Automating Patient Safety Event Report Classification using Natural Language Processing and Machine Learning

by

Shehnaz Islam

A thesis submitted in conformity with the requirements
for the degree of Masters of Applied Science

Department of Mechanical and Industrial Engineering
University of Toronto

Automating Patient Safety Event Report Classification using Natural Language Processing and Machine Learning

Shehnaz Islam
Masters of Applied Science
Department of Mechanical and Industrial Engineering
University of Toronto
2024

Abstract

Incident reporting (IR) in healthcare organizations aims to improve the quality of care and patient safety. The primary purpose of IR is to identify safety issues and develop interventions to mitigate these hazards, thereby reducing harm in healthcare [21]. IR enables healthcare organizations to document Patient Safety Event (PSE) reports, detailing incidents that compromise patient safety, including medical errors, near misses, or adverse events [178]. Accurate classification of PSE reports into their corresponding incident type and severity level is crucial for analyzing trends, guiding interventions, and enhancing organizational learning [51, 15]. However, the high volume of reported PSEs [94] and the complexity of the classification taxonomy, which requires specialized knowledge for categorization, make the labelling process labour-intensive, time-consuming, and expensive [102, 43]. In this study, we aim to develop and evaluate machine-learning models that predict the type and severity level of an incident based on textual description of the event in a PSE report. Our goal is two-fold. First, we aim to build predictive models that can identify high-severity PSE reports. This prioritization could speed up the analysis of these reports, improving overall patient safety and healthcare quality. Second, we investigate Active Learning (AL) strategies to streamline the manual labeling process for assigning event types to PSE reports. This approach aims to reduce labeling efforts while enhancing classification performance by querying instances that, if annotated by human experts, would significantly improve classification accuracy. We investigate the application of these models to PSE report datasets from two institutions: a large hospital in Canada and an academic hospital in the Southeastern United States. Our results show that models like LightGBM, trained on domain-specific representations such as GatorTron, significantly improve the ranking quality of incidents by severity level, with R-precision increasing from 0.197 to 0.4259 and MAP from 0.1546 to 0.4205. We also observe that using Active Learning strategies over baseline random sampling reduces the need for labeled samples by 24-69%, effectively decreasing manual workload while maintaining high classification accuracy.

Keywords: Incident report, Patient Safety Incident Classification, Incident Severity Prediction, Text Classification, NLP, Healthcare, AI, Deep Learning, Machine Learning

Acknowledgements

Firstly, I am immensely grateful to my supervisor, Professor Eldan Cohen, for believing in my potential and guiding me throughout my academic journey. Thank you for being flexible and allowing me to pursue a research direction aligned with my interests. His continuous support and invaluable insights have been pivotal to the success of this thesis. I would also like to thank him for giving me the opportunity to merge my master's studies with practical experience, leading a technical project in the healthcare organization, which helped me grow in both academic knowledge and professional skills. I have learned a great deal in the past two years. Thank you for helping me advance my knowledge in Machine Learning and Deep Learning and for guiding me in applying these skills to solve real-world problems.

I would also like to extend my gratitude to Professor Myrtede Alfred for introducing me to the field of Human Factors and the integration of AI in healthcare. Thank you for your constant support and feedback, and for giving me the opportunity to contribute to improving patient safety and present my work at the Human Factors Society. Additionally, I would like to thank the committee members of the healthcare organization and my managers for entrusting me with real-world data and allowing me to lead the quality and safety project. Their support, insights, guidance, and feedback on my thesis have been incredibly helpful.

I am thankful to Professor Mark Chingnell for being a part of the defense committee and dedicating his time to review my thesis and provide constructive feedback.

Fortunately, I have been part of an incredible team at the Optimization and Machine Learning (OPTIMAL) Lab at the University of Toronto. I would like to thank Rob, Nakul, Zhijian, Junda, James, Hriday, and Yulian for their support, creative ideas, and for making the past two years so memorable.

Finally, I must express my deepest gratitude to my family, friends, and previous mentors. Their belief in me and their unwavering love and support have been instrumental in shaping who I am today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In healthcare organizations, Patient Safety Events (PSEs) are incidents or circumstances, including medical errors, near misses, and adverse events, that could have or did result in unnecessary harm to a patient [172, 78]. PSEs resulting from unsafe care represent a significant and growing global public health challenge, and are among the leading causes of death and disability worldwide [124]. In 2000, the study "To err is human" estimated that clinical errors cause about 44,000 to 98,000 deaths annually in the USA, with later studies raising this estimate to 251,454 deaths, making medical errors the third-leading cause of death in the USA [44, 112]. The global cost of medication errors alone is estimated at US$ 42 billion annually [124]. An adverse event is a type of PSE that leads to patient harm resulting in prolonged hospitalization, disability, or death due to healthcare management [136]. Research from a large international review of patient charts indicates that approximately 4-17% of hospital admissions are associated with an adverse event and a considerable proportion of these events (one to two-thirds) can be prevented [136]. This signifies the need for effective incident reporting and analysis mechanisms to mitigate such occurrences.

Following the Global Patient Action Plan by the WHO in 2020 [124], healthcare organizations have widely adopted Incident Reporting (IR) systems to document PSE reports. These systems are crucial for the collection, analysis, and management of PSEs, thus contributing to the development of safer healthcare policies and mitigate future risk [44]. PSE reporting systems enable healthcare personnel, to voluntarily log incidents, in the form of structured and unstructured data including details like the type of incident (e.g., fall, burn), initial harm level (e.g. moderate, severe, critical), timing, location and incident description. Once submitted, these reports are then reviewed by hospital staff such as risk managers and Patient Safety Analysts (PSAs), playing a crucial role

in improving healthcare quality and safety by providing insights into factors leading to PSEs and facilitating the development of preventive measures [73].

Accurately categorizing the PSE reports into appropriate event types and harm level is crucial for analyzing trends, guiding interventions, and enhancing organizational learning [51, 15]. However, frontline reporters—such as point-of-care clinicians, nurses, physicians, or technicians—who are often not domain experts in PSE classification taxonomy, can inaccurately classify these reports when initiating them in the system. Therefore, the success of these reporting systems greatly depends on the organization's ability to efficiently post-process and analyze the large volume of PSEs reported [46, 111]. Currently, PSAs or risk managers are often required to manually perform retrospective review of all reports generated daily. This process includes, but is not limited to, validating or reclassifying incidents into their appropriate types and severity levels, as well as identifying potentially severe and preventable safety events for further and deeper analysis. A significant challenge faced by PSAs is the timely and accurate analysis of large volume of PSE reports and prevent their recurrence, given many reports are incomplete or inaccurate. This labor-intensive process not only reduces PSAs' efficiency and capacity to work on other care improvement initiatives but also results in only a small fraction of reports being accurately annotated with an event type or severity level.

The primary objective of this research is to utilize Natural Language Processing (NLP) and Machine Learning (ML) techniques for the efficient classification of Patient Safety Event (PSE) reports in healthcare environments. While previous studies have attempted to use NLP and ML to support PSE report classification [52, 123, 172, 48, 30, 29], demonstrating efficacy in accurately classifying these reports, the use of more advanced NLP techniques using neural network language models is limited [185]. Hence, in this study, we explore both traditional static NLP techniques as well as deep-learning based contextual NLP techniques involving language models, such as BERT (Bi-directional Encoder Representations from Transformers) [41], capable of capturing semantic meaning of text, as detailed in Chapter 2, Section 2.1.4. Specifically, in this study we make two main contributions:

1. First we train and evaluate Learning-to-rank Ranker and Classifier models based on Gradient Boosting Machines (GBM), across various NLP techniques, to automate the identification of critical safety events of highest severity level (HPI Serious Safety Event Levels 1-4). To our knowledge, this is the first work to implement LTR models for PSE report classification. Our findings indicate that LightGBM classifier model with healthcare domain-specific features significantly outperforms existing baselines, and has the potential to improve the efficiency of

the PSA review process and reduce manual screening efforts.

2. To address the challenge of limited availability of accurately labelled PSE reports, we evaluate the efficacy of integrating Active Learning (AL) strategies to guide PSE report labelling. Our results demonstrate that informed AL strategies significantly outperform random sampling methods. Thus, AL can optimize the use of human resources and facilitate the development of a robust dataset for training ML models, thereby improving model performance with limited annotated data.

The overall goal of this study is to improve patient safety and streamline incident classification in healthcare organizations, supporting timely and efficient identification of PSEs and their analysis by PSAs. This study utilizes PSE reports from a large hospital in Canada, to train and assess the predictive models for severity classification of PSEs. For event type classification using AL approaches, it utilizes a separate dataset of PSI reports specific to maternal care units from a Southeastern academic hospital in United States.

1.0.1 Organization

Chapter 2 provides background on various NLP and Active Learning techniques relevant to this study. In Chapter 3, we evaluate GBM Learning-to-rank models, across various NLP techniques, to automate the identification of critical safety events. In Chapter 4, investigates the efficacy of Active Learning strategies in improving the performance of ML models in event type classification of PSE reports. Finally, Chapter 5, we summarize our contributions and some potential future research directions stemming from the findings of this thesis.

Chapter 2

Background

This chapter outlines the Natural Language Processing (NLP) and Machine Learning (ML) techniques relevant to this thesis. Patient Safety Event (PSE) report, primarily composed of text data, details the type, severity, and sequence of events describing the patient safety incident. To train ML models with this textual information, it is essential to convert the free-text incident descriptions into numeric word representations. The following section provides background on various text representation methods and Active Learning techniques relevant to this study.

2.1 Text Representation

The biggest challenge with textual data, is that ML models cannot process textual input in their raw form. It requires the input text to be represented as a numeric vector instead, that can be used to optimize the model parameters to be able to learn the patterns in the text. In Natural Language Processing this conversion from text to numeric form, has been done using several approaches, evolving from simple word count representations to word feature representations. This section discusses the core ideas behind word representations, various techniques for converting text to numeric features, and how they have evolved over time.

We begin with traditional static text representation methods developed before the advent of deep learning, such as Bag-of-Words (BOW) and Term Frequency-Inverse Document Frequency (TF-IDF). We then progress to advanced neural contextual text representations, including Word2Vec and BERT.

2.1.1 Bag-of-Words (BOW)

The BOW model is a simple yet popular NLP approach [133] where a document is represented as a collection of words, ignoring the order of words. Each document is represented as a vector of dimension v, where v denotes the vocabulary size (i.e the total number of unique words across the entire corpus). The process begins by tokenizing the text into individual words or n-grams, using techniques like splitting the text on white spaces. A dictionary of all unique words is created, and for each document, a vector is generated based on this dictionary. While BOW representations are effective for certain analyses, it has limitations. It ignores word order and context, leading to potential loss of meaning, and also results in a sparse matrix due to the large number of zeros from absent words.

Binary vs. Non-Binary Count Vectorizer

Documents can be represented using a Binary Count vectorizer, which indicates the presence (1) or absence (0) of words, or a Non-Binary Count vectorizer, which counts the frequency of each word.

2.1.2 TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF [113] is the most commonly used text representation technique in information retrieval systems. It is based on the principle that the importance of a term within a document set increases proportionally to the term's frequency but inversely with its prevalence across all documents. As the term suggests, TF-IDF is computed by multiplying two quantities TF and IDF. TF (term frequency) measures how frequently a term appears in document, denoted as $tf_{t,d}$, where t represents the term and d the document. IDF, on the other hand, measures 'the amount of information' associated with a term [2]. The IDF of a term is calculated as the logarithm of the inverse of the document frequency, which refers to the number of documents containing the term t. The equation for the TF-IDF score for a term can be written as [35]:

$$\begin{aligned} tf.idf_{t,d} &= tf_{t,d} \times idf_t \\ &= tf_{t,d} \times log(\frac{N}{df_t}) \end{aligned} \tag{2.1}$$

2.1.3 Word2Vec

The Word2Vec approach, developed by Google in 2013 [84], uses neural networks [164] to learn word embeddings that capture the contextual relationships between words in a corpus. It consists of two models, each a shallow 2-layer neural network, namely skip-gram and continuous bag of word (CBOW) [186]. These models estimate the conditional probabilities of a word based on its surrounding words, encoding semantically meaningful numeric representations in a high-dimensional vector space [186]. Word2Vec models are trained on large text corpora, assigning each unique word a vector in a space typically spanning several hundred dimensions [63], learning an embedding matrix VxD, where V is the total number of unique words and D is the embedding dimension. The resulting embeddings group contextually similar words closely together, and dissimilar words farther apart [63].

Unlike traditional methods like BOW and TF-IDF, Word2Vec accounts for the order of words along with co-occurrences, generating more meaningful word representations. However, it has limitations. Word2Vec generates shallow, static embeddings for each word [87], which does not capture varying contextual relevance between entities [86]. Because a word can have multiple meaning depending on where it is used in the text sequence, static embedding algorithms such as Word2Vec fail to capture the context level information and higher level concepts like long-term dependencies and negation [87].

GloVe

Global Vectors (GloVe) [127] is a Word2Vec model, pretrained on a large English Wikipedia corpus, embededding each word in a latent vector space. For this study we use the GloVe6B model, where each word is represented by a 300-dimensional vector. Sentence-level embeddings are derived by averaging the vectors of all words in a sentence, producing a 300-dimensional feature vector for each instance.

2.1.4 Bidirectional Encoder Representations from Transformers (BERT)

To capture the contextual and positional information in texts, language models like Recurrent Neural Networks(RNN) [151], Long Short-Term Memory (LSTM) [151], Sequence-to-Sequence [158], and bidirectional LSTMs such as ELMO [130] have been developed. The introduction of transformer models in 2017 by Ashish Vaswami [166] led to the development of OpenAI GPT and BERT models. Among all models BERT has shown significant success achieving state-of-the-art performance

on eleven NLP tasks [41]. One of the key factors leading to BERT's success is that it uses a bidirectional training approach, which allows it to process text in both left-to-right and right-to-left directions during pre-training, unlike OpenAI GPT which only trains form left-to-right [87]. Thus BERT contextual embedding is able to capture the contextual meaning of words based on both the preceding and succeeding words in a sentence.

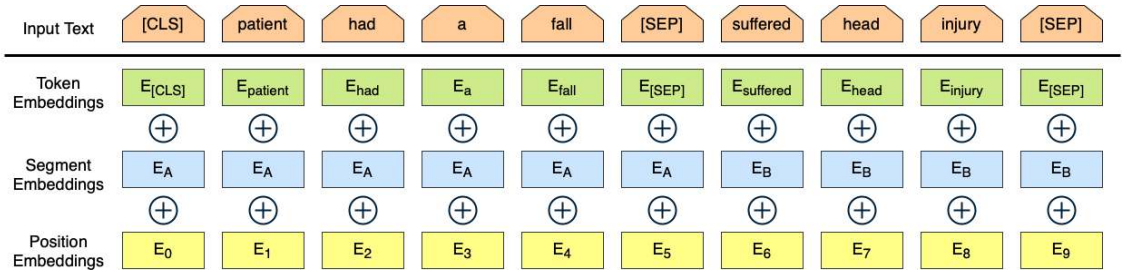


Figure 2.1: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings[41].

BERT's model architecture is a stacked multi-layer bi-directional encoder of original Transformer model [41, 166]. Each sequence begins with a special classification token ([CLS]), that captures the overall context of the text sequence [41]. The final hidden state of this token is used as the aggregate sequence representation for classification purposes. For sentence pairs (e.g., [Question, Answer] tasks) the separation token ([SEP]) is used to distinguish between the sentences. BERT can learn contextual sentence-level embeddings that are capable of capturing the syntactic and semantic relationships using distributed representation, outperforming static representations approaches [86].

2.1.5 BERT Variants

The success of BERT has inspired the development of several variants, aiming to refine or extend the original model's capabilities. Two such notable variants are RoBERTa (Robustly optimized BERT approach) and DeBERTa (Decoding-enhanced BERT with disentangled attention).

RoBERTa — A Robustly Optimized BERT Pretraining Approach:

Is a re-implementation of Google's BERT model, developed by researchers at Facebook AI [41, 106]. It aims to optimize key hyperparameters like learning rate, warmup steps, batch size, and input sequence length, as well as expanding training data size, to improve performance on various language-based tasks. Unlike BERT, which pre-trains on BooksCorpus (800M words) [189] and English Wikipedia (2,500M words), RoBERTa trains on a larger English-language corpus, incor-

porating CC-News, OpenWebText, and Stories, totalling over 160GB of uncompressed text [106]. This comprehensive training corpus enables the model to capture a wider range of linguistic contexts. RoBERTa model achieves state-of-the-art results on several major NLP benchmarks, including GLUE, RACE and SQuAD [106]. Due to its enhanced performance, RoBERTa is considered over BERT for text feature extraction in our experiments.

DeBERTa — Decoding-enhanced BERT with disentangled attention:

Is an improved version of BERT architecture developed by researchers at Microsoft. It introduces a novel disentangled attention mechanism, where each word is represented through two separate content and positional vectors, so that attention weights can be computed among words on both their content and positional information. Additionally, it incorporates an enhanced mask decoder that uses absolute positional information for accurately predicting masked tokens during pre-training [70]. This approach allows DeBERTa to better understand the relative positional information between words in a sentence, making it more effective in capturing the nuances of language structure. DeBERTa demonstrates superior performance on MNLI, SQuAD v2.0, and RACE NLP benchmarks, surpassing RoBERTa-Large models, with improvements of +0.9%, +2.3%, and +3.6% respectively [70]. The effectiveness of deberta on various NLP tasks like question answering, sentiment analysis, and text classification, as reported in various studies, makes DeBERTa an optimal choice for text feature representation [170, 188, 157].

Domain-Specific Contextual Representations

Inspired by BERT, domain-specific models like Med-BERT and Bio-BERT have been developed for healthcare applications. These models mainly differ in the dataset they were trained on. For instance, **Med-BERT** is trained on a large Electronic Health Record (EHR) dataset containing 28,490,650 patient records. Experiments have shown that Med-BERT substantially improves prediction accuracy and AUC (Area under curve) for small fine tuning tasks specific to clinic domain such as disease prediction, compared to other deep learning models [137]. **BioBERT** is another such domain-specific language model pre-trained on large biomedical corpus, including PubMed abstracts and PubMed Central (PMC) full-text articles [96]. Thus, BioBERT does well in understanding complex biomedical text and largely outperforms original BERT on biomedical text mining tasks such as biomedical named entity recognition, relation extraction, and question answering. [96].

2.2 Active Learning (AL)

In the modern digital era, vast amounts of data are generated, such as web documents, images, and emails, most of which are unlabeled. However, supervised machine learning models require a substantial amount of labeled data for optimal performance. The traditional practice of manually labeling training data is time-consuming, costly, and often impractical due to the immense volume of data available [61]. Active Learning (AL) is a semi-supervised machine learning technique that addresses this challenge by selectively querying a small subset of informative data points for human annotation, instead of labeling the entire dataset, and then uses these newly-labelled samples to re-train the model [53, 61, 118]. This process repeats for a set number of iterations or until reaching predefined criteria like a target classification accuracy or exhausting the labeling budget [97], iteratively improving the classifier performance. This AL cycle ensures that the selected data points, when accurately annotated, significantly enhance the machine learning model's performance [53]. Re-training of classifier using labeled samples can be done using two approaches: Batch, where the model is re-trained after labeling multiple examples, and Sequential, where retraining occurs immediately after each new element is labeled [118].

2.2.1 AL Sampling Strategy

There are three main sampling strategies for Active Learning in the literature [23, 118]:

- Pool-based sampling: This approach involves the active learner to determine the rank of each sample from the entire set or subset of unlabeled data (the "pool"), and selects the most informative sample based on a AL selection strategy (see Section 2.2.2) for manual labelling. This method is ideal for datasets that have a small set of labeled instances and a large set of unlabeled instances [23, 61].

- Stream-based selective sampling: Also known as sequential sampling, where each unlabeled instance is evaluated individually one at a time to decide whether to label it or not [23]. These decisions can be based on various methods, such as choosing to label if the AL selection strategy exceeds a certain threshold or by determining if instances fall within identified regions of uncertainty [23, 6]

- Membership query synthesis: Unlike pool-based or stream-based sampling, this approach allows the learner to request labels for any unlabeled instance in the input space. It can generate synthetic instances instead of selecting them from the dataset, which may lead to nonsensical

queries. For example, in MNIST handwritten digit classification, it might query the label for a non-existent digit (an artificial hybrid digit generated), which is not sensible even to humans [23].

2.2.2 AL Selection Strategy

The selection of data points to be labelled is done based on the chosen selection strategy, such as uncertainty, representativeness, or diversity [71]. Various selection strategies have been proposed in the literature. The strategies can be distinguished into three types: uncertainty-based, Query By Committee (QBC) and diversity-based.

- Uncertainty: Several methods apply the uncertainty sampling principal to select the unlabelled data points that have the largest classification uncertainty [71]. Some examples of uncertainty measures include "Least Confidence", "Margin" and "Entropy" [145].

- Query By Committee (QBC): In the Query By Committee (QBC) approach, a group of classifiers is used to identify and label the most informative examples. Each data point is evaluated for its potential informativeness, generally based on the level of disagreement among the committee of models regarding its predicted label. After labeling, these examples are added to the training set. This cycle continues until the number of available requests for labels is exhausted [115, 23].

- Diversity: An approach proposed by [71] which which provides a systematic way for measuring and combining the uncertainty, representativeness, and diversity of an instance. It first uses instances' uncertainty and representativeness measures to constitute the most informative set. Then, it uses kernel-means clustering algorithm to filter the redundant samples and the resulting samples are queried for labels [71].

Chapter 3

Learning to Rank PSE reports by Severity

3.1 Introduction

Patient safety events (PSEs) are incidents or circumstances, including medical errors, near misses, and adverse events, that could have or did result in unnecessary harm to a patient [172, 78]. Incident Reporting (IR) systems in healthcare are crucial for enhancing care quality and ensuring patient safety. It enables healthcare personnel (e.g. point-of-care clinicians), to voluntarily log incidents, in the form of structured and unstructured data such as, the occurrence and reporting date, location, event type, a narrative description of the incident, and an initial harm level assigned at the onset of the event by report initiator. These reports are then reviewed by hospital staff such as risk managers and Patient Safety Analysts (PSAs), playing a crucial role in improving healthcare quality and safety by providing insights into factors leading to PSEs and facilitating the development of preventive measures [73].

