating numerical tables and time series.
- **nltk** - Natural Language Toolkit, utilized for working with human language data (text). It supports tasks such as classification, tokenization, stemming, tagging, parsing, and semantic reasoning.
- **re** - Provides regular expression matching operations similar to those found in Perl. Used for text searching and manipulations.
- **numpy** - Fundamental package for scientific computing with Python. Supports large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- **from nltk.tokenize import word_tokenize** - Used to divide strings into lists of substrings or "tokens".

- **from nltk.corpus import stopwords** - Used to remove common 'stop words' from text data.
- **from nltk.stem import WordNetLemmatizer** - Implements a lemmatizer based on WordNet.
- **from sklearn.feature_extraction.text import CountVectorizer** - Converts a collection of text documents to a matrix of token counts.
- **from sklearn.decomposition import LatentDirichletAllocation** - Implements Latent Dirichlet Allocation (LDA), a model that is useful for finding abstract topics within a collection of documents.
- **torch** - Main library from PyTorch, used for tensor computations with strong GPU acceleration and building deep learning models.
- **torchvision** - Provides popular datasets, architectures, and image transformations for computer vision.
- **transformers** - Hugging Face's library for state-of-the-art natural language processing, typically used for models like BERT, GPT, etc.
- **matplotlib** - Used for creating static, interactive, and animated visualizations in Python.
- **seaborn** - A Python data visualization library based on matplotlib, providing a high-level interface for drawing attractive and informative statistical graphics.
- **scikit-learn** - A tool for data mining and data analysis. Used for implementing machine learning algorithms.
- **AutoTokenizer** - This library is instrumental in preprocessing text for deep learning models. 'AutoTokenizer' automatically retrieves the correct token class based on the pretrained model name or path. It ensures that the text inputs are transformed into a format compatible with the model, handling tasks such as tokenization, normalization, and padding.
- **AutoModel** - Complements the 'AutoTokenizer' by providing a method to instantiate the model architecture associated with the pre-trained weights. 'AutoModel' can dynamically load any transformer model from the Hugging Face model hub that matches the tokenizer's configuration. It abstracts the model specifics, making it easier to switch between different models without changing the underlying code structure.

Version control was managed using `Git` & `Overleaf` for LaTeX, documenting and enabling reversibility of all modifications.

SLURM The training of the BioClinicalBERT model leveraged the PyTorch library enhanced by CUDA for GPU acceleration, ensuring high computational efficiency. This setup was executed on a High-Performance Computing (HPC) system, managed via a SLURM job scheduler to effectively handle job queues and resource allocation, as demonstrated by the following SLURM configuration:

```
#!/bin/bash
#SBATCH --job-name=seqC
#SBATCH --output=outputbcb_%j.txt
#SBATCH --error=errorbcb_%j.txt
#SBATCH --partition=ampere80
#SBATCH --time=168:00:00
#SBATCH --gres gpu:0

# The environment and PyTorch module needs to be loaded in this exact
    order
# Activate the conda environment
module load anaconda/conda3
source activate
conda activate /home/22058122/BioClinicalBERT

# Load the PyTorch module with CUDA support
module load PyTorch/Python3.10

# Executing

# 1. Execute the notebook using nbconvert with Python 3 (Preliminary
    Model)
jupyter nbconvert --to notebook --inplace --execute BioClinicalBERT80.
    ipynb --ExecutePreprocessor.kernel_name=python3 --debug

# 2. Execute the notebook using nbconvert with Python 3 (Fine-tuning)
python3 seqClassifier.py --debug
```

Only one execution can be conducted at a time; the initial SLURM execution was configured solely for notebook operation, and the fine-tuning was limited to a separate execution line that did not include the Jupyter notebook execution.

5.4. Related Work and Contributions

5.4.1. Kinetic keyword based Regular Expressions

Dr. Jim Basilakis, my co-supervisor, has developed and refined a comprehensive regular expression (regex) with the gold standard dataset to accurately identify records related to vehicle incidents. This regex is crucial for filtering dataset entries that include specific terms associated with vehicle-related events. Below is the regex used in our study

Listing 1: Regular expression used to identify vehicle-related incidents.

```
vehicleRegex = (
    r'\b(?:mva|mba|vehicle|bus|pedestrian|passenger|ute|ped|bike|dirtbike
        |motorbike|pushbike|scooter|truck|'
    r'bicycle|motorcycle|driver|driving|rtc|rta|\d*km[a-zA-Z/]*|
        skateboard|surfing|surf|horse|collision|'
    r'crossing|buggy|ebike|jetski|vs␣car|car␣vs|car␣accident|moving␣car|
        traffic␣light|traffic␣lights|'
    r'hit␣by␣car|hit␣by␣a␣car|car␣hit|airbag|airbags|T␣boned)\b'
)
```

The regular expression `vehicleRegex` achieved a benchmark accuracy of 92% on the real data - the gold standard Ingham dataset.

The research was carried out to experiment and investigate potential applications of state of the art transformers or large language models for achieving better classification accuracy and address the limitations posed as challenge to key word based regular expression search. Transformers for classification has a requirement for annotated dataset to optimize the weight of the classification head.

5.4.2. Bio_ClinicalBERT

Emily Alsentzer's foundational work with `Bio_ClinicalBERT` - https://huggingface.co/emilyalsentzer/Bio_ClinicalBERT, developed during her tenure at Brigham and Women's Hospital and Harvard Medical School, is central to this study's approach. This project utilizes `Bio_ClinicalBERT`, a language model known for its robust performance in clinical settings, to improve decision-making and access to high-quality healthcare using machine learning (ML) and natural language processing (NLP).