Provincial hospital management legislation mandates that healthcare organizations promptly review critical (i.e. high severity) incidents following their occurrence. These incidents must undergo a thorough analysis and a plan developed with systemic steps aimed at minimizing or preventing similar future incidents. In the hospital, PSAs are responsible for screening all reports in the IR system within two days to identify potential critical incidents. Timely and accurate classification of these reports is essential for enhancing the safety and quality of healthcare services, especially in identifying and mitigating Serious Safety Events (SSEs). However, due to the rarity of SSEs,

the manual triage process by PSAs to screen through a large volume of reports and isolate critical incidents is labor-intensive, time-consuming, and costly.

In this study, our objective is to utilize machine learning (ML) models that can differentiate and prioritize high severity events (Serious Safety Event (SEC) Levels 1-4; see Section 3.2) from less severe ones after being reported and prior to analysis. We adopt a binary ranking approach, implementing Learning-to-Rank (LTR) models (see section 3.8) that learn to give potentially severe events precedence over other events, appearing higher on the review list. Such prioritization will enable PSAs to quickly address these critical events, thereby increasing the effectiveness of post-event analyses. Additionally, we expect this method to decrease the number of non-severe events undergoing detailed reviews, streamlining the overall evaluation workflow. Specifically, we make the following contributions:

1. We leverage tree-based ensemble models, specifically Extreme Gradient Boosting (XGB) and Light Gradient Boosting Machine (LGBM), considering both query-independent and query-dependent Learning-to-Rank (LTR) strategies in ranking patient safety event reports by severity. To our knowledge, this is the first work to implement LTR models for PSE report classification. We further analyze the effectiveness of these models across various static and deep-learning based contextual text representations.

2. Our evaluation reveals that the query-independent LTR strategy using the LGBM classifier with healthcare domain-specific GatorTron features, significantly outperforms the existing baseline. This model can assist PSAs in efficiently identifying potential critical incidents, thereby reducing the workload of manual screening and streamlining the review process.

3. We demonstrate that domain-specific embeddings, such as GatorTron, BioGPT, and BioMed-BERT, predominantly surpass traditional NLP representations across ranking metrics. This underscores the benefits of utilizing healthcare-domain specific contextual embeddings to enhance classification accuracy in PSE analysis.

3.1.1 Organization

In Section 3.2, we introduce the widely adopted standard coding taxonomy for categorizing Patient Safety Events (PSEs) used by healthcare organizations. In Section 3.3, we discuss the current PSE review workflow employed by Patient Safety Analysts (PSAs) to triage critical safety events. Section 3.4 discusses relevant work from previous studies in the literature on using ML to classify

incident reports. Section 3.5 describes the specific dataset used in this study, including predictive fields and target labels. Section 3.6 discusses the exploratory data analysis of input features, their distributions, and relation to target labels. Section 3.7 details strategies for feature engineering and extraction, such as text cleaning and conversion. Section 3.8 outlines the methodology for model building and evaluation. Section 3.9 presents the results, while Sections 3.10 and 3.11 cover discussions, limitations, potential future work and conclusion.

3.2 HPI Safety Event Classification (SEC) Code

The widely used standardized Patient Safety Event Classification (SEC) taxonomy, developed by the Healthcare Performance Improvement (HPI), categorizes safety events into three broad types —Serious Safety Event (SSE), Precursor Safety Event (PrSE), and Near Miss Event (NME) [161]. The SEC extends beyond merely assigning an event type, unlike traditional categorical classifications (e.g. fall, or burn), by assessing deviations from generally accepted performance standards (GAPS), establishing a cause-and-effect relationship between these deviations and outcomes, and the resulting level of patient harm [161].

A patient safety event that does not involve a deviation from GAPS is considered "Not a Safety Event", as defined by HPI [161]. Events representing a deviation from GAPS are categorized into one of the three main types in the Safety Event Classification Hierarchy, each associated with different levels of harm:

1. **Serious Safety Event (SSE)**: These incidents represent deviations that reach the patient and can lead to outcomes varying from moderate to severe harm, including death. It is important to note that not all SSEs qualify as "sentinel events" per the Joint Commission's definition or as "never events" by the National Quality Forum (NQF) [161]. The distinction lies in that "sentinel events" can include incidents that do not result in harm to the patient (e.g. near misses or minor harm) or instances where the standard of care and practice expectations are met by the organization.

2. **Precursor Safety Event (PrSE)**: This category includes events that reach the patient but result in "minimal harm, no detectable harm, or no harm" to the patient [161].

3. **Near Miss Safety Event (NME)**: These events are deviations where the error is caught and corrected before reaching the patient, either due to a safety barrier or by chance [161].

Figure 3.1 illustrates the classification methodology defined in HPI White Paper Series [161], used to assign safety events into the three types. Additionally, HPI divides these broad SEC categories into sub-levels depending on the severity of harm, as detailed in Table 3.1.

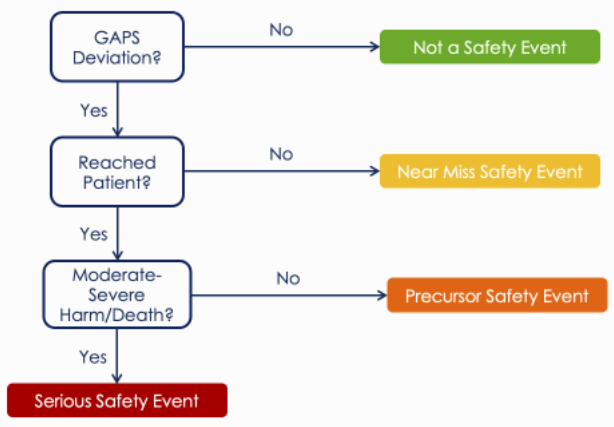


Figure 3.1: HPI SEC Algorithm [161]

Table 3.1: HPI SEC Levels of Harm [161]

HPI SEC	Code	Level of Harm
Serious Safety Event (SSE)	SSE 1	Death
	SSE 2	Severe Permanent Harm
	SSE 3	Moderate Permanent Harm
	SSE 4	Severe Temporary Harm
	SSE 5	Moderate Temporary Harm
Precursor Safety Event (PrSE)	PrSE 1	Minimal Permanent Harm
	PrSE 2	Minimal Temporary Harm
	PrSE 3	No Detectable Harm
	PrSE 4	No Harm
Near Miss Safety Event (NME)	NME 1	Unplanned Catch
	NME 2	Last Strong Barrier Catch
	NME 3	Early Barrier Catch

3.3 Triage of Critical Incidents (SSE Levels 1-4)

In the Incident Reporting (IR) system, incidents undergo post-event analysis by various teams within the organization. Patient Safety Analysts (PSAs) are tasked with reviewing preventable and severe harms, while other members are responsible for examining events with a lower level of harm. Specifically, PSAs are required to manually review a large number of reports submitted through the IR system daily. They are tasked with screening and prioritizing "potential Serious Safety Events (SSEs)" (i.e. events that appear to qualify for highest severity categories SSE 1-4), for further detailed analysis by the central quality and safety team. After this initial screening

by PSAs, these potential SSEs are subjected to an intensive review by a review team, which may include point-of-care clinicians, operational/medical/practice leaders, PSAs, or other subject matter experts. This review team identifies the underlying factors that contributed to the event in order to develop appropriate solutions. The review commitee meeting concludes with reaching a consensus on the Safety Event Classification (SEC) code (refer to Table 3.1), which reflects the confirmed severity level of the incident. This collaborative decision-making process is necessary due to the complexity of cases and the interpretation of the SEC definitions. Figure 3.2 shows this safety event classification workflow currently followed by the healthcare organization.

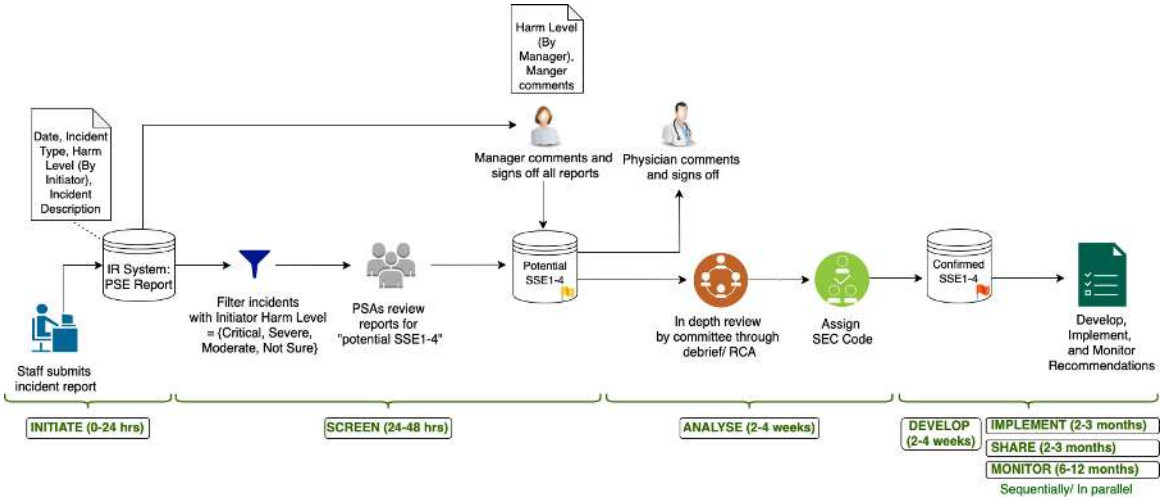


Figure 3.2: Safety Event Classification Workflow

This review process is typically conducted within 2 weeks to 3 months, depending on the complexity of the event. Due to this resource and time-intensive process, only a small fraction—approximately 3-4%—of safety events are accurately labeled with a SEC code after undergoing intensive review over a span of six years. The challenge is compounded by the relatively small number of highly severe incidents (SSE 1-4) among a large dataset of incidents, making it difficult for PSAs to pinpoint the critical cases that warrant comprehensive reviews. Of all the events flagged as "potential SSE 1-4" and subjected to in-depth review, about 9% are ultimately confirmed as SSE1-4 after these sessions, accounting to 0.38% of all reports collected over a six-year period. Figure 3.3 shows the proportion of incidents screened by PSAs as "Potential SSE1-4" for review and the subset of these screened events confirmed as SSE1-4 post review.

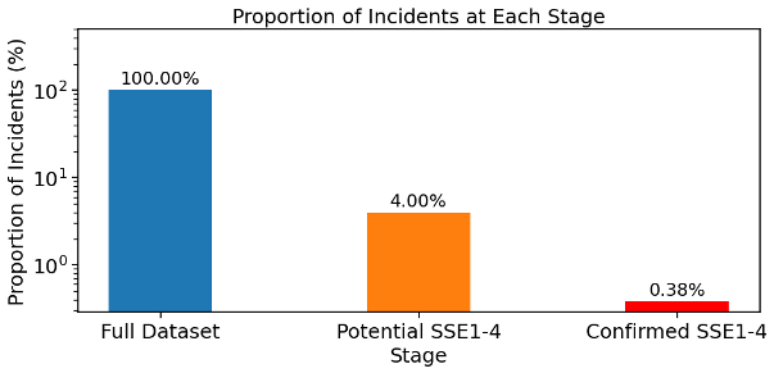


Figure 3.3: Incident proportions at each severe event screening stage

3.4 Related Work

This section discusses some of the relevant work in the literature, pertaining to the classification of medical reports like PSEs, and the results they obtained.

3.4.1 Traditional ML model to classify free text incident reports

Traditional Machine Learning (ML) models include Logistic Regression, Decision Trees, Support Vector Machines (SVM), Naive Bayes classifiers and others discussed in Chapter 2. There has been considerable attempts made to automate the classification of patient safety event (PSE) reports using traditional ML model among which SVM, Logistic Regression and Naive Bayes were most commonly used.

A paper published by Mei-Sing Ong et al in 2012, explored the feasibility of using statistical text classification to automatically detect extreme-risk events in clinical incident reports [123]. Naïve Bayes model and Support Vector Machines (SVM) text classifiers were developed to identify extreme-risk (severe) safety events based on the free-text description of the incident reports. Results show that SVM with a linear kernel performed best, with an accuracy in identifying severe events (SAC level 1) of 0.858 (0.95 Confidence Interval 0.796 to 0.921), precision=0.88, recall=0.83, F-measure=0.86, and AUC=0.92 [123].

In 2017, Ying Wang et al, evaluated SVM classifiers on linear and RBF kernels using the one-versus-one (OvsO) and one-vs-all (OvsA) emsemble strategies, to classify incident reports into 10 event types and 4 severity levels [172]. According to the results, OvsO ensembles of SVM RBF with binary count feature extraction (i.e. 1 if word appears 0 otherwise) were the most effective combination to identify incident type and severity level. For severity level, F-score for extreme risk event were much higher (0.873 for SAC 1 level) compared to low risk events (0.64 for SAC 4 level) on

balanced data. Overall, lower performance was observed with severity levels compared to incident types because each severity level included multiple incident types making it harder to obtain distinct vocabularies between levels [172].

A similar study conducted in 2020, by Huw Prosser Evans et al, also implemented Naive Bayes and SVM with polykernel to classify unstructured free text into incident type and severity [48]. Here again SVM reported highest AUROC (Area Under the Receiver Operating Curve) for both incident type classification and severity classification approximating to 0.84 and 0.71 respectively. Note this paper also reports lower performance on severity level compared to incident types [48].

A major issue with incident reporting in healthcare is that often classifying the incident can be complex and difficult to understand by frontline reporters, leading reporters to frequently classify incidents into the generic category such as "others" or "miscellaneous" as opposed to assigning a specific category thus impeding the very reason for incident reporting, which is to analyze incidents to improve patient safety and quality of care. The "Miscellaneous Patient Safety Event Reports" paper published in 2021 by Fong et al, attempted to address this issue by implementing various ML models such as LR, RF and SVM with RBF kernel, to reclassify the safety events that were classified as miscellaneous by frontline reporters [52]. Among all models the SVM with RBF kernel had the best F-score and accuracy of 0.89 and 0.92, respectively, mirroring the results of previous papers discussed.

Summarizing the findings, we see that SVM based models outperformed other traditional models in PSE text classification, in every study examined.

3.4.2 Advanced ML models to classify patient incident text

Although there has been several studies using traditional NLP techniques to automate the PSE report classification, the use of more advanced NLP techniques using neural network language models is limited [185]. More recent studies make use of contextual word embedding such as BERT—Bi-directional Encoder Representations from Transformer [41]. Contextual embedding takes into account not only the co-occurrence of words, but also the semantic meaning of the word based on other surrounding words as described in Chapter 2 section 2.1.4.

One such study [83] examines the effectiveness of pre-trained BERT embedding in medical code assignment tasks published in the year 2021 by Shaoxiong Ji et al [83]. The paper conducts a comprehensive quantitative analysis of various contextualized language models' performances, pretrained in different domains, for medical code assignment from clinical notes. The study investigated several

BERT variants pretrained on the following three domain categories:

1. Generic Domain: Vanilla BERT pre-trained on BooksCorpus and English Wikipedia
2. Mixed-domain pretraining combining closely related domains: A mixture of BERT variants that are closely related to the target clinical domain such as biomedical and health-related social domains were investigated. Both continued training and training from scratch were considered. Some BERT variants include:
    - BlueBERT- pretrained with PubMed text and MIMIC-III clinical notes [126]
    - BioBERT- continually pretrained on domain-specific data from PubMed abstracts and PMC full-text articles [96]
    - BioRedditBERT- initialized from BioBERT and continually pretrained on health-related posts from health-themed forums in Reddit [9]
    - PubMedBERT- domain-specific pre-trained from scratch in biomedical domain using PubMed publications [64]
3. In-domain continued pre-training: Continues pre-traning on the task specific ClinicalBERT in the second stage using clinical notes and discharge summaries.

Results from the study show that while pre-training in mixed domains improves predictive performance to some extent over the BERT-base pretrained in general domains, the PubMedBERT pretarined from scratch on biomedical corpora gains comparatively better performance with improvements of 0.039 and 0.033 for macro and micro F-scores on the MIMIC-III top-50 code set [83]. The explanation for this was that specific-domain pretraining makes downstream classifier concentrate on specified semantic knowledge as opposed to mixed-domain models that may be distracted from relatively broad information [83].

The study also investigates fine tuning all the BERT variants on custom dataset using Fully Connected Networks (FCN) and Label-wise Attention Network (LAN) (i.e connects document representation with label information) on top of each BERT variant [83]. It concludes that hierarchical fine tuning does improve prediction performance as well, where PubMedBERT with LAN outperforms all other fine tuned models, achieving 0.633 and 0.681 Macro and Micro F-scores respectively on MIMIC-III top-50 code set [83].

Although fine-tuning can enhance performance, this is not always the case. For example, the study indicates that fine tuning pre-trained language models struggles with high dimensional structured prediction tasks involving a full label set of over 8000 labels [83]. In cases where fine-tuning

does not enhance performance, utilizing pre-trained embeddings directly from transformer models as input features for traditional ML models can improve prediction accuracy. We plan to implement this approach by using various contextual embeddings, such as BERT, and traditional supervised ML models like SVM or LR to classify PSI reports.

Another study developed a deep Convolutional Neural Network (CNN) model with Word2Vec word embedding to identify 10 incident types and 4 severity levels of patient safety incident reports [171]. A CNN architecture with 6 layers outperformed SVMs, achieving F-scores as high as 0.85 across three test datasets when identifying common incident types (e.g. falls, pressure injury, and aggression) [171]. The CNN also outperformed SVMs, in common severity levels (medium SAC 3/low SAC 4) identification, improving F scores by 0.119 to 0.451 three across all three test datasets [171]. Although, for rarer incident types (e.g. blood products, infection, patient identification and clinical handover), the CNN on balanced dataset had relatively lower performance in real-world setting, it was able to better generalize on unseen datasets compared to SVMs [171]. While CNNs for sentence classification can yield promising results, their performance is highly dependent on the selection of different hyperparameters. These include filter size, the number of feature maps, activation functions, max pooling layers, and the application of regularization. Therefore, a careful and well-considered architecture is crucial for achieving optimal performance.

Approaches to combine well performing models to predict PSEs were also conducted. A study in 2019, by Wang et al, showed that combined models generally outperform individual models with some exceptions, where the best model identified Health Information Technology (HIT) events with approximately 0.90 accuracy and achieved approx 0.87 F-score [168]. A hybrid CNN and bidirectional long short-term memory (LSTM) has been used in the detection of bleeding events from electronic health record (EHR) notes with an overall F-score of 0.938 [99]. Models such as LSTM are particularly valuable when dealing with long input text documents, such as those found in EHR data, which contain an abundance of text. LSTMs are capable of capturing the long term dependencies within text, enabling them to extract the most relevant insights from lengthy sequence of text. Recent novel approaches to classify long documents, use BigBird sparse-attention based transformer models, such as the ICDBigBird model that combines Graph Convolutional Network (GCN) [92] and a contextual embedding model for International Classification of Diseases (ICD) classification task (i.e a widely used coding system maintained by World Health Organization) [116].

3.5 Dataset

This study utilized a dataset from the Patient Safety Event (PSE) reporting system of a large hospital in Canada, comprising 71,729 PSE reports over six years, detailing incidents that occurred from January 1st, 2017, to December 31st, 2022. The study was approved by the hospital's institutional review board (REB# 23-5237) and underwent additional reviews to ensure compliance with all privacy legislation and quality assurance privileges.

The dataset comprises twelve fields: Date of Birth, Date Incident Occurred, Date Incident Reported, Incident Type, Level of Harm (By Initiator), Level of Harm (By Manager), Incident Description, Physician Outcome (By Manager), Physician Comment, Responsible Manager Comments, Safety Event Code (SEC), and Event Type Code (ET).

Despite the large size of the dataset, several fields have significant proportion of missing (null) values, including "Date of Birth", "Physician Outcome (By Manager)", "Physician Comment", "Responsible Manager Comments", SEC, and ET codes. In the event reporting process, certain fields such as "Incident Type", "Level of Harm (By Initiator)", and "Incident Description" are required at the initiation of the report and therefore do not have null entries. However, additional information, including assessments by managers or physicians like "Level of Harm (By Manager)", "Physician Outcome (By Manager)", "Physician Comment", and "Responsible Manager Comments" may not be immediately available or remain unfilled depending on the event's severity. Table 3.2 provides the percentage of missing values for each field across the dataset.

Table 3.2: Percentage of null fields in the dataset

Field	**# Null values**	**% null of Dataset**
Date Of Birth	25430	35.45
Date Incident Occurred	0	0.00
Date Incident Reported	0	0.00
Incident Type	0	0.00
Level of Harm (By Initiator)	0	0.00
Level of Harm (By Manager)	0	0.00
Incident Description	0	0.00
Patient Outcome (Manager)	43975	61.31
Physician Comment	71383	99.52
Responsible Manager Comments	20028	27.92
Safety Event Code (SEC)	68746	95.84
RM Event Type (ET)	69288	96.60

3.5.1 Input Fields

When initiating an event report, fields such as "Incident Type", "Level of Harm (By Initiator)", and "Incident Description" are mandatory. Other fields can be completed later or left blank based on the event's severity. Consequently, during the screening for potential severe incidents (SSE1-4) by PSAs, information like "Level of Harm (By Manager)", "Physician Outcome (By Manager)", "Physician Comment", and "Responsible Manager Comments" is often unavailable as this information is usually entered after the review of an event. Hence, these fields are excluded from model development. Furthermore, we exclude temporal fields such as "Date of Birth", "Date Incident Occurred", and "Date Incident Reported", as they are not considered significant by domain experts in determining the severity of an incident.

Therefore, to distinguish high-risk severe incidents (SSE 1-4) from other levels, the model utilizes only three predictive fields: "Incident Type", "Level of Harm (By Initiator)", and "Incident Description." The free-text "Incident Description" field in the PSE reports was anonymized following data extraction to comply with privacy regulations.

3.5.2 Target Field

The dataset has 14 Safety Event Codes (SEC) that are assigned following the in-depth review, including Serious Safety Events (SSE 1-5, SSE TBD), Precursor Safety Events (PrSE 1-4), Near Miss Events (NME 1-3) and Not a Safety Event (NSE). Details on these levels are discussed in Section 3.1. The category "SSE TBD" refers to Serious Safety Events for which the level of SSE has not yet been determined. "Not a Safety Event" denotes events where there was no deviation from generally accepted performance standards (GAPS) followed by healthcare organizations.