Development and Training of Bio_ClinicalBERT The Bio_ClinicalBERT model extends the capabilities of Emily's initial model by integrating it with configurations from BioBERT and training it on the comprehensive MIMIC III database. The methodol-

ogy employs advanced NLP tools for precise data segmentation and sentence extraction, facilitating deep learning on a corpus of approximately 880 million words.

Model Pretraining and Deployment Model pretraining involved rigorous preprocessing of the MIMIC notes—segmenting each document into pertinent medical sections followed by sentence-level tokenization using SciSpacy's biomedical tokenizer. The training process, conducted on a GeForce GTX TITAN X GPU, leveraged a batch size of 32 and a maximum sequence length of 128, with a learning rate set at 5e-5. This meticulous training regimen, spanning over 150,000 steps, allowed the model to refine its linguistic comprehension, crucial for clinical applicability. The resulting model excels in nuanced language understanding, essential for its deployment in clinical settings.

Deployment is streamlined through the transformers library, facilitating seamless integration into clinical systems, as demonstrated by the following code snippet:

```
from transformers import AutoTokenizer, AutoModel
tokenizer = AutoTokenizer.from_pretrained("emilyalsentzer/
    Bio_ClinicalBERT")
model = AutoModel.from_pretrained("emilyalsentzer/Bio_ClinicalBERT")
```

5.5. Source Code

`Github Repository` : https://github.com/midhunshyam/PPB/

1. Preliminary modeling: https://github.com/midhunshyam/PPB/blob/main/premodel.ipynb
2. Fine-tuning Bio_ClinicalBERT https://github.com/midhunshyam/PPB/blob/main/seqClassifier.py
3. Data pre-processing and evaluation:
   - PDF version:https://github.com/midhunshyam/PPB/blob/main/mimiciii_datapreprocess.pdf
   - Jupyter notebook version:https://github.com/midhunshyam/PPB/blob/main/preprocess_mimiciii.ipynb
4. Final model is developed at Ingham, modifying the source code of seqClassifier.py - used for fine tuning. The modifications are limited to changing variable names, creating Tensor DataLoaders for training and testing using the gold standard NSW healthcare dataset. The code is not available on public platforms like `Github` due to privacy concerns of the healthcare data.

6. Preliminary Modeling

6.1. MIMIC-III Clinical Database Data Preprocessing

Data annotation In the preliminary phase of the study, the MIMIC III dataset was used to train the Bio_ClinicalBERT transformer model. The gold standard Ingham dataset was used as the test set, setting it as a robust measure of the model's generalisation capabilities on unseen data.

The annotation process begins with loading three key datasets: `NOTEEVENTS.csv`, `DIAGNOSES_ICD.csv`, and `D_ICD_DIAGNOSES.csv`. To enhance the details for each ICD diagnosis code, the `SHORT_TITLE` and `LONG_TITLE` columns from `D_ICD_DIAGNOSES.csv` are merged. This merged data creates a comprehensive description for each diagnosis. The ICD diagnosis codes encompassed over 14,000 unique entries.

The regular expression, `vehicleRegex` specifically designed to capture terms associated with vehicular and kinetic accidents is then applied to filter this combined dataset, isolating entries that are likely to involve such incidents. The filtered data undergoes several preprocessing steps, including converting text to lowercase, removing non-alphabetic characters, tokenizing words, removing common stopwords, and lemmatizing. These steps ensure the text is standardized and ready for further analysis.

With the text now preprocessed, a document-term matrix is created using a `CountVectorizer`. This matrix serves as the input for a Latent Dirichlet Allocation (LDA) model, which is used for topic modeling. The LDA model groups similar words into topics, providing insights into the prevalent themes within the clinical notes related to kinetic accidents. By analysing these topics, the most significant themes concerning various types of accidents can be identified, offering a deeper understanding of how these incidents are described in clinical documentation.

The documents associated with irrelevant themes are filtered out to refine the dataset. This crucial step narrows the focus to only accident-related content, enhancing the specificity and clarity of subsequent analyses. The refined dataset undergoes another round of LDA to provide a more focused insight into the relevant accident-related topics. This iterative process ensures that the analysis remains centered on the most pertinent information, removing any noise introduced by unrelated data.

Using the refined topics and extracted ICD-9 codes, the next step involves filtering the `DIAGNOSES_ICD` dataset to retrieve patient and admission IDs related to kinetic accidents. This integration links thematic findings from the LDA model with specific patient records. The corresponding clinical notes for these admissions are then extracted from the `NOTEEVENTS` table, focusing on text that mentions accidents. In the final step, the chief complaint and the first paragraph of the History of Present Illness are extracted from these notes. This extraction provides detailed insights into the primary reasons for each patient's hospital visit, shedding light on the nature and context of the injuries or accidents they experienced.

To create the training dataset, the relevant lines with the kinetic accident keywords of clinical notes that match the filtered ICD-9 codes are extracted and labeled as 1,

indicating their association with kinetic accidents. The dataset was balanced by including a complement of the ICD9 codes identified using LDA on `vehicleRegex`, ensuring it comprised both accident-related and unrelated entries for a balanced model training. A random sampling of clinical notes that do not meet the `vehicleRegex` criteria set by the ICD-9 codes is performed. These sampled entries are labeled as 0, representing non-accident-related cases. This approach ensures that the training dataset contains a mix of both positive and negative examples, which is crucial for training an effective binary classification model. By including this balanced representation, the model learns to distinguish between notes related to kinetic accidents and those that are not, and is expected to deliver predictive performance in real-world scenarios.

6.2. Bio_ClinicalBERT for classification - Preliminary model

A binary text classification algorithm is developed using *Bio_ClinicalBERT*, a variant of BERT fine-tuned on clinical text data. The algorithm utilises necessary libraries such as PyTorch, pandas, and Hugging Face's `transformers` library, which provides the pre-trained *Bio_ClinicalBERT* model. The model is configured for sequence classification with two labels, for binary classification task. The environment setup involves selecting the device (CPU or GPU) based on CUDA availability and freezing the pre-trained layers of *Bio_ClinicalBERT* to focus on fine-tuning only the classifier head. The approach intends to leverages the pre-trained knowledge of clinical language while allowing the model to adapt to the specific classification task presented in the training data.