Among the 14 SEC codes, only the first four—SSE1, SSE2, SSE3, and SSE4—are deemed extreme risk severe safety events that require thorough review and are prioritized by the healthcare organization. These events represent critical incidents that necessitate in-depth reviews, such as debriefs or Root Cause Analysis (RCA), and the development of plans to prevent such incidents in the future, as mandated by provincial law. Therefore, our analysis focuses solely on identifying these critical SSE1-4 events from the rest, framing it as a binary classification task. The target label is a binary variable: '1' indicates a severe event (SSE1-4) representing the positive class, and '0' represents all other events (including SSE5/TBD, PrSE, NME, and NSE) or those without a SEC code, which were not identified as potential SSE1-4 events by PSAs, and thus belong to the negative class.

Dealing with null SEC codes

The SEC and Event Type (ET) codes, assigned after comprehensive review, have the highest frequency of null values—approximately 95.84% for SEC and 96.6% for ET. This is expected since, as previously mentioned, only a small fraction—approximately 4%—of all events screened as "potentially severe (SSE1-4)" undergoes a detailed review, after which the appropriate SEC is assigned (see Figure 3.3).

Supervised machine learning typically uses only the labeled portion of data, often discarding unlabeled samples from model building. However, in this context we cannot simply discard all events with null values in the SEC field. This is because these events are indicative of incidents that were not flagged by PSAs as severe incidents requiring comprehensive review by the central quality and safety team—therefore lacking a SEC code—and thus considered non-severe incidents to be reviewed in a different process. Furthermore, relying solely on labeled data points with a SEC code for classifier training would restrict our model to recognize only "potentially severe (SSE1-4)" incidents, thereby overlooking the substantial volume of non-severe events that regularly occur in the Incident Reporting (IR) system. Hence, treating all unlabeled SEC events as non-severe allows us to fully leverage the dataset, ensuring a more accurate representation of the real-world data imbalance. Consequently, a label of '0' is assigned to all events without a SEC code, including these events in the training of the ML models. Table 3.3 shows the distribution of SSE1-4 versus non-SSE1-4 incidents, listing the specific SEC codes categorized under each class and the corresponding binary labels.

Table 3.3: Distribution of critical and non-critical incidents over 6 years

From January 1, 2017 to December 31, 2022

SEC Code	**Binary Label**	**# of Samples**	**% of Data**
SSE 1-4	1	273	0.38
SSE 5/TBD PrSE 1-4 NME 1-3 Not a Safety Event No SEC Code	0	71456	99.62

3.6 Exploratory Data Analysis (EDA)

This section analyzes the input fields we use for model building including the "Incident Type", "Level of Harm (By Initiator)", and the free-text "Incident Description" fields, and it's relation with the target binary SSE1-4 labels. Additionally, it discusses how these features were pre-processed and transformed into numeric representations for model building.

3.6.1 Incident Type

This field specifies the type of the incident based on predefined categories, chosen by the reporter at the time the incident is initiated in the IR system. The dataset has 13 incident types, with "Fall", "Miscellaneous", "Medication", "Operative/Invasive", and "Treatment" incidents being among the top five most frequently occurring incidents. Table 3.4 shows the distribution of all incident types, with decreasing order of frequency.

Table 3.4: Frequency of Incident Types

Incident Type	Count	% of Data
Fall	14789	20.62
Miscellaneous Incident	13563	18.91
Medication Incident	12682	17.68
Operative/Invasive Incident	7195	10.03
Treatment Incident	5474	7.63
Laboratory Incident	4944	6.89
Equipment Incident	4610	6.43
Privacy Incident	2939	4.10
Infection Control	2702	3.77
Hazard Incident	1407	1.96
Transfusion Incident	1016	1.42
Workplace Violence	245	0.34
Implanted Device Incident	163	0.23

Figure 3.4 displays the count matrix for all incident types alongside their corresponding frequencies of being assigned a severe (SSE1-4) code. The three incident types with the highest numbers of SSE1-4 classifications are "Treatment", "Infection Control", and "Operative/Invasive Incident". Additionally, it is observed that incidents categorized as "Hazard", "Privacy", and "Workplace Violence" are never assigned a SSE1-4 code. It is also important to note that these categories are among the bottom six in terms of frequency, thus occurring much less frequently than other types, with approximately only 200 to 3,000 incidents reported over a span of six years.

3.6.2 Level of Harm (By Initiator)

When reporting a safety event, the report initiator is required to choose a level of harm that most accurately describes the incident. This selection may vary based on the initiator's profession and their understanding of the classification taxonomy and patient outcome at the time of report. The available levels include "Critical", "Severe", "Moderate", "Not Sure", "Minor", "Near Miss/Potentially Severe", and "Near Miss". The frequency of each level is detailed in Table 3.5.

Currently, the harm levels assigned by initiators are the primary screening mechanism for PSAs

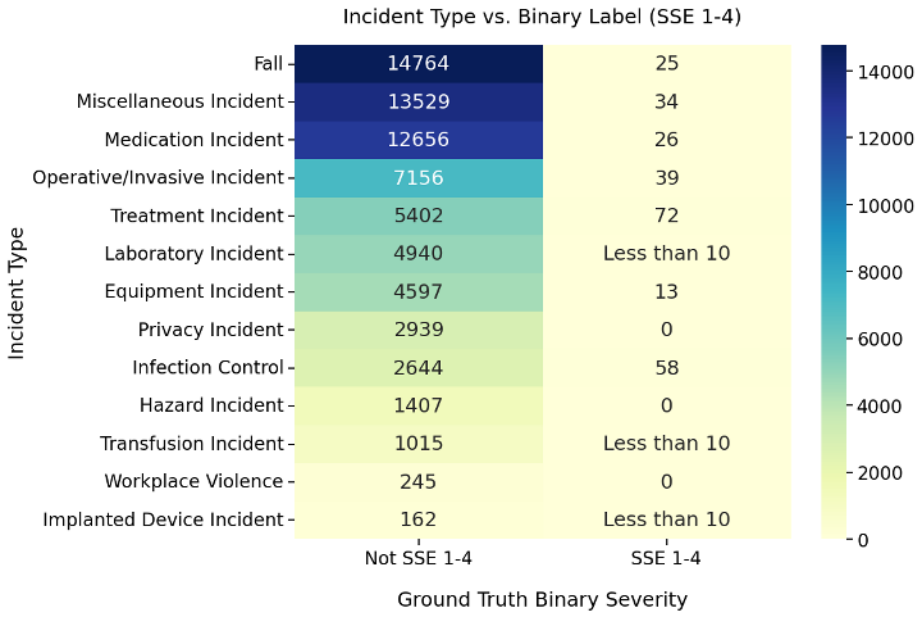


Figure 3.4: Distribution of Incidents by Type and Severity

Table 3.5: Frequency of Level of Harm (By Initiator)

Level of Harm (By Initiator)	Count	% of Data
Critical	111	0.15
Severe	1483	2.07
Moderate	7978	11.12
Not Sure	5046	7.03
Minor	36680	51.14
Near Miss/Potentially Severe	180	0.25
Near Miss	20251	28.23

to detect potential SSE incidents. Among these seven levels, PSAs concentrate on the first four — "Critical", "Severe", "Moderate", and "Not Sure" — to identify "potential SSE 1-4". This approach reduces the pool of events to approximately 20.6% of the entire dataset, accounting to 14,798 events, thereby narrowing the range of events for the central quality and safety team to review. However, this volume for PSA's to review is still high and resource intensive. To improve this process, we aim to develop classification and ranking algorithms that will further reduce this number, making the review process more efficient and allowing PSAs to concentrate on a smaller, more manageable subset of reports. This approach can significantly ease the PSA's workload and speed up the identification of severe events by reducing the time spent on screening through non-severe incidents, for which they are not responsible.

Figure 3.5 shows the count matrix comparing initiator harm levels against the binary SSE1-4 label. It shows that SSE1-4 incidents occur least frequently at the "Minor", "Near Miss", and "Near

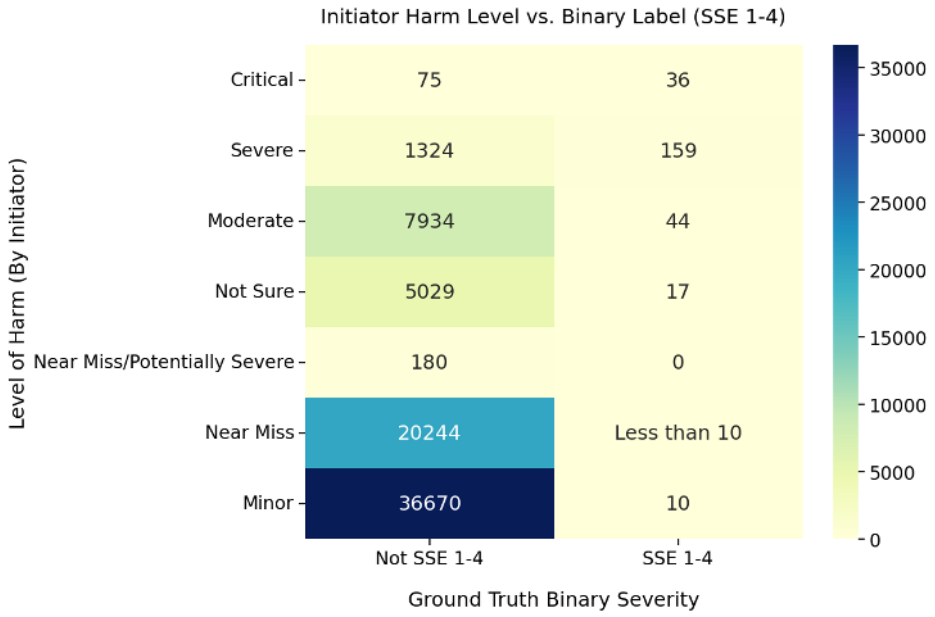


Figure 3.5: Distribution of Incidents by Harm Level (By Initiator) and Severity

Miss/Potentially Severe" initiator harm levels. From Chapter 1, we understand that the initial event type and severity assigned by the reporter can be inaccurate or misclassified due to the complexity of the safety event classification taxonomy and the subjective nature and variability of interpreting the event, particularly by those who may not be domain experts. Therefore, events initially assigned a non-severe harm level can sometimes be reclassified as confirmed SSE1-4 upon further review. This highlights an opportunity to enhance the PSA's initial screening process, which currently focuses on the first four levels, to ensure more comprehensive detection of severe events. The ML models can help address this limitation by considering all harm levels and learning the pattern that SSE1-4 incidents are predominantly associated with "Critical", "Severe", or "Moderate" harm levels and less frequently with other levels.

3.6.3 Incident Description

The "Incident Description" field in PSE reports is a narrative free-text area where initiators detail the patient safety incident. The length and format of these descriptions vary significantly. While some narratives are extensive, elaborating on the sequence of events in detail, others may be as brief as two words, such as "COVID 19".

Figure 3.6 illustrates the distribution of word lengths in Incident Descriptions across the entire dataset. We observe that the descriptions can range from 1-800 words, with most incidents having less than 200 words. This indicates that the texts are generally brief, suggesting that a token length

of 512 —standard for most Language Models (LMs)— is adequate for embedding these texts with little to no truncation required.

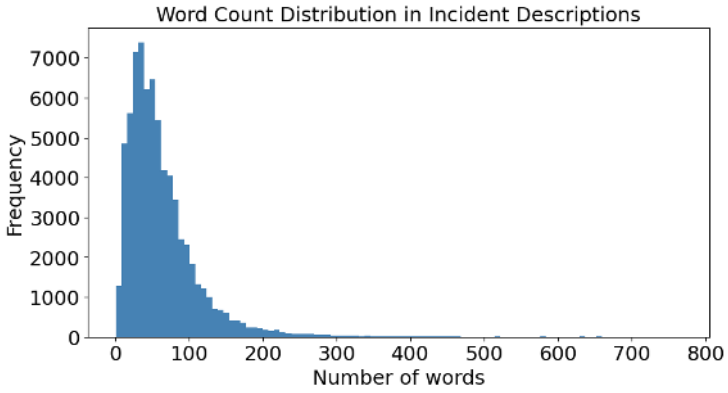


Figure 3.6: Distribution of Incident Description Length (in words)

3.7 Feature Engineering

In the first step of model building we perform feature extraction, converting the input fields into corresponding numeric vectors. These vectors can them be used for training the learning-to-rank XGB and LGBM model for the ranking process (see Figure 3.7) in the second step. This approach differs from fine-tuning, where the same model directly processes the text and predicts outcomes.

We encode the "Incident Type" and "Level of Harm (By Initiator)" categorical features as one-hot vectors [132]. Each category is represented as a boolean vector, where the vector length equals the number of unique category values and contains a single one indicating the active category. For example, a "Fall" incident type is represented by a 13-dimensional vector with zeros except for the position indicating a Fall. Similarly, harm levels are represented with a 7-dimensional vector. These vectors are concatenated to form a 20-dimensional vector for each event, capturing both type and harm level.

The incident description text field undergoes several pre-processing steps, including text cleaning and feature engineering, to extract both static and contextual features. Detailed descriptions of these techniques are provided below.

3.7.1 Text Cleaning

The incident description texts were first anonymized by removing personal identifiable information, to comply with privacy guidelines. Additionally, frequently occurring acronyms (e.g., "pt" for pa-

tient, “rn” for registered nurse, and “dr” for doctor) were expanded to their full forms using regular expression and pattern matching techniques [55] to improve text quality, along with removing excess whitespace within the texts.

For incident text feature extraction step, we explored both static and contextual NLP representations. Static representations, including Bag-of-Words (BOW), TF-IDF, and Word2Vec models (e.g., Glove), rely on the occurrence and frequency of words or tokens in the text. In contrast, contextual embeddings, including BERT and healthcare domain-specific models like BioGPT, consider not only word frequencies but also the context and positional information of words within a sentence. Detailed explanations on the workings of these text representations are provided in Chapter 2 section 2.1.

While additional text cleaning methods like lowercasing, word normalization, and stopword removal can enhance static features, it may degrade contextual representations by distorting context or meaning. Hence, we avoid further pre-processing for contextual embeddings. However, for static representations, we perform additional cleaning steps such as lowercasing, removing symbol, stopwords, non-alphanumeric characters and numbers, followed by lemmatization using SpaCy. This approach ensures that word vectors represent only relevant words and unify different forms of the same word into a single feature.

3.7.2 Text Feature Representations

In the text feature extraction stage, we investigate various NLP models including three static representations, three generic contextual representations, and three healthcare domain-specific contextual models. Here, we briefly mention each of the text representations used and provide explanations for how we extract them. For background on these representations refer to Chapter 2 section 2.1.

Static Representations:

1. **Count Binary (CB)**: Is a variant of BOW [133] model that represents each document as a binary vector of v dimension (vocabulary size), indicating the presence or absence of words, ignoring their order and frequency.

2. **Term frequency–inverse document frequency (TF-IDF)**: Weighs a term's frequency within a document against its inverse frequency across all documents [113], highlighting words that are frequent in a document but rare across the corpus.

   Note: The Count Binary and TF-IDF vectorizers tokenize text into unigrams, bigrams, and

trigrams, and converts all zero-frequencies to 1. To control vocabulary size, words appearing in 90% of documents or only in one document are excluded, and the maximum feature limit is set to 1 million for efficiency (originally was 1.8 million). Due to significant class imbalance, for feature selection stage, we balance the dataset by oversampling the positive class through random resampling, ensuring that both classes are accurately represented in the vocabulary.

3. **Global Vectors (GloVe)**: A Word2Vec [84, 127] approach that uses neural networks to create word embeddings from a large corpus like English Wikipedia. We use GloVe6B model that represents each word with a 300-dimensional vector and average them to retrieve sentence-level embeddings.

General Contextual Representations:

1. **Robustly Optimized BERT Pretraining Approach (RoBERTa)**: Is a re-implementation of Google's BERT model, developed by researchers at Facebook AI [41, 106]. It optimizes key hyperparameters, training on an larger corpus to achieve superior performance on multiple NLP benchmarks. Due to its improved performance, RoBERTa is considered over BERT for text feature extraction in our experiments.

2. **Decoding-enhanced BERT with Disentangled Attention (DeBERTa)**: DeBERTa [70], developed by Microsoft, enhances the BERT architecture with a novel disentangled attention mechanism and an enhanced mask decoder, improving performance on NLP benchmarks and making it highly effective for various text analysis tasks. It computes attention weights among words on both their content and positional information, and so can better understand the relative positional information.

3. **BGE M3-Embedding**: Developed by Beijing Academy of Artificial Intelligence (BAAI), BGE is capable of processing texts of varying lengths, from short sentences to long documents up to 8192 tokens, and excels particularly in multi-lingual retrieval tasks [31]. We use the bge-large-en-v1.5 English-specific model to retrieve the sentence embeddings for computational limitations.

Domain-Specific Contextual Representations:

1. **BiomedBERT**: BiomedBERT, initially known as PubMedBERT, is a domain-specific BERT model pre-trained on PubMed abstracts and PubMedCentral articles, achieving state-of-the-art results in biomedical NLP tasks [64].

2. **GatorTron**: GatorTron is a large clinical transformer model, trained on a diverse dataset including clinical notes, electronic health records, PubMed articles and MIMICIII texts. It outperforms existing clinical and biomedical models, including ClinicalBERT, BioBERT, and BioMegatron, across several clinical NLP tasks [183].

3. **BioGPT**: The BioGPT model, proposed by Luo et al. [110], is based on OpenAI's GPT language model, but is pre-trained from scratch for biomedical text generation and mining, using 15 million PubMed abstracts.

All contextual text representations were obtained from the HuggingFace library. Due to computational constraints, we did not fine-tune the transformer models for our specific downstream task, and instead used embeddings from pre-trained models. We set the maximum input token length for all models at 512, which is standard and sufficient for our dataset (see section 3.6.3). To derive the sentence embeddings, we performed mean-pooling on the output token embeddings. This approach, utilizing the average of all hidden layers, yielded slightly better results than using only the CLS token, which represents the final hidden state.

3.8 Model Development

For this study we utlize supervised 'learning-to-rank' (LTR) ML models, that can learn to sort documents by their relevance using labelled training data [105]. For detailed explanation on LTR models see Appendix B. Model inputs consisted of concatenated one-hot vectors for "Incident Type" and "Level of Harm (By Initiator)" combined with the various text representations of "Incident Description" as detailed in section 3.7.2.

Advancements in LTR, particularly through listwise methods and ensemble models, have significantly shown to improve ranking tasks [98, 19, 33, 24, 105, 159]. Therefore we choose these models for our study, leveraging their computational efficiency and capacity to handle complex, large-scale information retrieval tasks effectively. We implemented two powerful tree-based GBM ensemble models — Extreme Gradient Boosting (XGB) [34] and Light Gradient Boosting Machine (LGBM) [89]. For details on ensemble see Appendix A.4. These models are preferable due to their robustness in handling large datasets efficiently and particularly effective in managing the high dimensionality and sparsity typical of text data. [4]. While XGB is known to provide more accurate results, LGBM often facilitates faster convergence achieving comparable performance [10, 4, 89].

We implement both supervised ML ranking and classification models to solve the LTR task and

predict rank scores for binary severity labels. For details on query-dependent rankers, and query-independent classifiers, see the respective sections in Appendix B.3 and B.4. Figure 3.7 illustrates the proposed model development architecture for Safety Event Classification. The following sections provide implementation details and the methodology for model building, data splitting and data augmentation, hyperparameter tuning, and the rank-based evaluation metrics.

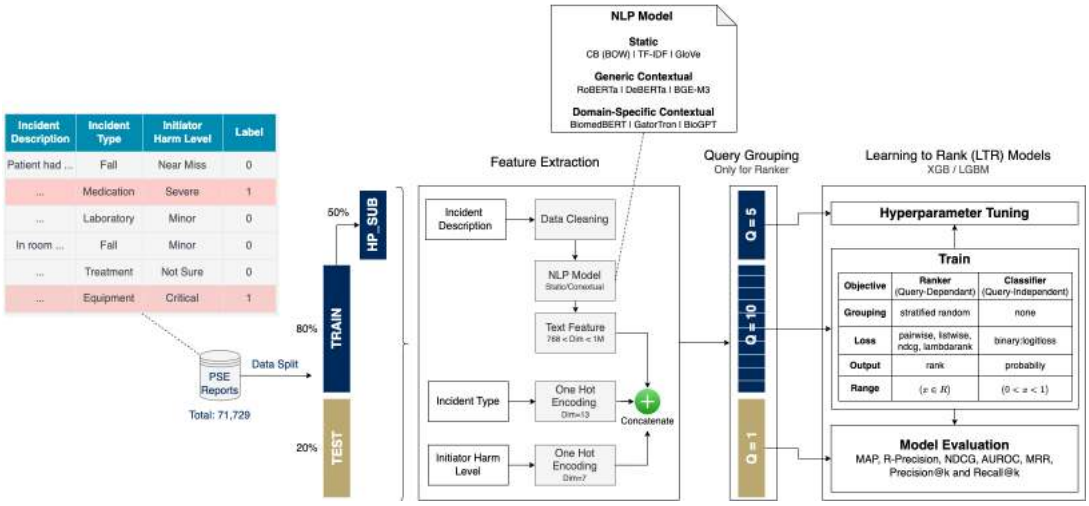


Figure 3.7: SEC Model Architecture

Note: HP_SUB is the subsample from the train set used for hyperparameter tuning. Q denotes the number of query groups in each set.

3.8.1 Ranking Models

In Information Retrieval, ranking models aim to sort a list of items in terms of their relevance to a query that represents an "information need" [113]. In this study, the information retrieval task focuses on retrieving rare severe events from a large set of PSE reports; therefore, the documents are not naturally grouped into queries for ranking. To facilitate the use of ranking models, we randomly assigned instances in the training set into 10 unique groups, ensuring a stratified split. This stratification ensures a similar distribution of binary severity labels and 'Level of Harm (By Initiator)' across all groups. However, to evaluate the learned ranking models, we treated the entire test set as a single query or group, avoiding random grouping. This approach allows us to assess the models' performance in ranking all available documents more effectively and eliminates any biases introduced by random grouping.