The annotated dataset, stored in a CSV file, contains clinical notes in string format and corresponding binary labels. Tokenization is performed using the *Bio_ClinicalBERT* tokenizer, which converts the text into input IDs suitable for the model. This process also adds special tokens to guide the model's understanding of sentence structure.

The tokenized data is then padded and truncated to maintain a uniform input size across batches, which is crucial for efficient processing. PyTorch tensors are created from the tokenized text and labels, and `DataLoader` objects are used to manage batch processing during training and evaluation. This step is critical in handling large datasets and optimizing model performance.

The model's classifier head is fine-tuned using the AdamW optimizer. The original parameters of the model—weights from the pre-trained model saved on Hugging Face—were preserved by freezing all layers except for the classification head. During training, the model computes the loss between predicted and actual labels, back propagating the errors to adjust the classifier weights. The model's performance is evaluated through metrics like accuracy and the algorithm saves the best-performing model.

The train-test split ratio was set at 20% for training and 80% for testing. The intention is to match the proportion of the gold standard dataset to the actual unlabeled data in the NSW Healthcare Emergency Department. This setup aims to evaluate how the transformer performs under realistic conditions. The model's test performance on the MIMICIII annotated dataset is assessed using a test set, with a classification report and confusion matrix providing insights into its precision, recall,

and accuracy to evaluate success in clinical text classification.

6.3. Results

6.3.1. Annotated dataset

An overview of the annotated pre-processed Pandas DataFrame is captured (See Figure 7). The dataset consists of 45,572 entries, spread across three columns: `HADM_ID`, `text`, and `label`. The dataset ensures a balanced class distribution, with each category ('1' for Kinetic accident-related and '0' for non-Kinetic accident-related texts) containing 22,786 instances. This balance and structure ensures that the dataset is well-prepared for the binary classification task.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45572 entries, 0 to 45571
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   HADM_ID  45360 non-null  float64
 1   text     45571 non-null  object
 2   label    45572 non-null  int64
dtypes: float64(1), int64(1), object(1)
memory usage: 1.0+ MB
None

Class distribution:
1    22786
0    22786
Name: label, dtype: int64
```

Figure 7: Annotated dataset information

Unevaluated for preliminary model

	HADM_ID	text	label
32257	131151.0	This 85 Y/O with H/O dementia with proteus pos...	0
30399	138701.0	1900-0700 NPN SEE FLOWSHEET FOR DETAILS 1.) IN...	0
1485	195762.0	Patient is a 23 yo male without PMH s/p MVA vs...	1
41886	127135.0	Sinus tachycardia with increase in rate as com...	0
5557	151385.0	88F tx from OSH, restrained passenger MVC, T-b...	1
16780	135411.0	Admitting Diagnosis: MOTOR CYCLE COLLISION, IN...	1
8093	107641.0	Reason: s/p pedestrian MVA, s/p ped v car, r/o...	1
42558	142968.0	**Last Name (LF) **,**First Name3 (LF) 817** C...	0
21467	164648.0	Pt remains intubated and sedated in the ICU s/...	1
15924	184164.0	Admitting Diagnosis: S/P MOTOR VEHICLE ACCIDEN...	1

Figure 8: Random sample of 10 rows

MIMIC Annotated (unevaluated) for preliminary modeling

A glimpse of the dataset structured for analysing kinetic accidents is provided, featuring columns `HADM_ID`, `text`, and `label` (See Figure 8). Each row provides a snippet

describing a patient's medical condition, with binary labels indicating whether the text is relevant (1) or not (0) to the study of kinetic accident cases. The text descriptions vary widely, from explicit mentions of MVAs and their implications, such as "88F tx from OSH, restrained passenger MVC, T-b..." which highlights a traumatic incident related to a motor vehicle collision, to entries unrelated to accidents like "This 85 Y/O with H/O dementia with proteus pos...".

6.3.2. Bio_ClinicalBERT

The training loss curve provides insight into the model training dynamics over time (See Figure 9). The model was trained over 2000 epochs, completing the `training in a duration of about 142,634.90 seconds (1 day, 16 hours approximately) using a 80GHz A100 GPU`. The training loss curve depicts a sharp decline in loss at the initial stages, quickly stabilizing to a plateau as training progresses. The loss was reduced to approximately 3% after 2000 epochs. The train and test data are split in a proportion of 80% for training and 20% for testing.

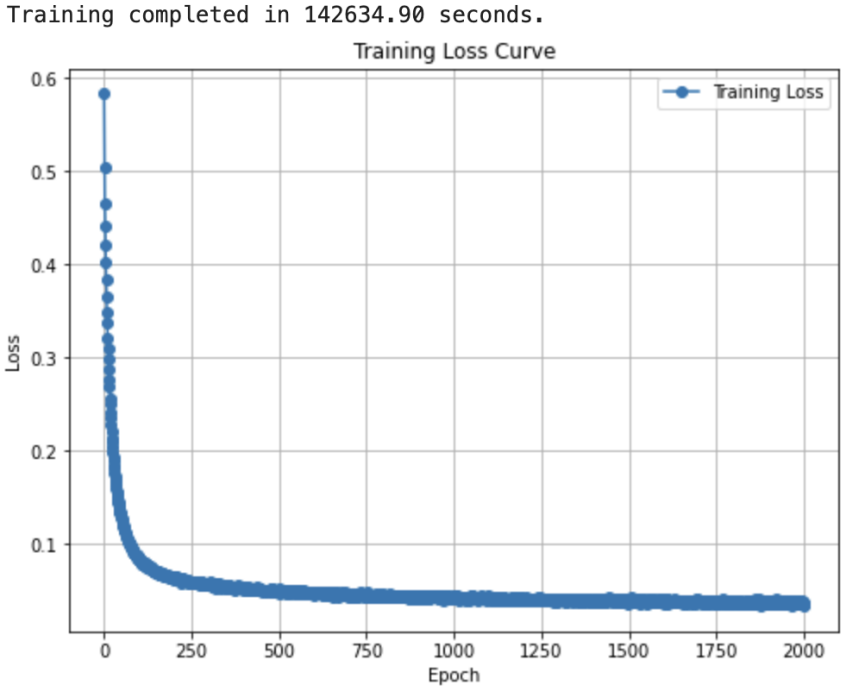

Figure 9: Bio_ClinicalBERT (Preliminary model) training loss curve

The model performance can be evaluated using the classification report and confusion matrix (See Figure 10).The model exhibits an accuracy of 98.83% as indicated by the testing results, which completed in 263.97 seconds. The misclassification error rate stands at a low 1.17%, demonstrating the model's effectiveness in classifying between

the two classes. The classification report reveals high precision, recall, and f1-score of 0.99 for both classes. However, 267 false negatives and 158 false positives are observed in the confusion matrix. False negatives are critical in this research because it is important to identify the patients who are involved in kinetic accidents. Implicating that model requires further fine tuning.

```
Testing completed in 263.97 seconds.
Accuracy: 0.9883
Misclassification Error Rate: 0.0117

Classification Report:

              precision    recall  f1-score   support

     Class 0       0.99      0.99      0.99     18074
     Class 1       0.99      0.99      0.99     18214

    accuracy                           0.99     36288
   macro avg       0.99      0.99      0.99     36288
weighted avg       0.99      0.99      0.99     36288
```

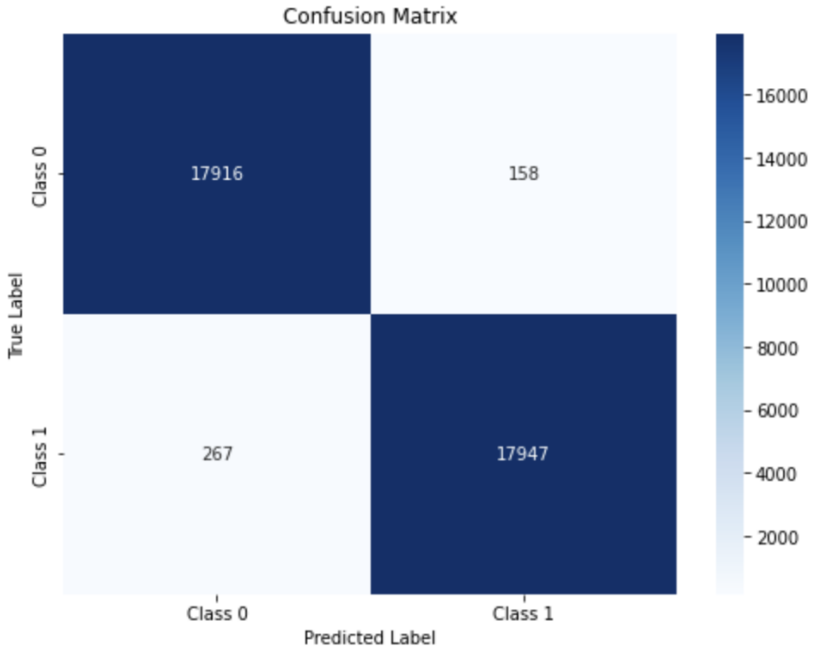

Figure 10: Preliminary model test

Classification Report (20-80 Split)

6.3.3. Gold standard test evaluation

Testing the model with a fine-tuned classification head trained on the annotated dataset for 2000 epochs demonstrated poor performance on the gold-standard dataset. It achieved an F1 score of 30 for true cases and 50 for false cases.

1. The trained model successfully identified less than 30% of true cases in the NSW Healthcare test dataset, failing to capture 70% of the variation.
2. The preliminary model with fine-tuned classification head cannot address the issue with typos.
3. There are masks and special characters in inputs of the annotated dataset.
4. The MIMIC-III dataset uses American English, while the NSW Healthcare data uses British English, leading to structural and spelling difference in the input sentences.

7. Model Fine-tuning

7.1. Evaluation of the annotated dataset

The annotated dataset `'train.csv'` comprises 45,572 non-null entries, with three columns: 'HADM_ID' (Hospital Admission ID), 'text' (clinical descriptions), and 'label' (binary indicator for whether the text describes a kinetic accident case). A subset of the data is created where 'label' equals 1, filtering the true kinetic accident cases. This subset, stored in 'ka', contains 22,786 entries.

Random sampling of 100 entries is performed ten times from the kinetic accident subset ('ka') with only true cases. Each sample is seeded differently (using integers from 0 to 9) to ensure reproducibility. The resulting samples are stored in the list 'samples'. Each of these samples is then printed, displaying both the 'HADM_ID' and the 'text' for each sampled record, providing a glimpse into the diversity of descriptions related to kinetic accidents.

The random samples were manually evaluated to identify false positives and false negatives; example: (See Figure 11). No incorrect labels could be identified in the 10 random samples manually evaluated. However, given that human evaluation could have mistakes and that the proportion of random samples evaluated is less than 10% of the true label observations.

Then text length analysis for conducted for both labels, kinetic and non-kinetic cases. The results show an average length of 18.14 words and a maximum of 343 words for kinetic accident-related texts. The length of 'text' in the non-kinetic subset entries averages 283.88 words, with a maximum of 6,918 words, indicating more verbose descriptions in non-kinetic accident-related texts.

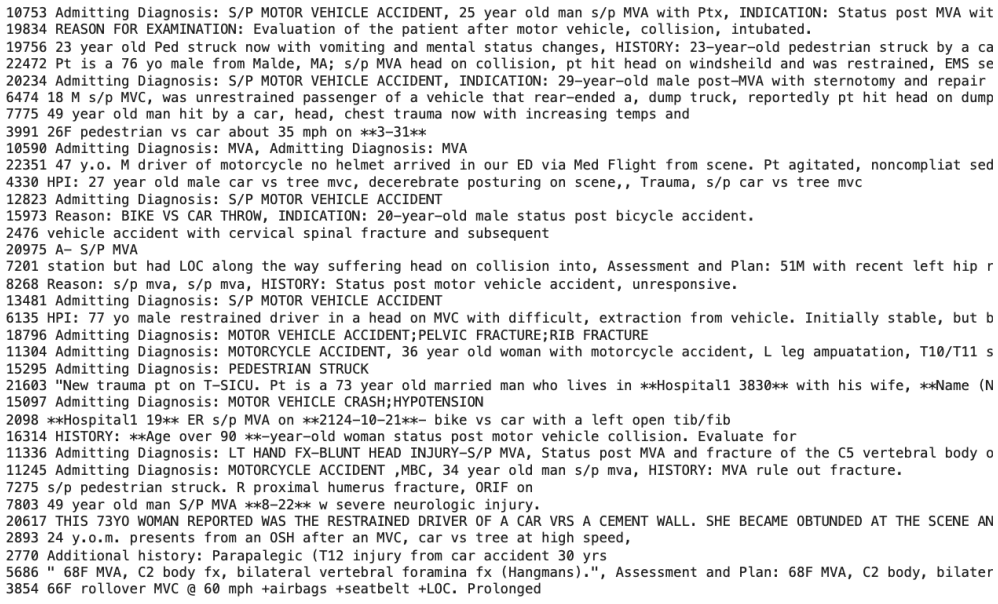

Figure 11: Manual evaluation of annotated dataset

Random sample 1/10

The Bio_ClinicalBERT transformer's maximum input is limited to 512 tokens. A function `extract_first_two_paragraphs` is defined to truncate the `text` to the first two paragraphs and limit the word count to 512. This function is applied to the non-kinetic subset, updating the `text` column with the truncated text. After updating the dataset, the Maximum Text Length is 512 words, Minimum Text Length is 2 words, and Average Text Length is 113.97 words.

Rows containing fewer than 2 words in the 'text' column are identified and removed, reducing the dataset by 21 entries. Duplicate rows based solely on the 'text' column are removed, eliminating a total of 10,094 entries.

To address an imbalance in the distribution of the labels, the dataset is balanced by sampling from the non-kinetic accident entries to match the number of kinetic accident-related entries. Both classes are subsequently shuffled and the indices reset, achieving an even distribution with 12,934 entries per class. This balanced dataset ensures equal representation of both labels and is crucial for unbiased machine learning model training.

The vital steps enhancing data quality, preparing the dataset for further training and finetuning on the Bio_ClinicalBERT transformer model. The updated dataset is saved as `train.csv` overwriting the older version.

7.2. Fine-tuning Bio_ClinicalBERT