3.8.2 Classifier Models

Most algorithms for Learning to Rank (LTR) in Information Retrieval are considered to be query-dependent models [105]. However, several studies have also explored the use of query-independent classification models for relevance ranking in information retrieval, such as [36, 58, 119, 20, 187], which distinguish between classes directly from the input features without considering any query/topic-related information. It is therefore worthwhile to investigate whether such classifiers can achieve retrieval performances comparable to or better than those of LTR query-dependent ranking models. To assess the performance of XGB and LGBM rankers relative to their classifier counterparts, we use the output probability scores of the positive class from XGB and LGBM classifiers as the rank scores.

3.8.3 Data Splitting and Data Augmentation

We allocated 80% of the data for training and 20% for testing, applying a stratified train-test split based on both the binary target label and "Level of Harm (By Initiator)" to ensure representative distribution across all harm and severity levels. Additionally, we set aside a 50% stratified sub-sample of the training set, for hyper-parameter tuning to save time and computational resources.

Various dataset balancing techniques were tested to enhance model performance, but due to significant imbalances, methods like bootstrapping, SMOTE [27], and SMOTE-NC (Synthetic Minority Over-sampling Technique for Nominal and Continuous data) [60] caused overfitting and reduced performance. Similarly, undersampling led to substantial information loss, also diminishing performance. Therefore, data augmentation was not applied, and the original distribution was maintained for training.

3.8.4 Hyperparameter Tuning

A hyperparameter is a variable associated with a ML model that can be adjusted directly by the user during the training phase [91]. Each ML model may have a distinct set of hyper-parameters. The optimal values for these parameters vary depending on the dataset and task, and can be found using hyperparameter optimization (HPO) methods such as grid search or random search [150]. We use grid search with 5-fold stratified cross-validation on a sub-sample of the training set, specifying a range of values for each hyperparameter. Grid search explores the given subset of the hyperparameter space, trying every possible combination [103].

Given the large size of the dataset with over 71,000 samples, performing grid search with cross-

validation on the entire 80% training set would be time-consuming. Therefore, to save time and computational resources, we perform grid search on a 50% stratified subsample of the train set instead (see HP_SUB in 3.7), averaging the results over 5 folds. To facilitate the use of ranking models (see Section 3.8.1), the instances in the subsample are randomly assigned into 5 unique query groups, ensuring a stratified split. During 5-fold grid search cross-validation, one group serves as the validation set in each iteration, while the remaining four groups form the training set. The hyperparameters are optimized based on the highest average Normalized Discounted Cumulative Gain (NDCG) [82, 173] achieved across the 5 folds.

We implemented two ranking models, XGBRanker and LGBMRanker, and two classification models, XGBClassifier and LGBMClassifier, with the objective of ranking severe events with higher scores. Table 3.6 displays the hyperparameters and their ranges for each model. These ranges were determined by manually testing different parameter values to identify reasonable bounds and by incorporating common ranges specified in the literature. It is crucial to balance the number of parameter combinations with the available computational time, ensuring each model completes within a reasonable timeframe. Given that rankers generally train faster than classifiers, and LGBM is faster than XGB, we allocated more extensive parameter space to faster models while limiting the combinations for those requiring more time.

Along with tuning general tree-based hyper-parameters, parameters specific to ranking task such as the rank loss (e.g. "rank:ndcg", and "rank:map" and "rank:pairwise") and pair generation strategy ("mean", "topk") provided by XGBRanker, were also optimized for each feature representation (see Appendix B.5).

Table 3.6: Model Hyperparameters Tested with GridSearch

Model	Hyperparameter	Values
XGB Ranker	objective	['rank:map', 'rank:ndcg', 'rank:pairwise']
	lambdarank_pair_method	['mean', 'topk']
	lambdarank_num_pair_per_sample	[None, 10, 100, 1000]
	gamma	[0, 0.001, 0.01, 0.1, 0.3]
	subsample	[1, 0.7]
	n_estimators	[100, 200]
	enable_categorical	True
	seed	42
LGBM Ranker	objective	lambdarank
	boosting_type	['rf', 'gbdt', 'dart']
	n_estimators	[100, 200]

Table 3.6 continued from previous page

Model	Hyperparameter	Values
	max_depth	[-1, 3]
	learning_rate	[0.1, 0.05]
	subsample	[1, 0.7]
	reg_lambda	[0.0, 0.005, 0.1, 1]
	num_leaves	[31, 251]
	subsample_freq	[0, 5]
	force_col_wise	True
	class_weight	None
	random_state	42
	For boosting_type = 'rf':	
	bagging_freq	1
	bagging_fraction	0.75
	feature_fraction	0.75
XGB Classifier	objective	'binary:logistic'
	eval_metric	'logloss'
	booster	'gbtree'
	eta	[0.1, 0.3]
	subsample	[1, 0.7]
	lambda	[1, 0.001, 0.1]
	n_estimators	[100, 200]
	scale_pos_weight	[1, 250]
	max_depth	6
	enable_categorical	True
	seed	42
LGBM Classifier	objective	'binary'
	boosting_type	['rf', 'gbdt', 'dart']
	n_estimators	200
	max_depth	[-1, 3]
	learning_rate	[0.1, 0.05]
	reg_lambda	[0.0, 0.005, 0.1, 1]
	num_leaves	[31, 251]
	scale_pos_weight	[1, 250]
	force_col_wise	True
	random_state	42
	For boosting_type = 'rf':	
	bagging_freq	1
	bagging_fraction	0.75

feature_fraction	0.75

3.8.5 Initiator Harm Level Baseline

Currently, in the healthcare organization, Patient Safety Analysts (PSAs) use the "Level of Harm (By Initiator)" field to identify potential SSE1-4 for in-depth reviews with the central quality and safety team, analyzing only the first four levels— "Critical", "Severe", "Moderate", and "Not sure" (see section 3.6.2). To benchmark our Learning To Rank (LTR) models, we use the "Level of Harm (By Initiator)" field as the baseline, ranking the harm levels from most to least severe as "Critical", "Severe", "Moderate", "Not sure", "Near Miss/Potentially Severe", "Near Miss", and "Minor". For calculating baseline metrics, we convert each level into an integer that represents its severity, from 0 representing "Minor" to 6 representing "Critical".

Since baseline rankings are discrete, the ranks within the same harm level can not be determined. For instance, within the "Moderate" harm level category we cannot rank incidents from most to least severe. To address this, we shuffle incidents within each harm level subgroup, repeating this process 100 times, and averaging metrics across these iterations. This method acts like a tie-breaking mechanism and ensures stable and consistent metrics across multiple runs. Table 3.7 presents the train and test baseline metrics, averaged over 100 repeats.

Table 3.7: Baseline Performance

Model	**Metric**	**Train**	**Test**
Baseline	R-Precision	0.1958	0.197
	NDCG	0.6766	0.613
	AUROC	0.9311	0.9392
	MAP	0.1447	0.1546
	MRR	0.5031	0.5351
	p@20	0.3245	0.316
	p@50	0.3256	0.2042
	p@100	0.3007	0.1542
	p@200	0.2044	0.1316
	p@500	0.1457	0.0798
	p@1000	0.1264	0.0429
	r@20	0.0296	0.117
	r@50	0.0743	0.1891
	r@100	0.1373	0.2856
	r@200	0.1867	0.4876
	r@500	0.3326	0.7387
	r@1000	0.577	0.7941

3.8.6 Model Evaluation Metrics

To evaluate our models, we use rank scores or probabilities to order the list of events, prioritizing the most severe incidents (see sections B.3 and B.4). Ranking tasks in Information Retrieval are typically evaluated on a query basis [105], measuring how well documents are ordered in response to a specific query (e.g., the search results for "basketball 2024"). However, in our context, Patient Safety Analysts focus solely on identifying severe events, and thus our dataset does not naturally fall into distinct query groups since the topic of interest is constant. Hence, we treat all samples in a set as part of a single query group for evaluation purposes, ranking all documents collectively. Note however, for the training phase of query-dependent ranking models, which rely on grouping for effective learning, we divide the training set into 10 random stratified query groups (see section B.3). This approach allows us to manage the large dataset effectively, enabling rankers to fine-tune their performance within each group using various ranking objectives.

We evaluate models using several rank-based metrics on both train and test sets to check for potential overfitting and assess generalization performance. These metrics include Precision@k and Recall@k for $k = 20, 50, 100, 200, 500, 1000$, RPrecision, Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG).

To compare the performance of ranking algorithms with classifiers, we also compute the Area Under the Receiver Operating Characteristic (AUROC) using rank scores. Because AUROC measures a model's ability to distinguish between classes for every possible threshold, this method is threshold-independent and has been shown to be equivalent to the Wilcoxon statistic rank test [77, 66, 65]. Therefore, we can directly compute AUROC from rank scores produced by ranking models.

Here we list the mathematical equations and definitions for each metric:

1. **Precision@k**: Measures the proportion of retrieved documents that are relevant among the top k items [113].

$$\text{Precision@k} = \frac{\#\text{ (relevant items retrieved in top } k)}{k} \tag{3.1}$$

2. **Recall@k**: Measures the proportion of relevant documents that are retrieved among the top k items [113].

$$\text{Recall@k} = \frac{\#\text{ (relevant items retrieved in top } k)}{\text{total number of relevant items}} \tag{3.2}$$

3. **R-Precision**: Is a variant of precision where k is set to the total number of relevant items

denoted as R [113].

$$\text{R-Precision} = \frac{\#\text{ (relevant items retrieved in top } R)}{R} \tag{3.3}$$

4. **Mean Average Precision (MAP)**: Average Precision (AP) averages the precision at each position where a relevant document is retrieved. MAP is the mean of these AP scores across all queries. When all documents belong to one query, MAP is equivalent to AP [113, 88].

   Average Precision (AP) can be defined as:

$$\text{AP} = \frac{\sum_{k=1}^{n}(\text{Precision@k} \times \text{rel@k})}{\text{total number of relevant items}} \tag{3.4}$$

   where $rel@k$ is an indicator function that equals 1 if the item at rank k is relevant, and 0 otherwise.

   Mean Average Precision (MAP) is then:

$$\text{MAP} = \frac{\sum_{q=1}^{Q} \text{AP}_q}{Q} \tag{3.5}$$

   where Q is the total number of queries. In our case $Q = 1$.

5. **Mean Reciprocal Rank (MRR)**: MRR is a statistical measure used for evaluating the performance of a query-response system, where the interest is in the rank of the first relevant or correct response [105].

   Reciprocal Rank (RR) is defined as [152]:

$$RR_q = \begin{cases} 0 & \text{if no relevant results} \\ \frac{1}{rank_q} & \text{otherwise} \end{cases} \tag{3.6}$$

   where $rank_q$ is the rank position of the first relevant document for the q-th query.

   Mean Reciprocal Rank (MRR) is then:

$$MRR = \frac{1}{Q}\sum_{q=1}^{Q} \text{RR}_q \tag{3.7}$$

   where Q is the total number of queries.

6. **Area Under the Receiver Operating Characteristic Curve (AUROC)**: AUROC is a

metric used to evaluate the ability of a classifier to distinguish between classes. It is defined as the area under the ROC curve, which plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings [177].

True Positive Rate (TPR), or sensitivity, is calculated as:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.8}$$

where TP is the number of true positives and FN is the number of false negatives.

False Positive Rate (FPR), or 1-specificity, is calculated as:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3.9}$$

where FP is the number of false positives and TN is the number of true negatives.

AUROC is the integral [74]:

$$\text{AUC} = \int \text{TPR}\, d(\text{FPR}) \tag{3.10}$$

7. **Normalized Discounted Cumulative Gain (NDCG)**: NDCG is a ranking metric derived from the Discounted Cumulative Gain (DCG) which sums the graded relevance of ranked documents weighted by the discount function that decreases with the document's position in the ranked list [184]. NDCG normalizes DCG by dividing it by the Ideal DCG. NDCG values range from 0 to 1, where 1 indicates perfect ranking. This metric is advantageous as it can handle multi-level relevance and prioritizes documents higher on the list, acknowledging that top-ranked documents are more likely to be viewed.

   Standard NDCG with logarithmic discount function is defined as:

$$DCG = \sum_{i=1}^{n} \frac{\text{rel}(x_i)}{\log_2(i+1)} \tag{3.11}$$

   where n is the number of documents, and $\text{rel}(x_i)$ represents the relevance function that orders the list of documents by their relevance upto position i.

   NDCG is derived from DCG:

$$NDCG = \frac{DCG}{IDCG} \tag{3.12}$$

   where IDCG is the Ideal DCG, which just represents the DCG of the best ranked list

3.9 Results

We evaluated the performance of XGB and LGBM rankers and classifiers across different text representations (see 3.7.2) using the rank-based metrics described in section 3.8.6. We compared their performance to the Initiator Harm Level baseline, which represents the current process for screening severe SSE 1-4 events in the healthcare organization (see 3.8.5). The subsequent sections detail the performance of each model-feature combination and discusses the results.

3.9.1 XGB Ranker

Tables 3.8 to 3.16 show results for XGB Ranker across various text representations. We observe GatorTron features report highest values for most metrics with 0.4074 R-Precision, 0.81 NDCG, 0.9704 AUROC, 0.4068 MAP, and 1.00 MRR on hold out test set. BioGPT feature follows, reporting 0.4259 R-Precision, 0.7756 NDCG, 0.9589 AUROC, 0.3427 MAP, and 1.00 MRR on test set. Hence, we can say domain specific features outperforms static and generic contextual features across most metrics.

Table 3.8: XGB Ranker - CB

Feature	Metric	Train	Test
CB	R-Precision	0.7763	0.2778
	NDCG	0.9580	0.7130
	AUROC	0.9992	0.9704
	MAP	0.8085	0.2458
	MRR	1.0000	1.0000
	p@20	0.9000	0.3000
	p@50	0.8800	0.2800
	p@100	0.8700	0.2200
	p@200	0.8100	0.1750
	p@500	0.4300	0.0860
	p@1000	0.2190	0.0480
	r@20	0.0822	0.1111
	r@50	0.2009	0.2593
	r@100	0.3973	0.4074
	r@200	0.7397	0.6481
	r@500	0.9817	0.7963
	r@1000	1.0000	0.8889

Table 3.9: XGB Ranker - TF-IDF

Feature	Metric	Train	Test
TF-IDF	R-Precision	0.4338	0.3333
	NDCG	0.8492	0.7574
	AUROC	0.9736	0.9651
	MAP	0.4350	0.3117
	MRR	1.0000	1.0000
	p@20	0.9500	0.5000
	p@50	0.8400	0.3400
	p@100	0.6700	0.2600
	p@200	0.4450	0.1800
	p@500	0.2620	0.0780
	p@1000	0.1550	0.0450
	r@20	0.0868	0.1852
	r@50	0.1918	0.3148
	r@100	0.3059	0.4815
	r@200	0.4064	0.6667
	r@500	0.5982	0.7222
	r@1000	0.7078	0.8333

Table 3.10: XGB Ranker - GloVe

Feature	Metric	Train	Test
GloVe	R-Precision	0.9954	0.3519
	NDCG	1.0000	0.7304
	AUROC	1.0000	0.9584
	MAP	1.0000	0.2964
	MRR	1.0000	1.0000
	p@20	1.0000	0.6000
	p@50	1.0000	0.3800
	p@100	1.0000	0.2300
	p@200	1.0000	0.1500
	p@500	0.4380	0.0880
	p@1000	0.2190	0.0480
	r@20	0.0913	0.2222
	r@50	0.2283	0.3519
	r@100	0.4566	0.4259
	r@200	0.9132	0.5556
	r@500	1.0000	0.8148
	r@1000	1.0000	0.8889

Table 3.11: XGB Ranker - RoBERTa

Feature	Metric	Train	Test
RoBERTa	R-Precision	1.0000	0.3519
	NDCG	1.0000	0.7147
	AUROC	1.0000	0.9594
	MAP	1.0000	0.2653
	MRR	1.0000	1.0000
	p@20	1.0000	0.4000
	p@50	1.0000	0.3600
	p@100	1.0000	0.2600
	p@200	1.0000	0.1550
	p@500	0.4380	0.0860
	p@1000	0.2190	0.0460
	r@20	0.0913	0.1481
	r@50	0.2283	0.3333
	r@100	0.4566	0.4815
	r@200	0.9132	0.5741
	r@500	1.0000	0.7963
	r@1000	1.0000	0.8519

Table 3.12: XGB Ranker - DeBERTa

Feature	Metric	Train	Test
DeBERTa	R-Precision	1.0000	0.3333
	NDCG	1.0000	0.7087
	AUROC	1.0000	0.9513
	MAP	1.0000	0.2464
	MRR	1.0000	1.0000
	p@20	1.0000	0.3500
	p@50	1.0000	0.3400
	p@100	1.0000	0.2300
	p@200	1.0000	0.1550
	p@500	0.4380	0.0820
	p@1000	0.2190	0.0460
	r@20	0.0913	0.1296
	r@50	0.2283	0.3148
	r@100	0.4566	0.4259
	r@200	0.9132	0.5741
	r@500	1.0000	0.7593
	r@1000	1.0000	0.8519

Table 3.13: XGB Ranker - BGE

Feature	Metric	Train	Test
BGE	R-Precision	1.0000	0.3148
	NDCG	1.0000	0.6884
	AUROC	1.0000	0.9578
	MAP	1.0000	0.2651
	MRR	1.0000	0.3333
	p@20	1.0000	0.5500
	p@50	1.0000	0.3400
	p@100	1.0000	0.2200
	p@200	1.0000	0.1550
	p@500	0.4380	0.0900
	p@1000	0.2190	0.0480
	r@20	0.0913	0.2037
	r@50	0.2283	0.3148
	r@100	0.4566	0.4074
	r@200	0.9132	0.5741
	r@500	1.0000	0.8333
	r@1000	1.0000	0.8889

Table 3.14: XGB Ranker - BiomedBERT

Feature	Metric	Train	Test
BiomedBERT	R-Precision	1.0000	0.3889
	NDCG	1.0000	0.6946
	AUROC	1.0000	0.9497
	MAP	1.0000	0.2864
	MRR	1.0000	0.3333
	p@20	1.0000	0.5500
	p@50	1.0000	0.3800
	p@100	1.0000	0.2800
	p@200	1.0000	0.1700
	p@500	0.4380	0.0820
	p@1000	0.2190	0.0450
	r@20	0.0913	0.2037
	r@50	0.2283	0.3519
	r@100	0.4566	0.5185
	r@200	0.9132	0.6296
	r@500	1.0000	0.7593
	r@1000	1.0000	0.8333

Table 3.15: XGB Ranker - GatorTron

Feature	Metric	Train	Test
GatorTron	R-Precision	0.8995	0.4074
	NDCG	0.9938	0.8100
	AUROC	0.9998	0.9704
	MAP	0.9622	0.4068
	MRR	1.0000	1.0000
	p@20	1.0000	0.6500
	p@50	1.0000	0.4400
	p@100	1.0000	0.3000
	p@200	0.9450	0.1750
	p@500	0.4360	0.0900
	p@1000	0.2190	0.0490
	r@20	0.0913	0.2407
	r@50	0.2283	0.4074
	r@100	0.4566	0.5556
	r@200	0.8630	0.6481
	r@500	0.9954	0.8333
	r@1000	1.0000	0.9074

Table 3.16: XGB Ranker - BioGPT

Feature	Metric	Train	Test
BioGPT	R-Precision	0.9543	0.4259
	NDCG	0.9994	0.7756
	AUROC	1.0000	0.9589
	MAP	0.9955	0.3427
	MRR	1.0000	1.0000
	p@20	1.0000	0.5000
	p@50	1.0000	0.4200
	p@100	1.0000	0.2400
	p@200	0.9900	0.1650
	p@500	0.4380	0.0840
	p@1000	0.2190	0.0460
	r@20	0.0913	0.1852
	r@50	0.2283	0.3889
	r@100	0.4566	0.4444
	r@200	0.9041	0.6111
	r@500	1.0000	0.7778
	r@1000	1.0000	0.8519

3.9.2 LGBM Ranker

Tables 3.17 to 3.25 show results for LGBM Ranker. We observe BioGPT features report high values for most metrics with 0.4074 R-Precision, 0.7211 NDCG, 0.70 Precision@20, 0.2593 Recall@20,

0.9602 AUROC, 0.3386 MAP, and 0.3333 MRR on hold out test set. BioMedBERT follows, reporting 0.3880 R-Precision, 0.7706 NDCG, 0.60 Precision@20, 0.2222 Recall@20, 0.9644 AUROC, 0.3322 MAP, and 1.00 MRR on test set. Overall domain specific features such as BioGPT, BioMedBERT and GatorTron seems to outperform other features. Deberta seems to perform comparably with domain specific features, however Roberta and all static features performs significantly worse than domain specific features.

Table 3.17: LGBM Ranker - CB

Feature	Metric	Train	Test
CB	R-Precision	0.6164	0.3333
	NDCG	0.9060	0.7653
	AUROC	0.9773	0.9623
	MAP	0.6445	0.3312
	MRR	1.0000	1.0000
	p@20	1.0000	0.5500
	p@50	0.9600	0.3400
	p@100	0.9600	0.2700
	p@200	0.6600	0.1750
	p@500	0.2960	0.0820
	p@1000	0.1660	0.0450
	r@20	0.0913	0.2037
	r@50	0.2192	0.3148
	r@100	0.4384	0.5000
	r@200	0.6027	0.6481
	r@500	0.6758	0.7593
	r@1000	0.7580	0.8333

Table 3.18: LGBM Ranker - TF-IDF

Feature	Metric	Train	Test
TF-IDF	R-Precision	0.3470	0.2593
	NDCG	0.8039	0.7534
	AUROC	0.9527	0.9383
	MAP	0.3381	0.3068
	MRR	1.0000	1.0000
	p@20	0.8500	0.5000
	p@50	0.7600	0.2800
	p@100	0.5500	0.2300
	p@200	0.3650	0.1750
	p@500	0.2100	0.0820
	p@1000	0.1470	0.0410
	r@20	0.0776	0.1852
	r@50	0.1735	0.2593
	r@100	0.2511	0.4259
	r@200	0.3333	0.6481
	r@500	0.4795	0.7593
	r@1000	0.6712	0.7593

3.9.3 XGB Classifier

Tables 3.26 to 3.34 show results for XGB Classifier. GatorTron outperforms all other features achieving 0.4630 R-Precision, 0.7358 NDCG, 0.55 Precision@20, 0.2037 Recall@20, 0.9610 AUROC, 0.3706 MAP, and 0.3333 MRR on the hold out test set. TF_IDF follows, reporting 0.3333 R-Precision, 0.7572 NDCG, 0.5500 Precision@20, 0.2037 Recall@20, 0.9685 AUROC, 0.3095 MAP, and 1.00 MRR on test set.

3.9.4 LGBM Classifier

Tables 3.35 to 3.43 show results for LGBM Classifier. GatorTron outperforms all other features achieving 0.4259 R-Precision, 0.8125 NDCG, 0.70 Precision@20, 0.2593 Recall@20, 0.9694 AUROC,

Table 3.19: LGBM Ranker - GloVe

Feature	Metric	Train	Test
GloVe	R-Precision	0.6119	0.3333
	NDCG	0.8899	0.7099
	AUROC	0.9428	0.8896
	MAP	0.6151	0.2598
	MRR	1.0000	1.0000
	p@20	1.0000	0.4000
	p@50	0.9800	0.3400
	p@100	0.9100	0.2500
	p@200	0.6500	0.1650
	p@500	0.2920	0.0780
	p@1000	0.1600	0.0430
	r@20	0.0913	0.1481
	r@50	0.2237	0.3148
	r@100	0.4155	0.4630
	r@200	0.5936	0.6111
	r@500	0.6667	0.7222
	r@1000	0.7306	0.7963

Table 3.20: LGBM Ranker - RoBERTa

Feature	Metric	Train	Test
RoBERTa	R-Precision	0.9863	0.3333
	NDCG	0.9994	0.6598
	AUROC	1.0000	0.9599
	MAP	0.9963	0.2403
	MRR	1.0000	0.2500
	p@20	1.0000	0.4500
	p@50	1.0000	0.3400
	p@100	1.0000	0.2700
	p@200	1.0000	0.1750
	p@500	0.4380	0.0820
	p@1000	0.2190	0.0480
	r@20	0.0913	0.1667
	r@50	0.2283	0.3148
	r@100	0.4566	0.5000
	r@200	0.9132	0.6481
	r@500	1.0000	0.7593
	r@1000	1.0000	0.8889

Table 3.21: LGBM Ranker - DeBERTa

Feature	Metric	Train	Test
DeBERTa	R-Precision	0.9863	0.4074
	NDCG	0.9996	0.6895
	AUROC	1.0000	0.9677
	MAP	0.9974	0.2835
	MRR	1.0000	0.3333
	p@20	1.0000	0.5000
	p@50	1.0000	0.4200
	p@100	1.0000	0.3000
	p@200	1.0000	0.1700
	p@500	0.4380	0.0900
	p@1000	0.2190	0.0500
	r@20	0.0913	0.1852
	r@50	0.2283	0.3889
	r@100	0.4566	0.5556
	r@200	0.9132	0.6296
	r@500	1.0000	0.8333
	r@1000	1.0000	0.9259

Table 3.22: LGBM Ranker - BGE

Feature	Metric	Train	Test
BGE	R-Precision	0.6986	0.2963
	NDCG	0.9258	0.7104
	AUROC	0.9745	0.9087
	MAP	0.7218	0.2569
	MRR	1.0000	1.0000
	p@20	1.0000	0.4500
	p@50	1.0000	0.3000
	p@100	0.9900	0.2800
	p@200	0.7350	0.1650
	p@500	0.3240	0.0820
	p@1000	0.1660	0.0420
	r@20	0.0913	0.1667
	r@50	0.2283	0.2778
	r@100	0.4521	0.5185
	r@200	0.6712	0.6111
	r@500	0.7397	0.7593
	r@1000	0.7580	0.7778

0.4205 MAP, and 1.00 MRR on hold out test set. BGE follows, reporting 0.3704 R-Precision, 0.7470 NDCG, 0.60 Precision@20, 0.2222 Recall@20, 0.9437 AUROC, 0.3029 MAP, and 1.00 MRR on test set.

Table 3.23: LGBM Ranker - BiomedBERT

Feature	Metric	Train	Test
BiomedBERT	R-Precision	0.9224	0.3889
	NDCG	0.9853	0.7706
	AUROC	0.9910	0.9644
	MAP	0.9497	0.3322
	MRR	1.0000	1.0000
	p@20	1.0000	0.6000
	p@50	1.0000	0.4000
	p@100	1.0000	0.2700
	p@200	0.9900	0.1600
	p@500	0.4200	0.0920
	p@1000	0.2100	0.0490
	r@20	0.0913	0.2222
	r@50	0.2283	0.3704
	r@100	0.4566	0.5000
	r@200	0.9041	0.5926
	r@500	0.9589	0.8519
	r@1000	0.9589	0.9074

Table 3.24: LGBM Ranker - GatorTron

Feature	Metric	Train	Test
GatorTron	R-Precision	1.0000	0.3519
	NDCG	1.0000	0.7174
	AUROC	1.0000	0.9507
	MAP	1.0000	0.3287
	MRR	1.0000	0.3333
	p@20	1.0000	0.6000
	p@50	1.0000	0.3600
	p@100	1.0000	0.3100
	p@200	1.0000	0.1800
	p@500	0.4380	0.0840
	p@1000	0.2190	0.0460
	r@20	0.0913	0.2222
	r@50	0.2283	0.3333
	r@100	0.4566	0.5741
	r@200	0.9132	0.6667
	r@500	1.0000	0.7778
	r@1000	1.0000	0.8519

Table 3.25: LGBM Ranker - BioGPT

Feature	Metric	Train	Test
BioGPT	R-Precision	1.0000	0.4074
	NDCG	1.0000	0.7211
	AUROC	1.0000	0.9602
	MAP	1.0000	0.3386
	MRR	1.0000	0.3333
	p@20	1.0000	0.7000
	p@50	1.0000	0.4400
	p@100	1.0000	0.2800
	p@200	1.0000	0.1550
	p@500	0.4380	0.0800
	p@1000	0.2190	0.0470
	r@20	0.0913	0.2593
	r@50	0.2283	0.4074
	r@100	0.4566	0.5185
	r@200	0.9132	0.5741
	r@500	1.0000	0.7407
	r@1000	1.0000	0.8704

Table 3.26: XGB Classifier - CB

Feature	Metric	Train	Test
CB	R-Precision	0.5799	0.1852
	NDCG	0.9178	0.6782
	AUROC	0.9970	0.9586
	MAP	0.6193	0.1921
	MRR	1.0000	1.0000
	p@20	0.9500	0.3000
	p@50	0.8400	0.2000
	p@100	0.7700	0.1500
	p@200	0.6150	0.1450
	p@500	0.3220	0.0820
	p@1000	0.2190	0.0430
	r@20	0.0868	0.1111
	r@50	0.1918	0.1852
	r@100	0.3516	0.2778
	r@200	0.5616	0.5370
	r@500	0.7352	0.7593
	r@1000	1.0000	0.7963

Table 3.27: XGB Classifier -TF-IDF

Feature	Metric	Train	Test
TF-IDF	R-Precision	0.7032	0.3333
	NDCG	0.9470	0.7572
	AUROC	0.9932	0.9685
	MAP	0.7644	0.3095
	MRR	1.0000	1.0000
	p@20	1.0000	0.5500
	p@50	0.9800	0.3400
	p@100	0.9700	0.2500
	p@200	0.7450	0.1550
	p@500	0.3560	0.0820
	p@1000	0.1950	0.0460
	r@20	0.0913	0.2037
	r@50	0.2237	0.3148
	r@100	0.4429	0.4630
	r@200	0.6804	0.5741
	r@500	0.8128	0.7593
	r@1000	0.8904	0.8519

Table 3.28: XGB Classifier - GloVe

Feature	Metric	Train	Test
GloVe	R-Precision	1.0000	0.3519
	NDCG	1.0000	0.6772
	AUROC	1.0000	0.9590
	MAP	1.0000	0.2550
	MRR	1.0000	0.3333
	p@20	1.0000	0.5000
	p@50	1.0000	0.3800
	p@100	1.0000	0.2300
	p@200	1.0000	0.1700
	p@500	0.4380	0.0820
	p@1000	0.2190	0.0480
	r@20	0.0913	0.1852
	r@50	0.2283	0.3519
	r@100	0.4566	0.4259
	r@200	0.9132	0.6296
	r@500	1.0000	0.7593
	r@1000	1.0000	0.8889

Table 3.29: XGB Classifier - RoBERTa

Feature	Metric	Train	Test
RoBERTa	R-Precision	1.0000	0.3148
	NDCG	1.0000	0.6686
	AUROC	1.0000	0.9635
	MAP	1.0000	0.2366
	MRR	1.0000	0.3333
	p@20	1.0000	0.4000
	p@50	1.0000	0.3200
	p@100	1.0000	0.2400
	p@200	1.0000	0.1550
	p@500	0.4380	0.0820
	p@1000	0.2190	0.0470
	r@20	0.0913	0.1481
	r@50	0.2283	0.2963
	r@100	0.4566	0.4444
	r@200	0.9132	0.5741
	r@500	1.0000	0.7593
	r@1000	1.0000	0.8704

3.9.5 Cross-Validation: Baseline Vs. Best Model

This section compares the performance of our best model—the LGBM Classifier with GatorTron features— against the baseline, which ranks based on Initiator Harm level, averaged across five

Table 3.30: XGB Classifier - DeBERTa

Feature	Metric	Train	Test
DeBERTa	R-Precision	1.0000	0.3148
	NDCG	1.0000	0.6992
	AUROC	1.0000	0.9582
	MAP	1.0000	0.2869
	MRR	1.0000	0.3333
	p@20	1.0000	0.5500
	p@50	1.0000	0.3400
	p@100	1.0000	0.2600
	p@200	1.0000	0.1750
	p@500	0.4380	0.0820
	p@1000	0.2190	0.0470
	r@20	0.0913	0.2037
	r@50	0.2283	0.3148
	r@100	0.4566	0.4815
	r@200	0.9132	0.6481
	r@500	1.0000	0.7593
	r@1000	1.0000	0.8704

Table 3.31: XGB Classifier - BGE

Feature	Metric	Train	Test
BGE	R-Precision	1.0000	0.3519
	NDCG	1.0000	0.6674
	AUROC	1.0000	0.9472
	MAP	1.0000	0.2418
	MRR	1.0000	0.3333
	p@20	1.0000	0.5000
	p@50	1.0000	0.3800
	p@100	1.0000	0.2200
	p@200	1.0000	0.1550
	p@500	0.4380	0.0820
	p@1000	0.2190	0.0450
	r@20	0.0913	0.1852
	r@50	0.2283	0.3519
	r@100	0.4566	0.4074
	r@200	0.9132	0.5741
	r@500	1.0000	0.7593
	r@1000	1.0000	0.8333

Table 3.32: XGB Classifier -BiomedBERT

Feature	Metric	Train	Test
BiomedBERT	R-Precision	1.0000	0.3333
	NDCG	1.0000	0.7252
	AUROC	1.0000	0.9534
	MAP	1.0000	0.2817
	MRR	1.0000	1.0000
	p@20	1.0000	0.5500
	p@50	1.0000	0.3600
	p@100	1.0000	0.2600
	p@200	1.0000	0.1650
	p@500	0.4380	0.0800
	p@1000	0.2190	0.0470
	r@20	0.0913	0.2037
	r@50	0.2283	0.3333
	r@100	0.4566	0.4815
	r@200	0.9132	0.6111
	r@500	1.0000	0.7407
	r@1000	1.0000	0.8704

Table 3.33: XGB Classifier - GatorTron

Feature	Metric	Train	Test
GatorTron	R-Precision	1.0000	0.4630
	NDCG	1.0000	0.7358
	AUROC	1.0000	0.9610
	MAP	1.0000	0.3706
	MRR	1.0000	0.3333
	p@20	1.0000	0.5500
	p@50	1.0000	0.4800
	p@100	1.0000	0.3200
	p@200	1.0000	0.2050
	p@500	0.4380	0.0900
	p@1000	0.2190	0.0490
	r@20	0.0913	0.2037
	r@50	0.2283	0.4444
	r@100	0.4566	0.5926
	r@200	0.9132	0.7593
	r@500	1.0000	0.8333
	r@1000	1.0000	0.9074

different random dataset splits, detailed in Table 3.44. This approach demonstrates the model's robustness, as it consistently outperforms the baseline across different test sets. For reproducibility, we provide the hyperparameter values of the best model in table 3.45.

Table 3.34: XGB Classifier - BioGPT

Feature	Metric	Train	Test
BioGPT	R-Precision	1.0000	0.3519
	NDCG	1.0000	0.6963
	AUROC	1.0000	0.9604
	MAP	1.0000	0.2871
	MRR	1.0000	0.3333
	p@20	1.0000	0.5500
	p@50	1.0000	0.3800
	p@100	1.0000	0.2700
	p@200	1.0000	0.1600
	p@500	0.4380	0.0800
	p@1000	0.2190	0.0460
	r@20	0.0913	0.2037
	r@50	0.2283	0.3519
	r@100	0.4566	0.5000
	r@200	0.9132	0.5926
	r@500	1.0000	0.7407
	r@1000	1.0000	0.8519

Table 3.35: LGBM Classifier - CB

Feature	Metric	Train	Test
CB	R-Precision	1.000	0.3148
	NDCG	1.000	0.7080
	AUROC	1.000	0.9566
	MAP	1.000	0.2414
	MRR	1.000	1.000
	p@20	1.000	0.4000
	p@50	1.000	0.3000
	p@100	1.000	0.2100
	p@200	1.000	0.1550
	p@500	0.4380	0.0820
	p@1000	0.2190	0.0480
	r@20	0.0913	0.1481
	r@50	0.2283	0.2778
	r@100	0.4566	0.3889
	r@200	0.9132	0.5741
	r@500	1.000	0.7593
	r@1000	1.000	0.8889

Table 3.36: LGBM Classifier - TF-IDF

Feature	Metric	Train	Test
TF-IDF	R-Precision	0.3744	0.2222
	NDCG	0.7929	0.7122
	AUROC	0.9956	0.9403
	MAP	0.3493	0.2422
	MRR	0.3333	1.000
	p@20	0.4500	0.4000
	p@50	0.4800	0.2400
	p@100	0.4200	0.2000
	p@200	0.3800	0.1550
	p@500	0.2900	0.0840
	p@1000	0.2190	0.0440
	r@20	0.0411	0.1481
	r@50	0.1096	0.2222
	r@100	0.1918	0.3704
	r@200	0.3470	0.5741
	r@500	0.6621	0.7778
	r@1000	1.000	0.8148

Table 3.37: LGBM Classifier - GloVe

Feature	Metric	Train	Test
GloVe	R-Precision	0.9954	0.3333
	NDCG	1.0000	0.6241
	AUROC	1.0000	0.9475
	MAP	0.9998	0.1966
	MRR	1.0000	0.2500
	p@20	1.0000	0.3000
	p@50	1.0000	0.3200
	p@100	1.0000	0.2200
	p@200	1.0000	0.1700
	p@500	0.4380	0.0820
	p@1000	0.2190	0.0430
	r@20	0.0913	0.1111
	r@50	0.2283	0.2963
	r@100	0.4566	0.4074
	r@200	0.9132	0.6296
	r@500	1.0000	0.7593
	r@1000	1.0000	0.7963

Table 3.38: LGBM Classifier - RoBERTa

Feature	Metric	Train	Test
RoBERTa	R-Precision	1.0000	0.2593
	NDCG	1.0000	0.6260
	AUROC	1.0000	0.9495
	MAP	1.0000	0.1809
	MRR	1.0000	0.3333
	p@20	1.0000	0.4000
	p@50	1.0000	0.2600
	p@100	1.0000	0.2100
	p@200	1.0000	0.1300
	p@500	0.4380	0.0700
	p@1000	0.2190	0.0420
	r@20	0.0913	0.1481
	r@50	0.2283	0.2407
	r@100	0.4566	0.3889
	r@200	0.9132	0.4815
	r@500	1.0000	0.6481
	r@1000	1.0000	0.7778

Table 3.39: LGBM Classifier - DeBERTa

Feature	Metric	Train	Test
DeBERTa	R-Precision	1.0000	0.3148
	NDCG	1.0000	0.6835
	AUROC	1.0000	0.9527
	MAP	1.0000	0.2617
	MRR	1.0000	0.3333
	p@20	1.0000	0.4500
	p@50	1.0000	0.3400
	p@100	1.0000	0.2600
	p@200	1.0000	0.1600
	p@500	0.4380	0.0740
	p@1000	0.2190	0.0470
	r@20	0.0913	0.1667
	r@50	0.2283	0.3148
	r@100	0.4566	0.4815
	r@200	0.9132	0.5926
	r@500	1.0000	0.6852
	r@1000	1.0000	0.8704

Table 3.40: LGBM Classifier - BGE

Feature	Metric	Train	Test
BGE	R-Precision	0.6438	0.3704
	NDCG	0.8912	0.7470
	AUROC	0.9546	0.9437
	MAP	0.6277	0.3029
	MRR	1.0000	1.0000
	p@20	0.9500	0.6000
	p@50	0.8800	0.3800
	p@100	0.9100	0.2300
	p@200	0.6950	0.1600
	p@500	0.3120	0.0840
	p@1000	0.1560	0.0450
	r@20	0.0868	0.2222
	r@50	0.2009	0.3519
	r@100	0.4155	0.4259
	r@200	0.6347	0.5926
	r@500	0.7123	0.7778
	r@1000	0.7123	0.8333

3.10 Discussion

Our results indicate that domain-specific features, particularly GatorTron, significantly improves ranking performance achieving overall high rank scores across all models. Among the rankers, all

Table 3.41: LGBM Classifier - BiomedBERT

Feature	Metric	Train	Test
BiomedBERT	R-Precision	1.0000	0.2407
	NDCG	1.0000	0.6528
	AUROC	1.0000	0.9537
	MAP	1.0000	0.2108
	MRR	1.0000	0.3333
	p@20	1.0000	0.3500
	p@50	1.0000	0.2600
	p@100	1.0000	0.2200
	p@200	1.0000	0.1550
	p@500	0.4380	0.0760
	p@1000	0.2190	0.0450
	r@20	0.0913	0.1296
	r@50	0.2283	0.2407
	r@100	0.4566	0.4074
	r@200	0.9132	0.5741
	r@500	1.0000	0.7037
	r@1000	1.0000	0.8333

Table 3.42: LGBM Classifier - GatorTron

Feature	Metric	Train	Test
GatorTron	R-Precision	0.8402	0.4259
	NDCG	0.9836	0.8125
	AUROC	0.9992	0.9694
	MAP	0.9099	0.4205
	MRR	1.0000	1.0000
	p@20	1.0000	0.7000
	p@50	1.0000	0.4600
	p@100	1.0000	0.2800
	p@200	0.8850	0.1800
	p@500	0.4080	0.0880
	p@1000	0.2190	0.0470
	r@20	0.0913	0.2593
	r@50	0.2283	0.4259
	r@100	0.4566	0.5185
	r@200	0.8082	0.6667
	r@500	0.9315	0.8148
	r@1000	1.0000	0.8704

Table 3.43: LGBM Classifier - BioGPT

Feature	Metric	Train	Test
BioGPT	R-Precision	0.6712	0.2963
	NDCG	0.9191	0.7091
	AUROC	0.9806	0.9612
	MAP	0.6876	0.2396
	MRR	1.0000	1.0000
	p@20	1.0000	0.4500
	p@50	0.9800	0.2800
	p@100	0.9300	0.2000
	p@200	0.7250	0.1550
	p@500	0.3140	0.0780
	p@1000	0.1690	0.0440
	r@20	0.0913	0.1667
	r@50	0.2237	0.2593
	r@100	0.4247	0.3704
	r@200	0.6621	0.5741
	r@500	0.7169	0.7222
	r@1000	0.7717	0.8148

domain-specific embeddings, including GatorTron, BioGPT, and BioMedBERT, consistently achieve the best or near-best results across most metrics. This underscores the performance benefits of using domain-specific language models, which can more effectively interpret the linguistic patterns inherent

Table 3.44: Average Performance of Baseline Vs. Best Model; CV=5 (Mean ± SD)

Metric	**Baseline**		**LGBM-CLF-GTR**	
	Train	Test	Train	Test
R-Precision	0.1961 ± 0.0011	0.1931 ± 0.0041	0.8228 ± 0.0186	0.3222 ± 0.0465
NDCG	0.6768 ± 0.0000	0.6129 ± 0.0000	0.9845 ± 0.0014	0.7391 ± 0.0208
AUROC	0.9347 ± 0.0000	0.9382 ± 0.0000	0.9993 ± 0.0000	0.9576 ± 0.0039
MAP	0.1448 ± 0.0003	0.1534 ± 0.0007	0.9134 ± 0.0083	0.2881 ± 0.0380
MRR	0.5492 ± 0.0508	0.5529 ± 0.0192	1.0000 ± 0.0000	1.0000 ± 0.0000
p@20	0.2559 ± 0.1413	0.3187 ± 0.0049	1.0000 ± 0.0000	0.5000 ± 0.0935
p@50	0.3273 ± 0.0042	0.1994 ± 0.0030	1.0000 ± 0.0000	0.3280 ± 0.0460
p@100	0.3017 ± 0.0005	0.1528 ± 0.0029	1.0000 ± 0.0000	0.2440 ± 0.0321
p@200	0.2046 ± 0.0010	0.1299 ± 0.0010	0.8760 ± 0.0143	0.1700 ± 0.0079
p@500	0.1461 ± 0.0006	0.0800 ± 0.0001	0.4176 ± 0.0059	0.0864 ± 0.0022
p@1000	0.1265 ± 0.0007	0.0429 ± 0.0002	0.2182 ± 0.0008	0.0474 ± 0.0015
r@20	0.0292 ± 0.0005	0.1180 ± 0.0018	0.0913 ± 0.0000	0.1852 ± 0.0346
r@50	0.0747 ± 0.0010	0.1846 ± 0.0027	0.2283 ± 0.0000	0.3037 ± 0.0426
r@100	0.1378 ± 0.0002	0.2829 ± 0.0054	0.4566 ± 0.0000	0.4519 ± 0.0594
r@200	0.1869 ± 0.0009	0.4812 ± 0.0037	0.8000 ± 0.0131	0.6296 ± 0.0293
r@500	0.3335 ± 0.0014	0.7405 ± 0.0012	0.9534 ± 0.0135	0.8000 ± 0.0203
r@1000	0.5775 ± 0.0033	0.7952 ± 0.0027	0.9963 ± 0.0038	0.8778 ± 0.0281

Table 3.45: Hyperparameters of best model —LGBM Classifier with GatorTron

Hyperparameter	**Best value**
force_col_wise	TRUE
objective	binary
random_state	42
n_estimators	200
boosting_type	gbdt
max_depth	3
learning_rate	0.05
reg_lambda	1
num_leaves	31
scale_pos_weight	1

in medical texts.