The model's parameters were adjusted by freezing certain layers while unfreezing others, such as the classifier and the last two layers of the encoder. This approach is designed to fine-tune the model on specific features, adjusting more parameters with regard to the new dataset without altering the fundamental representations learned from vast clinical datasets.

The manually evaluated dataset, cleaned and balanced `train.csv` is loaded into the workspace, with labels verified as integers to maintain data integrity. Text and labels are extracted, then tokenized using Bio_ClinicalBERT's pretrained Autotokenizer with padding and truncation to ensure uniform sequence lengths. These are converted into tensors for PyTorch processing.

A DataLoader is then established to efficiently manage data batching and shuffling during training. The model is set to evaluation mode to stabilize results before training begins.

The data was split into training and testing subsets, with a 70-30 ratio, ensuring sufficient data was available for both training and validating the model's performance. Corresponding DataLoaders for each subset were prepared, setting the stage for the fine-tuning process, setting up the learning rate and optimizer.

The model underwent a training process over 10 epochs, leveraging the AdamW optimizer to implement gradient descent, iteratively refining the model by minimizing loss at each epoch. Each training epoch was timed, with performance metrics calculated in real-time to evaluate the model's effectiveness and adjust the gradient updates accordingly.

`Training time: 25 mins on A100 80GHz Ampere80 node` After training, the model was saved to reuse or deploy the fine-tuned weights. The training loss values were plotted to illustrate the learning curve, shedding light on the training dynamics.

The model was set to evaluation mode for final testing against the test set, where predictions and labels were efficiently inferred. This phase was timed, with accuracy and misclassification rates calculated to assess the model's performance, along with classification report and confusion matrix provided insights into the model's precision, recall, and F1-score, crucial for evaluating its performance in classifying kinetic and non-kinetic accident cases.

7.3. Results

The significant decrease in the initial epochs suggests that significant learning and model adjustments occurred early in the training phase (See Figure 12), with subsequent epochs refining the model's weights and biases to optimize performance. The loss value approaches zero and stabilizes towards the end of the training, which typically suggests that the model has reached or is approaching an optimum state, where further training would yield diminishing returns.

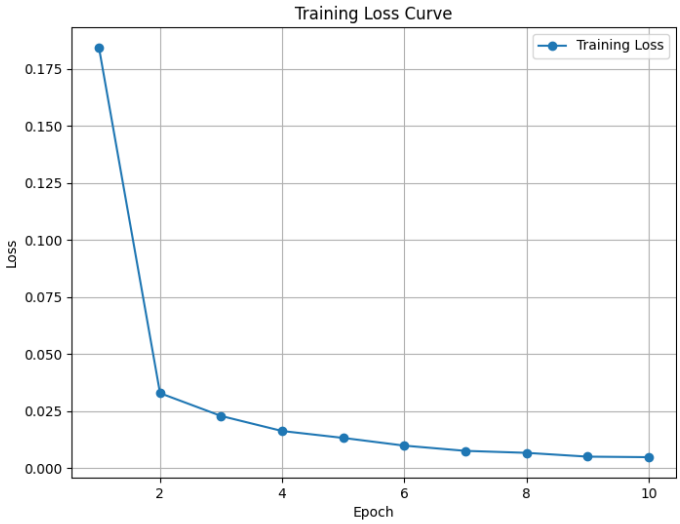

Figure 12: Training loss curve - 10 Epochs

MIMICIII dataset

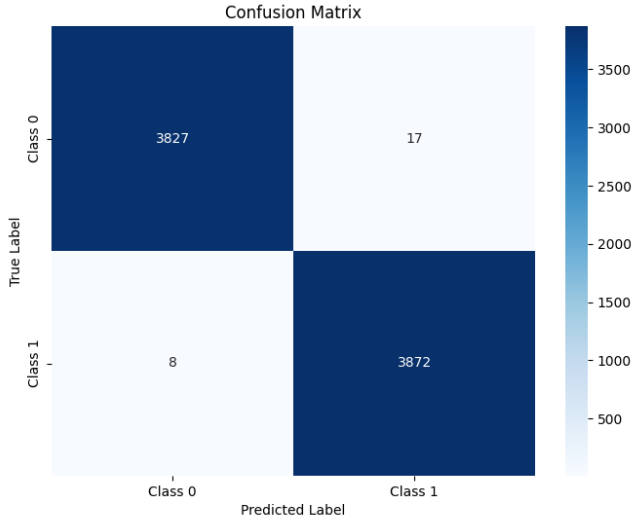

Figure 13: MIMIC Test-split Confusion Matrix (70-30 split)

The confusion matrix (See Figure 13) reveals a high degree of model precision and recall, with 3827 true positives and 3872 true negatives for classes 0 and 1, respectively, and only a minimal number of misclassifications: 17 false positives and 8 false negatives.

This underscores the model's effectiveness in accurately classifying cases, a result of successful fine-tuning and training processes.

The high accuracy and low misclassification rate, reported as 99.68% and 0.32% respectively, highlight the model's capability to generalize well from the training data to unseen data. The difference between the model's perfect F1-score and its slightly lower accuracy arises primarily due to the small number of false positives and false negatives impacting the accuracy more significantly than the F1-score. Accuracy, which measures overall correctness, is slightly reduced by these errors, while the F1-score, which balances precision and recall, remains perfect due to the relatively higher number of true positives and negatives. This indicates that the model is very effective at correctly classifying most cases, with only a minimal number of errors not significantly impacting its ability to balance recall and precision.

7.3.1. Gold standard test evaluation

Testing the model, which featured a fine-tuned classification head and the 11th layer of Bio_ClinicalBERT, on a refined annotated dataset over 10 epochs, showcased better performance on the gold-standard dataset compared to the preliminary model which was only fine-tuned on the classification head.

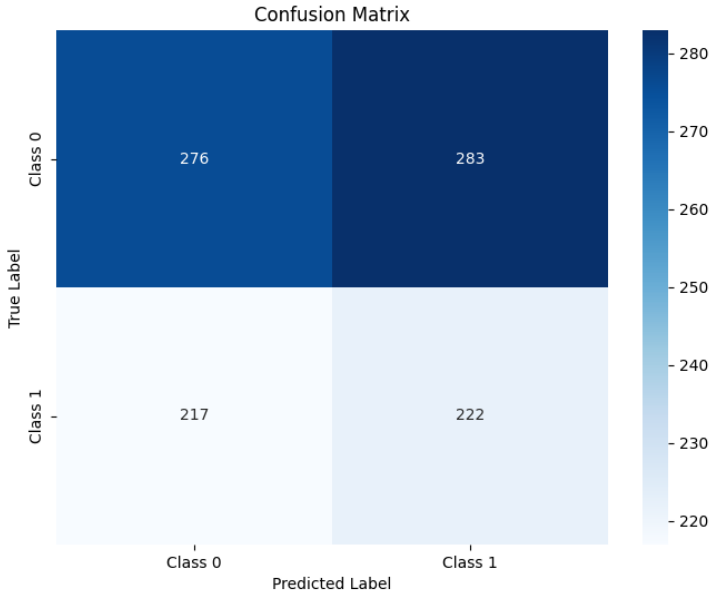

Figure 14: Fine-tuned model (C.H. + L11)

Gold standard test (1k obs)

The classification report provides insights into the fine-tuned model's performance on a dataset consisting of 998 observations, divided into two categories labelled "0" and "1". For class 0, the model's precision rate was 56%, indicating that slightly more than half of the predictions made for this category were accurate. The recall rate for the same class was 49%, showing that the model could identify only about half of the actual instances belonging to class 0. The F1-score, which harmonises precision and recall, was 0.52, suggesting a reasonable but not optimal balance between the two metrics. There were 559 instances belonging to class 0 that the model had to predict.

For class 1, the model had a lower precision of 44%, meaning less than half of the predictions for this class were correct. However, its recall was slightly higher at 51%, indicating that the model was somewhat more successful at identifying the true class 1 instances than class 0. The F1-score for class 1 was 0.47, again reflecting an average performance that combines precision and recall. This class consisted of 439 instances.

The overall accuracy of the fine-tuned model was 0.56, which demonstrates that it correctly predicted the classification of an observation half the time across the entire dataset. Both the macro average and weighted average for precision, recall, and F1-scores stood at about 0.50, signalling moderate effectiveness. These figures suggest that although the model performs evenly across both categories, its ability to distinguish and accurately classify each instance could be enhanced further.

8. Final model

The pre-trained weights of the fine-tuned model was retained and the classification head and layer 11 weights were further fine-tuned and optimsed using AdamW optimiser on the gold standard dataset with 80% training and 20% test ratio.

The final model was trained for 130 epochs which took 3054 seconds (50 mins) on a cpu for a balanced gold standard dataset with 962 observations, 481 belonging to each class. The model completed its testing phase in 3.19 seconds, achieving a high accuracy rate of 93.75%. This performance is commendable given the complexity of natural language processing tasks, particularly those involving clinical texts. The misclassification error rate stands at a low 6.25%, underscoring the model's efficacy in categorizing observations correctly.

The detailed classification report provides additional insights into the model's performance across the two classes. For Class 0, the model reached a precision of 95% and a recall of 94%, yielding an F1-score of 94%. This class had 93 instances, indicating a robust ability to identify and correctly classify these samples. Class 1, with 83 instances, also showed strong results, with a precision of 93% and both recall and F1-score at 94%.

These metrics highlight the model's balanced performance in distinguishing between the two classes, supported by high precision (the percentage of relevant instances among the retrieved instances) and recall rates (the percentage of relevant instances that were retrieved). The overall F1-scores, averaging at 94% both on a macro and weighted basis, suggest a highly effective harmony between precision and recall across the classes,

reinforced by the uniform accuracy.

In summary, the optimization of the model's classification head and specific layers using the AdamW optimizer, coupled with extensive training over 100 epochs, has led to a model that not only performs with high accuracy but also maintains consistency across different statistical measures, thus providing a reliable tool for clinical text analysis. This level of performance showcases the potential of advanced machine learning techniques in enhancing the interpretation and classification of complex datasets in the medical domain.

8.1. Result

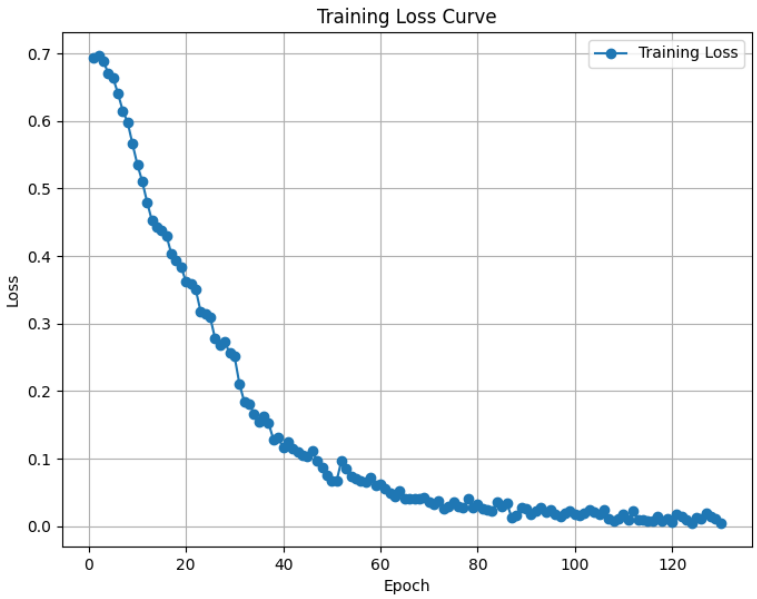

Figure 15: Final fine-tuning on gold-standard dataset

The training loss curve (See Figure 15) provides a clear depiction of the model's learning progress over approximately 120 epochs. Initially, the loss starts quite high at around 0.7, indicating a significant discrepancy between the predicted outputs and the actual labels in the dataset. As the training progresses, the loss swiftly decreases, suggesting rapid learning and adjustment by the model to minimize errors in prediction.