Comparative analysis of XGB and LGBM models shows that while both rankers and classifiers perform similarly, classifiers, especially the LGBM classifier trained with GatorTron features, exhibit superior performance. The LGBM Classifier reports overall high metrics: 0.4259 R-precision, 0.8125 NDCG, 0.70 Precision@20, 0.2593 Recall@20, 0.9694 AUROC, 0.4205 MAP, and 1.00 MRR on the holdout test set. In contrast, the XGB classifier achieves the highest R-precision of 0.4630 but reports relatively lower scores in other metrics such as NDCG (0.7358), MAP (0.3706), and MRR (0.3333).

With XGB classifier, TF-IDF features rank second to GatorTron, illustrating that static representations can occasionally surpass the performance of contextual deep-learning-based embeddings. Similarly, BGE generic English language embeddings with LGBM classifier perform second best to GatorTron features, outperforming other static and contextual embeddings.

Although we allow the model to choose higher regularization values during the hyper-parameter tuning phase to prevent overfitting (see section 3.8.4), the models tend to do better with lower regularization, allowing them to overfit on the train set. This could be due to the large class imbalance with significantly small positive class, where higher regularization can prevent the model from capturing the patterns present in rarely occurring severe events. Hence, while smaller regularization does make models overfit on train set, it may enable models to learn new patterns, thereby improving performance on both train and test set.

The top-performing model—the LGBM classifier trained with GatorTron features—significantly outperforms the current Initiator Harm Level baseline. Its performance more than doubles in R-Precision (from 0.1970 to 0.4259), Precision@k, and Recall@k for $k = 20, 50, 100$, with MRR improving from 0.5351 to 1.00, and MAP nearly tripling from 0.1546 to 0.4205 on the test set. These results demonstrate the substantial benefits of applying ML techniques to rank severe safety events. This approach can greatly enhance current processes by reducing the time required for PSAs to identify these events, thereby making the process more efficient, reducing workload, and improving overall patient safety.

3.10.1 Limitations and Future Work

While our models significantly outperform current screening processes, the rank-based performance metrics observed, such as R-precision and MAP, do not align with typical scores in the ML field with more balanced datasets. Our challenges stem from a substantial imbalance in our dataset, with

a ratio of 0.38% positive to 99.62% negative classes. Furthermore, the variability in report quality, ranging from comprehensive and well-documented to vague and incomplete, poses a challenge for ML models in consistently labeling and extracting useful information from the textual descriptions [102]. For instance, reports often contain text descriptions as brief as 1-10 words, which may not provide enough context for accurate representation in the embedding space, potentially leading to incorrect predictions. Additionally, the complexity of these events often results in reports being mislabeled or assigned vague generic labels under pre-defined categories such as "Not Sure" for Initiator Harm Level or "Miscellaneous Incident" for Incident Type, which presents significant data quality issues [102]. This further complicates accurate model training and can increase the likelihood of incorrect classifications.

Therefore, with reduced class imbalance and more precise assignment of event types and harm levels, we anticipate further improvements in model performance. Additionally, due to computational constraints and data imbalance, we did not fine-tune the contextual embeddings. Future research could enhance performance by fine-tuning these embeddings.

Future work can explore different ways of grouping incidents as critical or non-critical during the training phase. For instance, if the incident descriptions of SSE5 or near miss events appear similar to those of critical SSE1-4 events, treating these as critical during training could enhance model performance. Additionally, using various multiclass SEC groupings instead of binary grouping to train the rankers may further improve model performance.

While data annotation is time-consuming, we envision these models as an initial step in accelerating the triage process by quickly identifying severe events and reducing the number of non-severe events undergoing in-depth review by the central quality and safety team. By enhancing classification efficiency, this could ultimately lead to a larger pool of incidents labeled with a SEC code, which can then form the basis for building more improved models with more training samples.

3.11 Conclusion

Healthcare organizations use Incident Reporting Systems to collect Patient Safety Event (PSE) reports, aiming to monitor and enhance care quality and safety. The effectiveness of these systems hinges on thorough analysis and review of these reports to develop preventive measures and recommendations. Patient Safety Analysts (PSAs) are required to manually review a large volume of PSE reports daily, to screen and prioritize potentially severe events (SSE1-4) for further detailed analysis. However, the current screening process, based on harm level assigned by the event initiator,

is inefficient and labor-intensive, requiring PSAs to scan through many non-severe incidents.

This study adopts a binary ranking approach, implementing XGB and LGBM ranker and classifier models across various text feature representations, that learn to rank potentially severe events higher on the review list. These models were evaluated against the existing initiator harm level screening method used by PSAs. The results indicate that the models significantly enhance rank quality over the baseline, with the LGBM classifier trained on GatorTron text features achieving the best results.

These models are intended to support the decision-making process of PSAs, by more accurately predicting the severity of incidents compared to baseline, thereby identifying potential Severe Safety Events (SSE 1-4). Integrating these models into the PSE report classification workflow could substantially reduce the workload of PSAs, speed up the identification of severe events, and expand the pool of incidents labeled with an SEC code, ultimately enhancing patient safety and quality of care.

Moreover, the reduction in triage workload for PSAs could allow more time for proactive quality improvement efforts that prevent harm before it occurs, rather than the current model of retrospective analysis after incidents occur. If applied on a broad scale, this shift could exponentially increase the benefits for patient safety, making a significant impact on healthcare quality through preventative measures.

Chapter 4

Evaluating Active Learning Strategies for PSE Type Classification

4.1 Introduction

In healthcare organizations, patient safety events (PSE), referring to incidents that compromise patient safety, such as medical errors, near misses, and adverse events pose a significant threat and can cause avoidable harm, injuries and even death [78]. The widespread implementation of Incident Reporting (IR) systems has been a major step forward in identifying and addressing PSEs. PSE reporting systems enable healthcare personnel to voluntarily log incidents, including medical errors and unsafe conditions, in the form of structured (e.g. event type, harm level, date, location), and unstructured data (e.g. textual event descriptions and outcomes) [3, 122]. These reports are then reviewed by hospital staff such as risk managers and patient safety analysts, playing a crucial role in enhancing healthcare quality by providing insights into factors leading to PSEs and facilitating the development of preventive measures [73].

Accurate classification of PSE reports into corresponding event types is crucial for analyzing trends, guiding interventions, and enhancing organizational learning [51, 15]. However, the high volume of PSEs reported [94] and the complexity of the classification taxonomy, which require specialized knowledge, make the classification process labour-intensive, time-consuming, and expensive

[102, 43]. Several studies have demonstrated the efficacy of supervised Machine Learning (ML) models in automating PSE report classification [172, 48, 30, 29]. Improving the reliability of report classification can enhance safety. However, supervised ML models require high-quality and voluminous manually-labelled datasets for optimal performance. Manually labeling data is time-consuming, costly, and often impractical due to the large volume of data generated [61]. Active Learning (AL) is a semi-supervised learning technique that involves selectively querying the most informative data points for manual labeling, thereby optimizing the use of human resources [144]. Our study aims to utilize human labelling resources efficiently to develop a robust dataset for training ML models, seeking to balance labelling costs and classification accuracy. This study investigates the efficacy of AL strategies in improving the performance of ML models in event type classification of PSE reports. Thus, our goal is to enhance the PSE labelling process by integrating Artificial Intelligence with human expertise. In this chapter we make the following contributions:

1. We simulate pool-based Active Learning with Support Vector Machine (SVM) and Logistic Regression (LR) models utilizing four Active Learning (AL) strategies, including three uncertainty-based and one combining information density with uncertainty. We compare and evaluate the performance of these strategies to a random selection baseline across various text representations, focusing on accuracy and Macro F-score improvements through iterative data sampling.

2. Our experiments reveal that AL strategies significantly outperform random sampling in both accuracy and Macro F-score, reducing the need for labeled samples by 24-69%. Notably, BioMedBERT, a medical domain-specific language model, achieves the highest accuracy gains within 20 iterations for both SVM and LR models, utilizing only 54% of PSE reports. This underscores the efficacy of domain-specific models in enhancing PSE report labeling and reducing manual workload.

3. Among the AL strategies, we find Entropy and Information Density (ID) Cosine with Least Confidence were particularly effective for SVM and LR models, respectively, in terms of both accuracy and Macro F-score. This demonstrates the efficacy of AL strategies, particularly when integrated with representativeness measures such as ID.

Some of the results from the work in this chapter have been accepted for publication [79].

4.1.1 Organization

Section 4.2 reviews related work in the literature. Section 4.3 describes the dataset used in this study. Section 4.4 discusses data pre-processing, including feature engineering, data augmentation, and splitting, in preparation for model building. Section 4.5 outlines the experimental setup, including AL strategies implemented and hyperparameter tuning. Section 4.6 presents the results. Finally, Sections 4.7 and 4.8 discuss the findings, limitations, potential future work and conclusions.

4.2 Related Work

In the literature, several Active Learning (AL) methods have been proposed that incorporate human expert labelling into the training loop of classification models, in efforts to improve the its performance.

4.2.1 AL with SVM Ensemble

One such study proposed a method that queries samples based on the posterior probabilities provided by a set of multi-class SVM classifiers at each iteration [61]. Samples with an average posterior probability below a certain threshold (i.e confidence in probability estimate) are then selected for labelling. The main objective is to minimize the number of labeled samples without sacrificing classification performance. The results demonstrate the method enhances accuracy across three datasets: it improved the Reuters-21578 dataset accuracy by 0.36% using only 10.77% of samples, the WebKB dataset by 1.27% with 48.4% of samples, and the 20,000 newsgroup dataset by 2.4% using only 61.5% of all samples.

4.2.2 AL with BERT

In 2020, Ein-dor et al. conducted a large-scale empirical study on active learning techniques for BERT-based classification, evaluating a diverse set of AL strategies across 10 datasets spanning various domains [45]. The study implemented pool-based active learning [144] querying a batch of 50 samples at each iteration, due to high computational demands of training BERT. At each iteration, the BERT model was fine-tuned from scratch to prevent overfitting from data from previous iteration [45]. The study analyzed seven state-of-the-art AL strategies including uncertainty-sampling, uncertainty-sampling using ensemble methods, and diversity-sampling. Additionally, it implements random-sampling as a baseline, where the batch of instances are chosen at random from the unlabeled

set for performance comparison.

Experiments involved binary classification on both balanced and imbalanced datasets. The results demonstrated that AL strategies can significantly enhance BERT's performance, particularly with skewed dataset common in real-world settings with limited annotation budgets. The AL strategies improved the F-score of the random baseline by a large margin of 4 - 8% on average. Despite these improvements, no single strategy consistently outperformed across all datasets—a common phenomenon with AL results as noted by Lowell et al. [107]. Furthermore, the study states that the uncertainty-based AL strategies tends to select outlier examples that do not properly represent the overall data distribution. Therefore diversity based sampling such as Discriminative Active Learning (DAL), ensures the selection of representative samples compared to all other strategies [45].

4.2.3 AL in Healthcare

AL using traditional SVM model

A study by Fong et al. (2017), compared three different AL strategies integrated with SVM to support binary classification of HIT-related Patient Safety Event (PSE) reports from a large set of diverse reports [53]. The dataset comprised of a subset of PSE reports from the Institute for Safe Medication Practices (ISMP) between 2007 and 2016 from hospitals in Pennsylvania, US [53]. The dataset consisted of a large unlabelled portion with 252 labelled samples and 5000 unlabelled samples [53]. The classifer would predict the class of the sample as either of the two classes: (1) "Likely", referring to HIT likely being a contributing factor to the safety event, (2) "Unlikely", referring to HIT unlikely being a contributing factor.

Three AL sampling strategies were evaluated—baseline uncertainty sampling (USB), uncertainty sampling prioritizing the selection of samples belonging to "Likely" class (UBL), and confirmatory sampling (CSL) where annotators are asked to confirm reports that the model predicts as "Likely" with a high level of confidence. The results demonstrate that all three strategies had high precision when evaluating high probability predictions (i.e. easily identifiable HIT-related events), with the USL, slightly outperforming other models [53]. However, for more complex reports, the CSL approach is better for earlier detection and removal of false positives. It concludes that the active learning approach offers a potentially resource efficient alternative to manual review of PSE (Patient Safety Event) reports, particularly in reducing the challenge of comprehending trends and patterns within large datasets [53].

AL using U-net deep learning model

Another study implements Deep Active Learning (DAL) approach for right ventricular segmentation to facilitate labelling of large number on unlabeled images from Cardiac Magnetic Resonance Imaging (CMRI) [5]. This method aims to enhance segmentation accuracy by selecting highly informative unlabeled data to train a U-Net based architecture, based on a two-level uncertainty estimation considering both epistemic (i.e. model) and aleatoric (i.e. interpretation) uncertainties. Tested on approximately 2000 images gathered from public and custom datasets, the results demonstrate improved segmentation metrics, proving the efficacy of DAL and the benefits of using uncertainty measures in leveraging unlabeled data effectively.

4.3 Dataset

This study utilized 861 PSE reports from a Southeastern U.S. academic hospital's maternal-care units from January 2019 to December 2020. The study was approved by the hospital's institutional review board (Pro00105892). Among the 25 distinct event types present in the dataset, the ML classifiers were exclusively trained on PSE-types with at least 40—samples constituting about 73% of all reports—to ensure sufficient learning data (see Table 4.1). For event type classification, only the free-text incident descriptions were used as model inputs. To abide by the privacy regulations, all PSE reports were anonymized after data extraction.

Table 4.1: Distribution of PSE types

Event type	**Extracted Reports (n=861)**	
	Count	**Percent %**
Care coordination or communication	186	21.6
Laboratory test	122	14.2
Medication related	89	10.3
Omission or errors in assessment, diagnosis, and monitoring	67	7.8
Maternal	58	6.7
Equipment or devices	56	6.5
Supplies	49	5.7
Total	**627**	**72.8**

4.4 Data Preprocessing

The following sections outline the preprocessing steps we implemented to prepare the incident descriptions for classifier input.

4.4.1 Data cleaning

To eliminate noise and irrelevant components from the incident texts, we perform data cleaning before obtaining word-level static text representations. This involved lowercasing, removing stop words and non-alphabetic characters, and normalizing word forms through stemming. However, for deep contextual representations, which represent sequences of words, we use the raw text to preserve context and avoid distortions from cleaning.

4.4.2 Feature extraction

We consider two types of static features that can be extracted from textual incident descriptions namely binary bag-of-words (CB) and TF-IDF—which focus on word occurrence and frequency but ignore the order [113]. The static representations were obtained from cleaned text, and tokenized into unigrams, bigrams, and trigrams to create the vocabulary. For deep learning-based contextual NLP representations, we evaluated two models trained on a generic English corpus: RoBERTa [106] and DeBERTa [70], both enhancements of the original BERT transformer model [41], reporting superior performance on multiple NLP benchmarks. Contextual representations are known to better capture the meaning of words by incorporating their position and context within sentences [140, 175, 156, 30, 29]. Additionally, we explored two domain-specific language models that were trained on biomedical and clinical data, namely BiomedBERT [64] and GatorTron [183], to better capture the unique linguistic patterns of medical texts. For detailed background on these text representations, see section 2.1 in Chapter 2.

Due to the small size of the dataset, we did not fine-tune the transformer models for our specific task but directly retrieved sentence embeddings from pre-trained models, by performing mean-pooling on the output token embeddings. The mean pooled embedding yielded slightly better results, utilizing the average of all hidden layers, compared to using only the CLS token, which represents the final hidden state. The dimensions of the resulting embeddings vary depending on the transformer model used. For example, RoBERTa base model produces an embedding of 768 dimensions, while larger models like GatorTron generate embeddings of 1024 dimensions. The maximum input token length for all models was set at 512, which is standard and sufficient for the dataset, given most descriptions are under 200 words (see Figure 4.1).

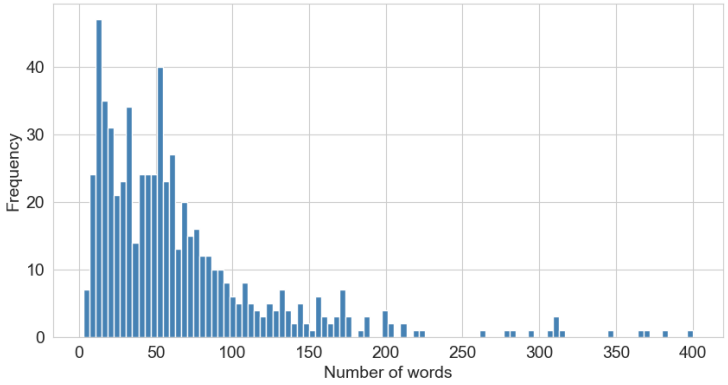


Figure 4.1: Word count distribution of incident description in PSE reports

4.4.3 Data augmentation

The dataset is imbalanced with uneven representation across different event type categories (see Table 4.1). Training on imbalanced datasets often degrades classifier performance [67, 26, 69]. To mitigate this, we apply the synthetic minority oversampling technique (SMOTE) [27], which generates synthetic examples by interpolating among several minority class instances, rather than simply replicating them [50]. SMOTE was applied to the training dataset after feature extraction at each AL iteration before retraining the classifier, while the validation and test set remained unchanged.

4.4.4 Data splitting

The dataset was divided into 70% training, 15% validation, and 15% testing sets, using a stratified split to ensure consistent class distribution. The train set was used for classifier learning, while the validation set facilitated hyperparameter tuning and was used to select the best model across iterations. This approach prevented overfitting and enhanced the stability of AL experiments.

4.5 Experimental Setup

In this study, we particularly focus on simulating the pool-based AL sampling approach [62] (see section 2.2.1), by initially introducing a small portion of the labeled dataset to the ML model for training. Then, additional informative samples are selected iteratively by the AL strategy and the model is retrained from scratch to evaluate if the additional samples help improve classification accuracy (see Figure 4.2).

All experiments start with a balanced set of 35 samples, with 5 samples per event type, drawn

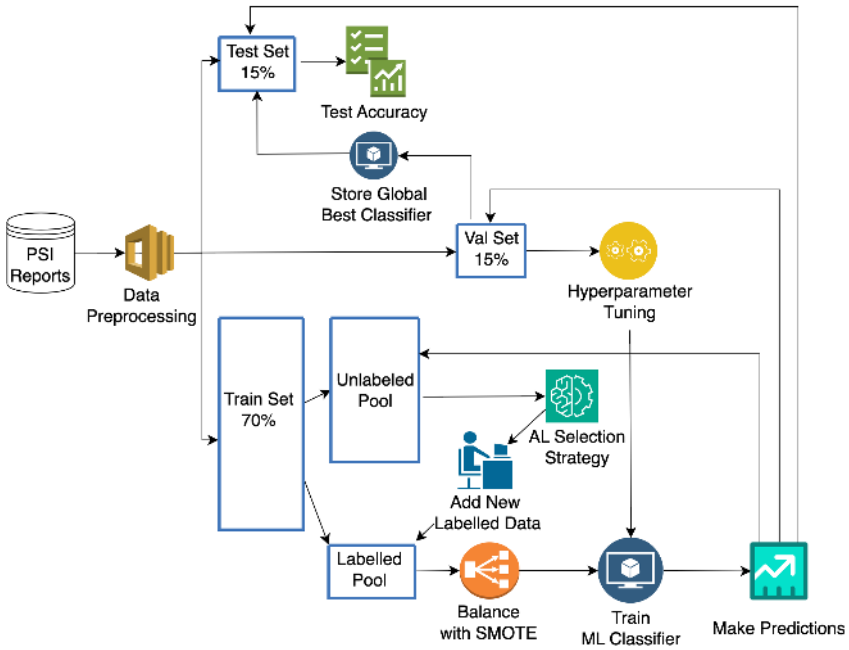


Figure 4.2: Active Learning Cycle

from the training set, allocating the rest to the unlabeled pool. In each AL iteration, 10 samples from the unlabeled pool are selected using the specified AL strategy and, consequently, these samples are added to the labelled set to simulate manual labelling of the selected samples. The ML model was then re-trained on this expanded set, after applying SMOTE oversampling to address potential class-imbalance. Re-training from scratch is done to prevent classifiers from overfitting on data from previous iterations, as suggested in [75]. This process was repeated until all samples from the unlabelled set were exhausted. We tested the models after each AL iteration with an unseen test set, using them to predict the corresponding PSE type. These predictions were evaluated against the ground truth labels using classification accuracy as the metric. To mitigate bias due to initial selection of labelled samples, each experiment was repeated with 10 random balanced initial labelled sets, and the average test accuracy was reported for each iteration.

AL experiments were conducted using two popular models in text classification tasks: Logistic Regression (LR) [93] and Support Vector Machines (SVM) [38]. We selected SVM based on previous studies using the same dataset, which demonstrated its superiority over other models such as XGB and LGBM [30, 29]. Our analysis confirmed that SVM slightly outperformed XGB with AL for this dataset, a finding consistent with SVM's known effectiveness in scenarios where the dataset size is smaller relative to the dimensionality of the feature space [131]. Furthermore, an extensive literature review in Chapter 3, Section 3.4 revealed that SVM often outperforms other ML models in classifying

PSE reports across various NLP techniques [123, 52, 172, 48, 30, 29]. We also evaluated performance using Logistic Regression (LR) because it is a widely used data mining method and serves as a popular baseline for comparison against state-of-the-art ML models. For detailed explanations of these ML models, see Appendix A. Four AL strategies, along with a random sampling baseline, were evaluated across six text representations using both SVM and LR classifiers, totaling 60 experiments.