Steep decline in loss continues sharply until about the 40th epoch, where it levels off around a loss value of 0.2. Beyond this point, the curve gradually flattens, indicating that further training yields diminishing returns in terms of loss reduction. This behavior typically shows that the model has reached a state where it has effectively learned the

patterns in the data and further training would likely not lead to significantly better performance, at least in terms of reducing training error.

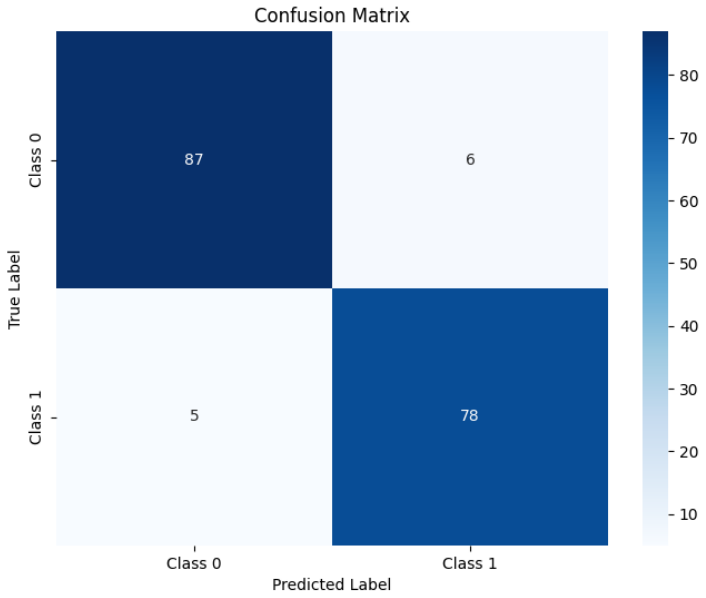

Figure 16: Final model test (GS:80-20)

The Bio_ClinicalBERT with 110 million parameters were able to achieve a benchmark accuracy of 93.75% on the test dataset (Gold standard Ingham dataset split at 20% for test ratio).

9. Extended analysis & Findings

The 11 misclassifications were manually evaluated for root cause analysis, which provided the below findings;

1. The model incorrectly labeled cases which had sentence structured "post MVA", "skate board accidents", and "fall from horse" as false or none-kinetic accidents, whereas in the gold standard it is labeled kinetic accidents.

2. There are mislabels in training data which are crucial in the model's performance, and needs to be evaluated. Four cases identified where the model wrongly predicted the label, meaning that the training should have involved mislabeled observations.

3. The remaining five cases were correctly predicted by the fine-tuned final model of sequential classifier, it was able to identify some of the incorrectly labeled observations in the gold standard dataset.

	Predicted Class 0	**Predicted Class 1**
Actual Class 0	90	4
Actual Class 1	2	80

Table 1: Final model test confusion matrix

Post analysis

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{80 + 90}{80 + 90 + 2 + 4} = \frac{170}{176} \approx 0.9659 \text{ or } 96.59\%$$

10. Recommendations & Conclusion

Recommendations:

- The quality of training data is crucial in training the `Bio_ClinicalBERT`
- Annotation or labeled data is expensive and is a requirement for successful implementation of the transformer model `Bio_ClinicalBERT`
- The model has potential in achieving high classification accuracy and can address challenges associated with other methods such as keyword based regular expressions for direct classification.
- However the algorithm is an integration of existing techniques such as keyword based regular expression, latent Dirichlet allocation, and state of the art, transformers, for achieving a higher performance.

References

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Cardinal, R. N. (2017). Clinical records anonymisation and text extraction (crate): an open-source software system. *BMC medical informatics and decision making*, 17:1–12.

Herland, M., Khoshgoftaar, T. M., and Wald, R. (2014). A review of data mining using big data in health informatics. *Journal of Big data*, 1:1–35.

Kuo, C. (2023). *The Handbook of NLP with Gensim: Leverage topic modeling to uncover hidden patterns, themes, and valuable insights within textual data*. Packt Publishing Ltd.

Lee, J. and Toutanova, K. (2018). Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 3(8).

Levesque, H., Davis, E., and Morgenstern, L. (2012). The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Rothman, D. (2022). *Transformers for Natural Language Processing: Build, train, and fine-tune deep neural network architectures for NLP with Python, Hugging Face, and OpenAI's GPT-3, ChatGPT, and GPT-4*. Packt Publishing Ltd.

Vajjala, S., Majumder, B., Gupta, A., and Surana, H. (2020). *Practical natural language processing: a comprehensive guide to building real-world NLP systems*. O'Reilly Media.

Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

Appendix

A. Output

A.1. Fine-tuned model gold standard test output details

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 3 columns):
#    Column            Non-Null Count  Dtype
---  ----------------  --------------  -----
0    'Encntr_ID'       1000 non-null   int64
1    'ED Triage Comment'  1000 non-null   object
2    'label'           1000 non-null   int64
dtypes: int64(2), object(1)
Memory usage: 23.6+ KB

Value count after removing the 'maybe' labels:
label
0 559
1 439

Name: count, dtype: int64
Some weights of BertForSequenceClassification were not initialized from the model checkpo
You should probably TRAIN this model on a downstream task to be able to use it for predic

Test time: 43.969003438949585
[276 283]
[217 222]
```

Class	Precision	Recall	F1-Score	Support
Class 0	0.56	0.65	0.60	655
Class 1	0.44	0.51	0.47	439
Accuracy			0.56	998
Macro Avg	0.50	0.50	0.50	998
Weighted Avg	0.50	0.50	0.56	998

A.2. Final model: Training and Testing Logs

```
Epoch 105/130 completed. Loss: 0.0176
Epoch 106/130 completed. Loss: 0.0236
Epoch 107/130 completed. Loss: 0.0112
Epoch 108/130 completed. Loss: 0.0074
Epoch 109/130 completed. Loss: 0.0107
Epoch 110/130 completed. Loss: 0.0176
Epoch 111/130 completed. Loss: 0.0092
Epoch 112/130 completed. Loss: 0.0220
Epoch 113/130 completed. Loss: 0.0088
Epoch 114/130 completed. Loss: 0.0089
Epoch 115/130 completed. Loss: 0.0071
Epoch 116/130 completed. Loss: 0.0075
Epoch 117/130 completed. Loss: 0.0146
Epoch 118/130 completed. Loss: 0.0076
Epoch 119/130 completed. Loss: 0.0107
Epoch 120/130 completed. Loss: 0.0054
Epoch 121/130 completed. Loss: 0.0180
Epoch 122/130 completed. Loss: 0.0146
Epoch 123/130 completed. Loss: 0.0092
Epoch 124/130 completed. Loss: 0.0046
Epoch 125/130 completed. Loss: 0.0125
Epoch 126/130 completed. Loss: 0.0105
Epoch 127/130 completed. Loss: 0.0189
```

```
Epoch 128/130 completed. Loss: 0.0145
Epoch 129/130 completed. Loss: 0.0108
Epoch 130/130 completed. Loss: 0.0047

Model weights saved to /home/midhun/Downloads/fine_tuned_clinical_bert_sequential_classi

Training completed in 3054.12 seconds.

/home/midhun/Downloads/seqClassifierIn.py:146: UserWarning: FigureCanvasAgg is non-intera
plt.show()

Testing completed in 3.19 seconds.
Accuracy: 0.9375
Misclassification Error Rate: 0.0625

Classification Report:

              precision    recall  f1-score   support

     Class 0       0.94      0.94      0.94        93
     Class 1       0.94      0.94      0.94        83

    accuracy                           0.94       176
   macro avg       0.94      0.94      0.94       176
weighted avg       0.94      0.94      0.94       176
```