4.5.1 AL strategies

Several AL selection strategies have been proposed in literature, including uncertainty-based, Query By Committee (QBC), and Information Density (ID) based approaches [146]. For background on these sampling techniques refer to Section 2.2.2. Our study explores three uncertainty-based methods: Least Confidence (LC), Margin (M), and Entropy (E) and a cosine similarity-based ID sampling combined with LC ($IDCos \times LC$). Additionally, we implement a random selection (R) as a baseline to compare against AL strategies. We provide a brief explanation for each AL strategy. For implementation details on these AL strategies refer to Appendix C.

- Least Confidence (LC): Queries the instance with the least confident prediction
- Margin (M): Measures uncertainty as the difference between the probabilities of the top two class predictions
- Entropy (E): Measures uncertainty using the probability distribution of an event's outcome, with high values indicating multiple equally likely outcomes
- Information Density with LC ($IDCos \times LC$): Weighs the informativeness of an instance against its similarity to the entire dataset, using measures such as cosine similarity. It is used in conjunction with a base sampling strategy like LC, querying instances that are both informative and representative of the input space distribution [145].
- Random selection (R): This method randomly selects instances from the unlabeled set for labeling, in contrast to AL strategies that select instances in a more informed manner. It is commonly used as a baseline to compare against AL strategies [45, 107].

4.5.2 Hyperparameter tuning

The 15% validation set was used for hyperparameter tuning instead of using cross-validation on the training set, to save time and computational resources. Tuning was conducted for each model-feature

combination, keeping the parameters consistent across all AL strategies for effective comparison. We used grid search technique [103] with 3-fold out-of-sample testing on the validation set, exploring a set of values for each hyperparameter. Table 4.2 lists the hyperparameter ranges for both SVM and LR.

Table 4.2: Hyperparameter ranges tuned for SVM and LR

Model	Hyperparameter	Values
SVM	C	[0.2, 0.5, 1, 3, 5, 7]
	kernel	[linear, rbf, poly, sigmoid]
	gamma	[scale, auto]
	degree	[3, 5]
	class_weight	balanced
	max_iter	1000
	random_state	42
LR	C	[0.1, 0.2, 0.5, 1, 3, 5, 7]
	penalty	[l2, None]
	solver	[newton-cg, lbfgs, liblinear]
	max_iter	500
	random_state	42

4.6 Results

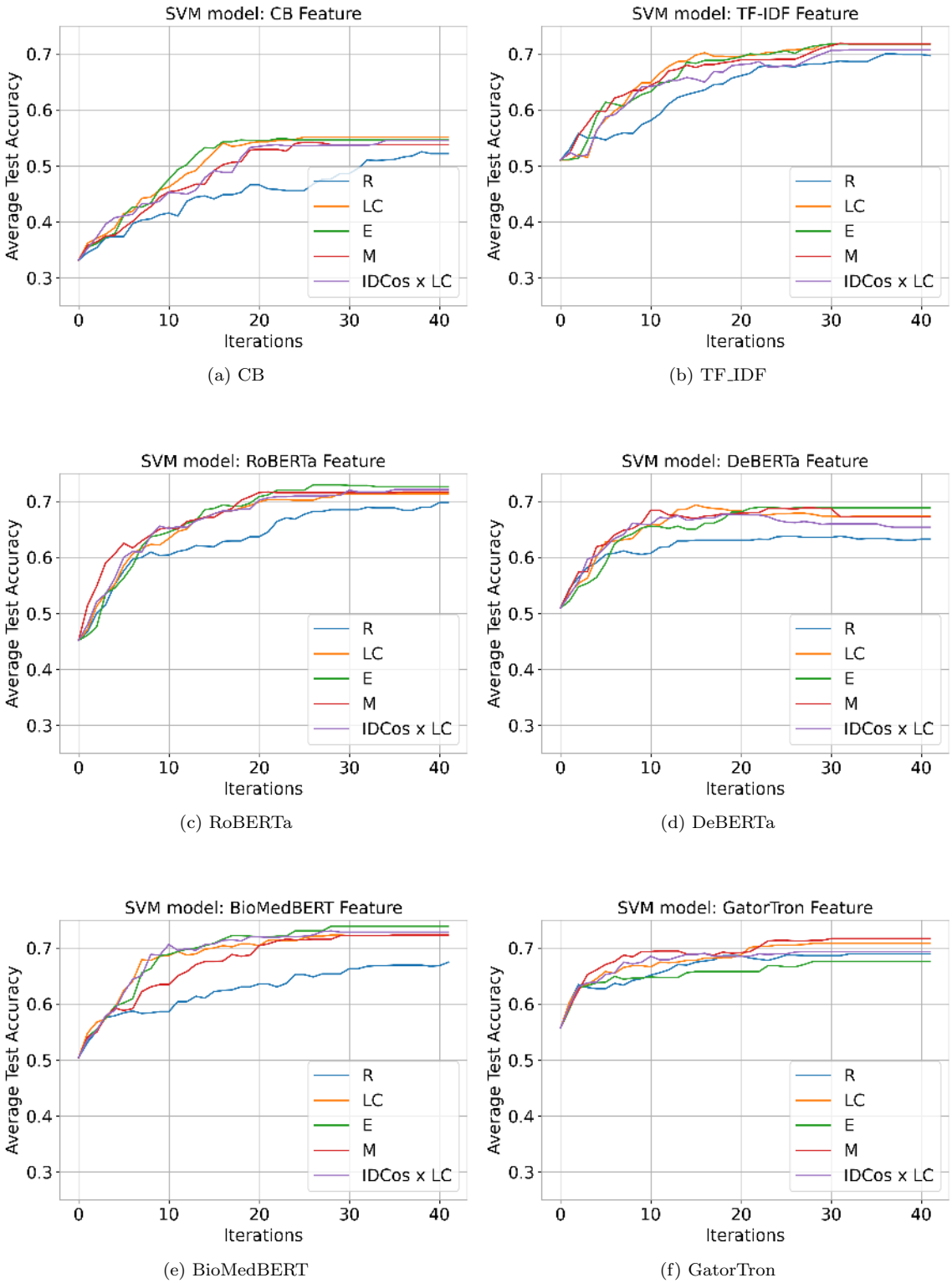
4.6.1 Evaluation using Accuracy

The performance of ML classifiers, using various AL strategies, was evaluated over 41 iterations by analyzing average test set accuracy until all unlabelled samples were used (see Figures 4.3 and 4.4).

To compare the effectiveness of AL strategies against the random baseline R, we report the test set accuracy achieved at the 20th iteration, as well as average improvements over R at this iteration and between 10-30 iterations (see Table 4.3).

BioMedBERT features yielded the highest accuracy at 20 iterations, with SVM and LR achieving 0.72 and 0.717, respectively, surpassing the R baseline by 0.084 and 0.039 using both E and $IDCos \times LC$ strategies. Following BioMedBERT, RoBERTa, TF-IDF, and GatorTron features reported high accuracies, ranging between 0.692-0.716 for SVM and 0.701-0.716 for LR classifier. The static TF-IDF representation performed comparably to the clinical domain-specific GatorTron embeddings, indicating that static representations can sometimes match the performance of deep learning models. At the 20th iteration, for SVM, LC strategy improves accuracy by 0.0493 on average; while for LR, $IDCos \times LC$ strategy increases accuracy by 0.0426 on average, over the R baseline across all features.

Static CB features reported the lowest accuracy at the 20th iteration, reaching 0.545 with SVM



(a) CB

(b) TF_IDF

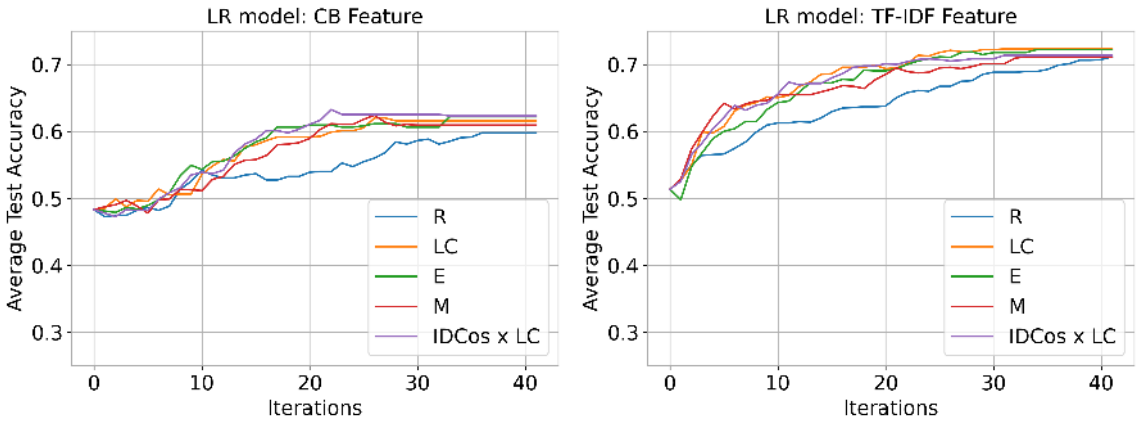
(c) RoBERTa

(d) DeBERTa

(e) BioMedBERT

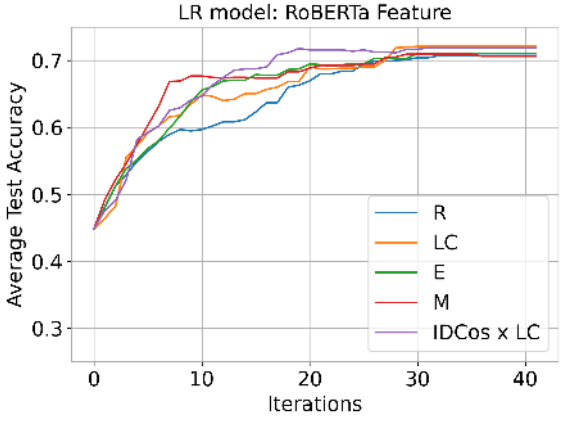
(f) GatorTron

Figure 4.3: SVM Average Test Accuracy Improvements Across Text Representations

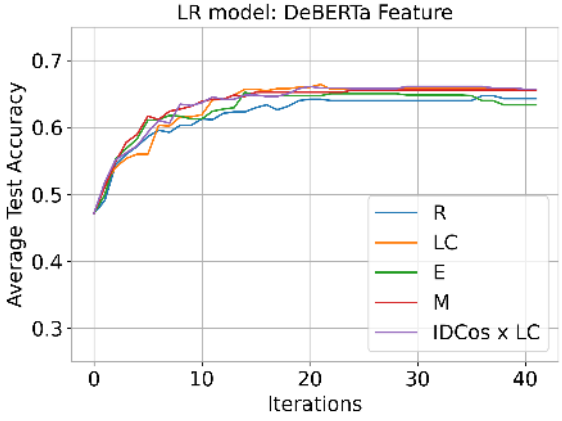


(a) CB

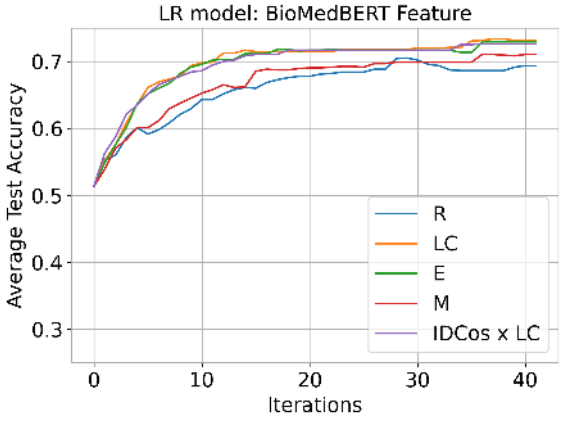
(b) TF_IDF



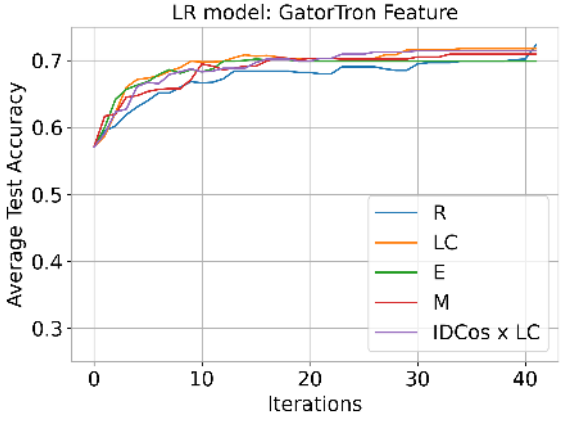
(c) RoBERTa



(d) DeBERTa



(e) BioMedBERT



(f) GatorTron

Figure 4.4: LR Average Test Accuracy Improvements Across Text Representations

Table 4.3: Comparison of Average Accuracy for SVM and LR using Active Learning

AL Strategy	**SVM**			**LR**		
	A_{20}	ΔA_{20}	ΔA_{10-30}	A_{20}	ΔA_{20}	ΔA_{10-30}
CB						
R	0.466	0	0	0.539	0	0
LC	0.543	0.077	0.076	0.592	0.053	0.044
E	**0.545**	**0.079**	**0.082**	0.609	0.071	0.049
M	0.529	0.063	0.056	0.589	0.051	0.037
IDCos × *LC*	0.535	0.068	0.055	**0.611**	**0.072**	**0.055**
TF-IDF						
IR	0.661	0	0	0.638	0	0
LC	**0.696**	**0.035**	**0.044**	0.694	0.056	**0.052**
E	0.695	0.034	0.037	0.691	0.053	0.044
M	0.689	0.028	0.033	0.685	0.047	0.033
IDCos × *LC*	0.681	0.02	0.02	**0.701**	**0.063**	0.049
RoBERTa						
R	0.637	0	0	0.669	0	0
LC	0.699	0.062	0.041	0.691	0.021	0.018
E	0.708	0.072	**0.054**	0.695	0.025	0.029
M	**0.716**	**0.079**	0.05	0.688	0.019	0.03
IDCos × *LC*	0.702	0.065	0.044	**0.716**	**0.046**	**0.044**
DeBERTa						
R	0.631	0	0	0.642	0	0
LC	**0.683**	**0.053**	0.049	**0.66**	0.018	**0.021**
E	0.681	0.051	0.043	0.647	0.005	0.011
M	0.679	0.048	**0.051**	0.653	0.011	0.018
IDCos × *LC*	0.676	0.045	0.039	**0.661**	**0.019**	0.02
BiomedBERT						
R	0.636	0	0	0.678	0	0
LC	0.704	0.068	0.077	0.715	0.037	**0.04**
E	**0.72**	**0.084**	**0.088**	**0.717**	**0.039**	0.038
M	0.704	0.068	0.062	0.691	0.013	0.009
IDCos × *LC*	**0.72**	**0.084**	0.085	**0.717**	**0.039**	0.037
GatorTron						
R	0.686	0	0	0.682	0	0
LC	0.687	0.001	0.013	**0.703**	**0.021**	**0.022**
E	0.658	-0.028	-0.018	0.699	0.017	0.015
M	**0.692**	**0.005**	**0.022**	**0.703**	**0.021**	0.016
IDCos × *LC*	0.685	-0.001	0.01	0.699	0.017	0.018

Note: This table compares various AL strategies against the random baseline (R) for both SVM and LR models, using three metrics: A_{20} (accuracy at the 20th iteration), ΔA_{20} (accuracy improvement over R at the 20th iteration), and ΔA_{10-30} (average accuracy improvement over R between the 10th and 30th iteration). The representation achieving highest accuracy at 20th iteration is highlighted in yellow.

and 0.611 with LR, using E and $IDCos \times LC$ strategies, respectively. However, the AL strategies still significantly outperformed the R baseline, with improvements of 0.079 and 0.072 by 20 queries. This demonstrates the effectiveness of AL strategies across both static and deep-learning text representations, with highest accuracy gains seen with domain specific embeddings.

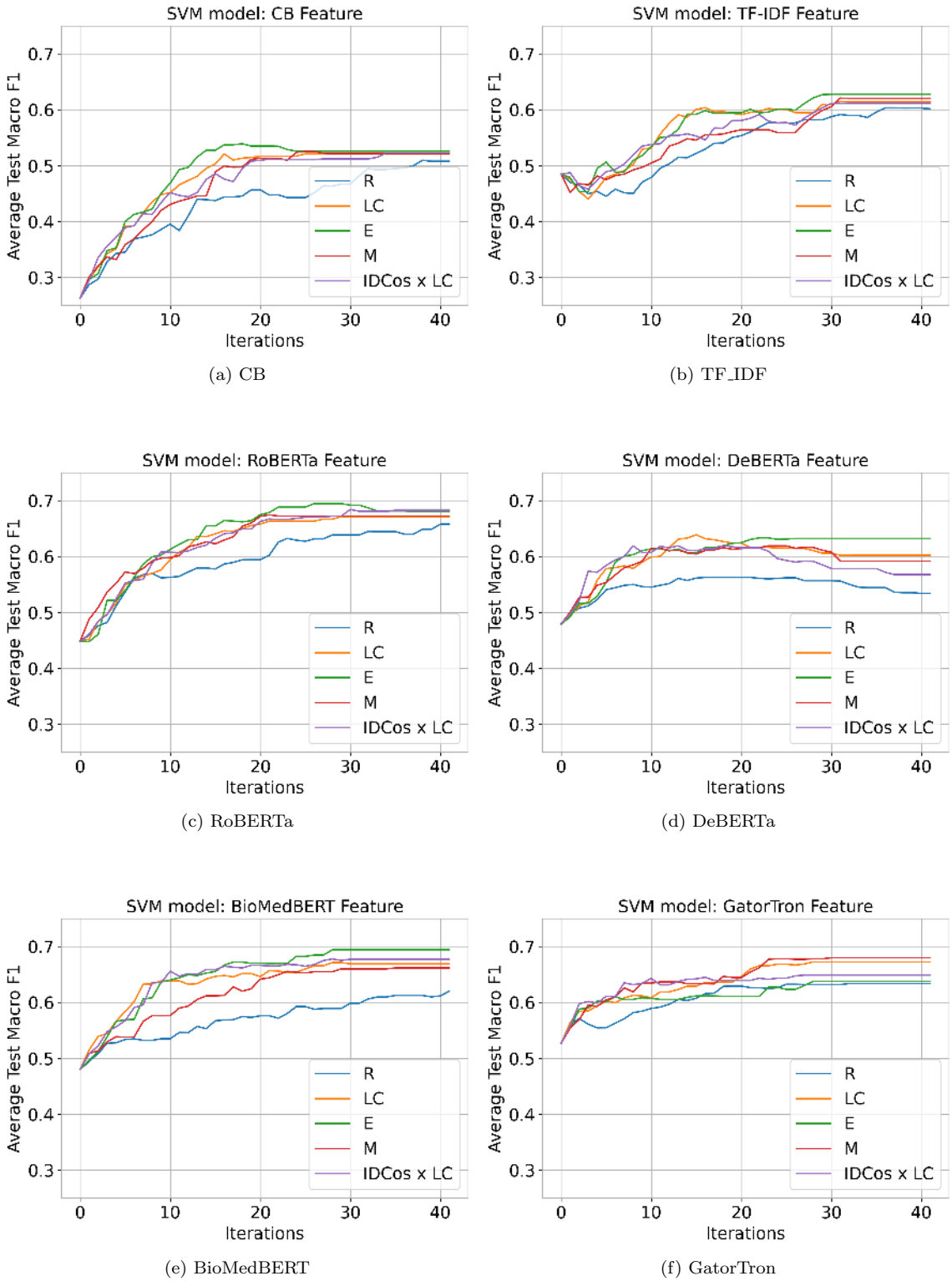
4.6.2 Evaluation using Macro F-score

In addition to accuracy, we analyze the macro F-score on the test set as more instances are sampled (see Figures 4.5 and 4.6) and report corresponding statistics in Table 4.4. The macro F-score can be considered a more appropriate measure for imbalanced datasets as it evaluates precision and recall for each class, assessing average performance across all classes.

In terms of macro F-score, RoBERTa, BioMedBERT, and GatorTron features perform well, achieving between 0.646-0.675 at the 20th iteration using the SVM model, and 0.667-0.679 with the LR model. These results show improvements of 0.017-0.094 and 0.048-0.053 over Random (R) sampling for SVM and LR models respectively. DeBERTa, TF-IDF and CountBinary follow, with static features reporting the lowest F-scores at 20th iteration, ranging between 0.535-0.595 with SVM and 0.579-0.594 with LR, demonstrating that all contextual features outperform static features in terms of macro F-score. At the 20th iteration, for SVM, the E strategy improves macro F-score by 0.0563 on average; while for LR, the $IDCos \times LC$ strategy increases macro F-score by 0.0542 on average. These strategies represent the highest improvements in macro F-score over the R baseline when averaged across all features.

4.7 Discussion

Many hospitals across the world use incident reporting systems and the data collected on PSEs play a significant role in quality improvement efforts [68]. Reliable classification of PSE types is crucial for patient safety, and while ML classifiers can accurately classify PSE-reports [172, 30, 29], they typically require large amount of labelled data to perform effectively. Our study investigated how AL techniques can minimize labeling efforts by utilizing a smaller subset of informative data points. We evaluated the effectiveness of four AL strategies in improving ML model performance for classifying event types in PSE reports. Additionally, we compared the impact of static versus contextual text representations and generic versus domain-specific embeddings.



(a) CB

(b) TF_IDF

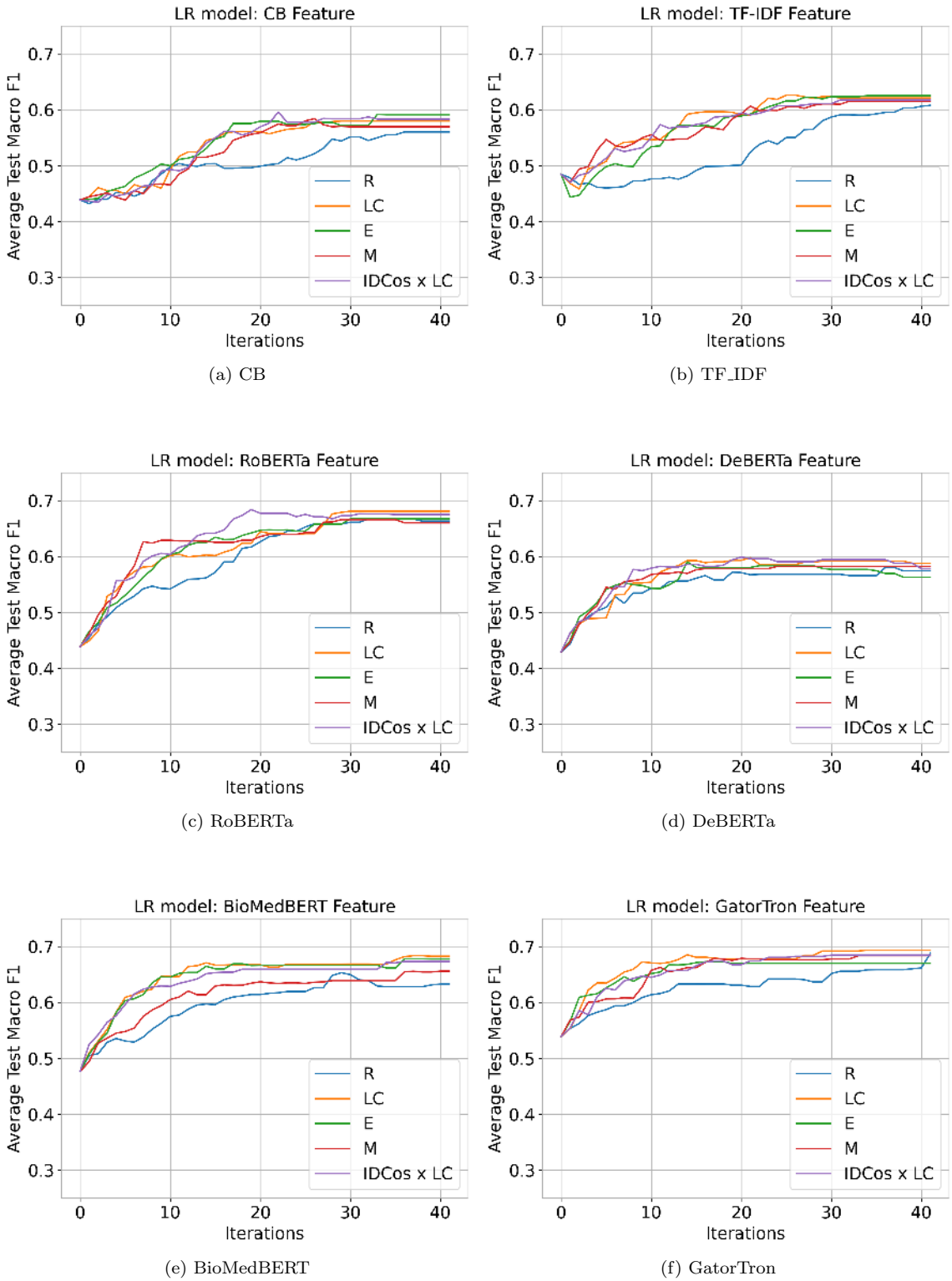
(c) RoBERTa

(d) DeBERTa

(e) BioMedBERT

(f) GatorTron

Figure 4.5: SVM Average Test Macro F-score Improvements Across Text Representations



(a) CB

(b) TF_IDF

(c) RoBERTa

(d) DeBERTa

(e) BioMedBERT

(f) GatorTron

Figure 4.6: LR Average Test Macro F-score Improvements Across Text Representations

Table 4.4: Comparison of Average Macro F-score for SVM and LR using Active Learning

AL Strategy	**SVM**			**LR**		
	F_{20}	ΔF_{20}	ΔF_{10-30}	F_{20}	ΔF_{20}	ΔF_{10-30}
CB						
R	0.457	0	0	0.499	0	0
LC	0.516	0.06	0.065	0.561	0.062	0.045
E	**0.535**	**0.078**	**0.082**	**0.579**	**0.081**	**0.049**
M	0.513	0.057	0.053	0.559	0.06	0.036
$IDCos \times LC$	0.509	0.053	0.05	0.57	0.071	0.048
TF-IDF						
R	0.554	0	0	0.501	0	0
LC	0.591	0.038	**0.045**	0.591	0.09	**0.08**
E	**0.595**	**0.041**	0.044	0.589	0.088	0.071
M	0.564	0.011	0.009	0.592	0.091	0.064
$IDCos \times LC$	0.581	0.027	0.024	**0.594**	**0.093**	0.072
RoBERTa						
R	0.594	0	0	0.626	0	0
LC	0.658	0.064	0.046	0.645	0.018	0.017
E	**0.675**	**0.081**	**0.065**	0.647	0.021	0.027
M	0.672	0.078	0.047	0.636	0.009	0.025
$IDCos \times LC$	0.663	0.069	0.046	**0.677**	**0.051**	**0.047**
DeBERTa						
R	0.563	0	0	0.571	0	0
LC	0.624	0.061	0.06	0.593	0.022	0.024
E	**0.625**	**0.062**	**0.064**	0.58	0.009	0.013
M	0.614	0.052	0.055	0.578	0.007	0.015
$IDCos \times LC$	0.616	0.053	0.049	**0.599**	**0.028**	**0.026**
BiomedBERT						
R	0.576	0	0	0.614	0	0
LC	0.646	0.07	0.08	**0.667**	**0.053**	**0.054**
E	**0.67**	**0.094**	**0.097**	0.666	0.052	0.051
M	0.642	0.066	0.058	0.637	0.023	0.017
$IDCos \times LC$	0.666	0.09	0.092	0.659	0.045	0.042
GatorTron						
R	0.629	0	0	0.631	0	0
LC	**0.646**	**0.017**	0.027	0.678	0.047	**0.046**
E	0.611	-0.018	-0.002	0.67	0.039	0.036
M	0.644	0.015	**0.035**	**0.679**	**0.048**	0.039
$IDCos \times LC$	0.639	0.01	0.023	0.668	0.037	0.038

Note: This table compares various AL strategies against the random baseline (R) for both SVM and LR models, using three metrics: F_{20} (macro F-score at the 20th iteration), ΔF_{20} (improvement in macro F-score over R at the 20th iteration), and ΔF_{10-30} (average macro F-score improvement over R between the 10th and 30th iteration). The representation achieving highest macro F-score at 20th iteration is highlighted in yellow.

AL strategies vs. random sampling (R) Our analysis revealed that most AL strategies significantly outperformed random selection (R) in terms of both accuracy and macro F-score. AL strategies enhance labeling efficiency across all text representations, achieving near peak performance within 10 to 30 iterations (see Figures 4.3, 4.4, 4.5, 4.6), reducing the required number of labelled samples by 24-69%, considering the total number of samples at the last iteration is 438. These strategies lead to highest accuracy gains particularly for domain-specific embeddings, notably BioMedBERT, improving SVM and LR accuracy to 0.72 and 0.717 respectively, and macro F-score to 0.67 and 0.667 within 20 iterations, using only 54% of PSE-reports. Hence, integrating AL into PSE-report classification can streamline labeling, reduce workload for PSE analysts, and optimize both classifier performance and resource allocation. It will enable analysts to focus on annotating fewer incidents for model development, fostering human-AI collaboration.

Best AL strategy Among the AL strategies, Entropy (E) and the combination of Information Density (ID) Cosine with Least Confidence ($IDCos \times LC$) were notably effective for SVM and LR, respectively in terms of both accuracy and macro F-score. This demonstrates the efficacy of AL strategies, particularly when integrated with representativeness measures such as ID.

Static vs. contextual representations For both ML models, static CB representations consistently underperformed other features in all experiments. However, static TF-IDF features sometimes matched or even surpassed the performance of contextual DeBERTa and GatorTron embeddings. Therefore, we cannot definitively state that contextual text representations always outperform static ones. Nonetheless, contextual features can represent text more efficiently in lower-dimensional spaces than TF-IDF enabling faster training, often with comparable or superior classification performance.

Generic vs. domain-specific representations BioMedBERT domain-specific contextual embedding slightly outperforms generic English RoBERTa embeddings by 20 iterations in accuracy, which may be attributed to their ability to better capture medical terminology present in the text. With SVM, BioMedBERT achieves the largest overall improvement from the R baseline between 10-30 iterations, leading to a 0.078 increase in accuracy and an 0.082 increase in macro F-score, averaged across all AL strategies. However, with LR, static TF-IDF features yield the highest overall improvement from R, increasing accuracy by 0.045 and macro F-score by 0.072. Thus, we conclude, while domain specific contextual embeddings generally lead to better performance compared to static text representations, they do not necessarily result in larger improvements using AL strategies across all models. Interestingly, when SVM is used with GatorTron features, the random baseline R

performs relatively well, even outperforming the Entropy strategy and performing comparably with other AL strategies.

Regardless of the text features used, AL strategies prove to be effective tools in helping improve labelling efficiency over random sampling. By 20 iterations of AL sampling, we observe average improvements over the random baseline of 0.0483 in accuracy and 0.0512 in macro F-score for SVM, and 0.0348 in accuracy and 0.0477 in macro F-score for LR, across all features.

4.7.1 Limitations

The ML models developed in our study were trained using PSE reports from maternal care units of one hospital in the United States, potentially limiting the generalizability of the results to different settings. Additionally, the scope of the analysis was limited by the small quantity of PSE reports available. Only the seven most frequent classes were used for this study. Thus, by not accounting for rarer event types, the performance of our models might be overestimated. We recommended that future research assess the efficacy of ML models integrated with AL strategies over more extensive and disparate datasets.

4.8 Conclusion

PSE labelling is particularly challenging as it requires specialized knowledge and human intervention, making full automation difficult. AL aims to support rather than replace manual labeling by filtering instances that are most likely to improve classification accuracy. AL also helps address the imbalanced-data problem typical of PSEs, by prioritizing rare or difficult-to-classify incidents, thus learning from a diverse incident set [7]. PSE reports can change over time due to evolving medical practices and emerging safety issues. AL systems can adapt to these changes more effectively than static models, as they can incrementally learn and re-train from the newly labelled data [128]. Therefore, AL emerges as an essential tool in the classification of PSE-reports, offering more insightful, efficient, and effective use of this critical data in improving patient safety.

Chapter 5

Summary

5.1 Concluding Remarks

In this thesis, we evaluate the efficacy of machine learning models for automating the classification of Patient Safety Event (PSE) reports in healthcare settings. Initially in Chapter 3, we assess decision tree ensemble models like XGB and LGBM, using them to rank incidents by severity according to the Healthcare Performance Improvement (HPI) SEC code.

As PSE reports predominantly contain textual descriptions, we apply various Natural Language Processing techniques ranging from traditional bag-of-words to advanced deep-learning models like BERT, including domain-specific versions such as GatorTron, pretrained on clinical datasets. We compare these models using both query-dependent and query-independent learning-to-rank strategies (see section 3.8) to assess their performance in incident severity ranking.

The thesis also addresses the challenges posed by the limited availability of labeled data in PSE reports—as a result of the high volume of reports and complex classification taxonomy which makes manual classification labor-intensive and costly [94, 102, 43]. In Chapter 4, we explore the potential of Active Learning (AL) strategies to enhance labeling efficiency of PSE reports. These strategies prioritize labeling a smaller subset of incidents, which significantly boosts the performance of ML classifiers with fewer labeled samples, achieving near-peak performance earlier than random sampling, as demonstrated by improved accuracy and macro F-score metrics (see section 4.6).

Overall, this work aims to reduce the workload on Patient Safety Analysts (PSAs) by developing ML models that support the PSE report classification process. By integrating state-of-the-art NLP techniques with real-world PSE data, we explore the feasibility of automating PSE report classifica-

tion and support decision-making, thus enhancing human-AI collaboration, improving classification efficiency in the triage of severe events, and ultimately improving patient care and safety.

5.2 Summary of Contributions

The main contributions of this thesis are summarized as follows.

5.2.1 Learning to Rank PSE reports by Severity (Chapter 3)

1. We leverage tree-based ensemble models, specifically Extreme Gradient Boosting (XGB) and Light Gradient Boosting Machine (LGBM), considering both query-independent and query-dependent Learning-to-Rank (LTR) strategies in ranking patient safety event reports by severity. To our knowledge, this is the first work to implement LTR models for PSE report classification. We further analyze the effectiveness of these models across various static and deep-learning based contextual text representations.

2. Our evaluation reveals that the query-independent LTR strategy using the LGBM classifier with healthcare domain-specific GatorTron features, significantly outperforms the existing baseline. This model can assist PSAs in efficiently identifying potential critical incidents, thereby reducing the workload of manual screening and streamlining the review process.

3. We demonstrate that domain-specific embeddings, such as GatorTron, BioGPT, and BioMedBERT, predominantly surpass traditional NLP representations across ranking metrics. This underscores the benefits of utilizing healthcare-domain specific contextual embeddings to enhance classification accuracy in PSE analysis.

5.2.2 Evaluating Active Learning Strategies for PSE Type Classification (Chapter 4)

1. We simulate pool-based Active Learning with Support Vector Machine (SVM) and Logistic Regression (LR) models utilizing four Active Learning (AL) strategies, including three uncertainty-based and one combining information density with uncertainty. We compare and evaluate the performance of these strategies to a random selection baseline across various text representations, focusing on accuracy and Macro F-score improvements through iterative data sampling.

2. Our experiments reveal that AL strategies significantly outperform random sampling in both accuracy and Macro F-score, reducing the need for labeled samples by 24-69%. Notably, BioMedBERT, a medical domain-specific language model, achieves the highest accuracy gains within 20 iterations for both SVM and LR models, utilizing only 54% of PSE reports. This underscores the efficacy of domain-specific models in enhancing PSE report labeling and reducing manual workload.

3. Among the AL strategies, we find Entropy and Information Density (ID) Cosine with Least Confidence were particularly effective for SVM and LR models, respectively, in terms of both accuracy and Macro F-score. This demonstrates the efficacy of AL strategies, particularly when integrated with representativeness measures such as ID.

Some of the results from the work in this chapter have been accepted for publication [79].

5.3 Future Work

This section suggests several research directions and potential future work stemming from the findings of this thesis.

5.3.1 Fine-tuning Language Models

In our experiments, we only performed feature extraction to obtain sentence embeddings using pre-trained language models such as RoBERTa, BioMedBERT, and GatorTron. These embeddings were then used to train separate learning-to-rank models. Due to computational constraints and significant data imbalance, we did not extend to fine-tuning the language models. Some studies indicate that fine-tuning can worsen performance when pre-trained features are good and distribution shifts are large, distorting the embedding space [95, 1, 117].

Fine-tuning BERT may or may not improve performance depending on the specific downstream task [129]. While some studies report that pre-trained embeddings outperform fine-tuned ones [100, 57], others find fine-tuning effective for medical code assignment or text classification tasks [114, 101, 143, 13]. Therefore, future research should explore fine-tuning language models on specific patient safety event (PSE) datasets.

This can be achieved by adding one or more neural network layers, called the classification head, on top of the BERT model with a cross-entropy loss function. This approach would replace the supervised ML classifiers currently in use, such as Gradient Boosted Machines or Support Vector

Machines, allowing the embedding space to update based on classification performance on the training set. Fine-tuning may enable models to adjust to the specific language nuances in the dataset, potentially enhancing their performance in classifying PSE reports.

5.3.2 Zero-shot and Few-shot Learning

In our experiment, we proposed Active Learning strategies to address the challenge of limited labeled PSE reports, which are often insufficient for training machine learning models. Another promising approach is zero-shot and few-shot learning, which allows models to make predictions with limited or no labeled training data [32], particularly useful when obtaining labeled data is costly or impractical. These approaches have proven effective in text classification tasks across various domains, including medical text classification, as demonstrated in studies by [90, 109, 139, 182, 22, 28, 153, 154]. Future research can leverage pre-trained transformer models in combination with zero-shot and few-shot learning [14], to classify PSE report.

Two common approaches to zero and few-shot learning are entailment and similarity:

1. **Entailment-based Zero-shot Learning:** This involves transformer-based language models pre-trained on large entailment datasets (e.g., MNLI), consisting of premise (e.g. input PSE reports) and hypothesis (e.g. PSE type or severity label). Models learn to determine if the premise entails the hypothesis. Labels are converted to hypothesis statements by completing their interpretation (e.g., from "medication incident" to "This report specifies a medication incident type") or by defining the labels (e.g. from "medication incident" to "Medication incidents are safety events related to administering incorrect type or dose of medication..."). The pre-trained entailment model then predicts output probabilities indicating the likelihood that the premise entails the hypothesis for each input and label pair. The label with the highest entailment probability is assigned to the report.

2. **Similarity-based Learning:** This approach classifies instances based on similarities between document representations and class description representations within the same embedding space [142]. For zero-shot similarity-based learning, the distance between reports and labels in the embedding space are computed using metrics like cosine or Euclidean distance, choosing the label with the smallest distance. For few-shot similarity-based learning, where a small number of labeled examples are available, the distance in the embedding space between an input sentence and the labeled examples determines the class labels for the input sentence.

Another popular few-shot learning approach is fine-tuning pre-trained models using a few labeled samples. For example, the SetFit (Sentence Transformer Fine-tuning) model [163] fine-tunes pre-trained sentence transformers on a small number of text pairs in a contrastive Siamese manner.

While zero-shot and few-shot approaches show efficacy in text classification tasks, applying these techniques to PSE report classification can be challenging. The effectiveness of these approaches heavily relies on the model's ability to generalize from its training on general tasks to the specific domain of PSE reports. Challenges arise due to the complex classification taxonomy and specialized nomenclature used by patient safety specialists that may not be present in the pre-training data.

5.3.3 Prompt Engineering using LLMs

With advancements in large language models (LLMs), prompt engineering has become a prominent technique in NLP. It involves designing and optimizing prompts (i.e., instructions written in natural language) to enhance model performance on specific tasks, showing significant superiority across various domains, including healthcare [169]. Prompting techniques obtain desired responses from LLMs such as ChatGPT, Google Bard, Claude, or LLaMA2, by altering the input prompt without changing the model's architecture [162]. This approach involves converting a text input into a structured template, asking a fine-tuned language model to complete the template, and then translating the model's output into a predicted class [160].

Several studies have successfully used LLMs with prompting techniques for medical text classification [174, 108, 160, 181], even outperforming few-shot learning approaches [108]. Recent work has seen the rise of healthcare domain-specific LLMs for prompting, such as HealthPrompt [153] and ERNIE-Health [174]. Therefore, future research could explore different prompting techniques for PSE report classification. It can be particularly useful for this task, as prompts can include additional context, such as definitions or examples of each severity level or event type, addressing domain-specific challenges like complex medical terminology and clinical contexts [181], present in PSE reports.

Furthermore, future work can explore chain-of-thought prompting [176], which requires the model to list reasons or steps leading to a classification, thus providing a logical reasoning process before providing the answer. This is particularly useful for complex decision-making tasks like PSE report classification.

Appendix A

Machine Learning Algorithms for Text Classification

A.1 Support Vector Machines (SVM)

Support Vector Machines (SVM) is most commonly used for text classification problems, as it tends to have better generalization capability on unseen data compared to other similarity-based learning algorithms [138]. SVM classifiers are supervised learning model that aims to find the optimal hyperplane (by projecting the feature vector into N-dimensional space), separating the data points into two spaces: one for the vectors which belong to the given class and one for the vectors which do not belong to it. Assume that the training set consists of N vectors $x_i (i = 1, 2, \cdots, N)$ from the n-dimensional feature space $X \in R^n$ [62]. Each vector x_i has an associated target $y_i \in \{-1, +1\}$ representing the two different classes. To solve a linear problem (where classes linearly separable), SVM would search for a line separating the the two classes called the hyperplane. For non-linear problems, SVM would first map the vectors x_i to a high dimensional space using a kernel function (e.g linear, polynomial, radial basis function) to find the best hyperplane [125], transforming X into $\Phi(X) \in R^{n''}$, where Φ is the chosen kernel function [62].

Then the hyperplane can be written as:

$$f(x) = w\Phi(x) + b$$

The data points in each class near the hyperplane are called the support vectors. The optimal

hyperplane is the one that maximizes the distance between the hyperplane and the closet data point, called the margin [62]. "A large margin is preferred to reduce the odds of misclassification and to prevent over-fitting the data" [138]. Along with maximizing the margin M, SVM often allow some number of observation C to be misclassified (to be on the wrong side or margin or wrong side of hyperplane) and thus optimizes to find a soft margin. This is to achieve greater robustness and better classification of most of the training observations x_i. The misclassification error ϵ_i for each data point, called slack variable, measure the distance of the data point from the margin of the class it belongs to. The sum of these errors must be $\leqslant C$.

In Chapter 9 of "An Introduction to Statistical Learning", James et al. outlines the optimization of SVM classifiers as follows [81]:

$$\underset{\beta_0,\beta_1,...,\beta_p,\epsilon_1,...,\epsilon_n}{\text{maximize}} \text{ M subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$$

where, $\beta_0, ..., \beta_p$ is the bias and weights of the hyperplane

Although SVM is a binary classifier, it can perform multi-class classification as well, using either One-vs-one and One-vs-All approach [59]. While both approach breaks down the multi-class problem into multiple binary classification problems, the way they define the binary separation differs.

- In the One-vs-All (OVA) strategy the binary classifier finds a hyperplane that separates one class from all other classes. Assuming there are K classes in a problem and $K > 2$, K two-class SVM classifiers will be constructed [59]. The maximum of these classifier scores is then used to predict the final class. Since each of the K classifiers is trained using all available samples, this approach slower and computationally expensive

- The One-vs-One (OVO) method constructs a binary classifier for every pair of classes in a multi-class problem, each predicting a class label. This results in $K(K-1)/2$ binary classifiers for K classes, where $K > 2$ [59]. Thus, in the OVO strategy the final classification is done by selecting the class with the majority of 'votes', across these binary classifiers. Since each classifier only require training samples from two of all K classes, the number of samples used to train the classifier is much smaller compared to OVA resulting in faster training time. However, it may result in slower training time when number of the classes in the problem is large, since

every test sample has to be presented to large number of classifiers N(N-1)/2 [59]

A.2 Logistic Regression

Logistic Regression is one of the most widely used data mining methods, especially for binary classification problems such as sentiment analysis of tweets. It is a statistical method that learns the significance of each independent predictor variable x_i in determining the probability of an outcome y_i [147].

For instance let's take the sentiment classification task on tweets as an example. Here each word or phrase from a tweet, known as tokens, is represented as an independent variable x_i within a set of features $[x_1, x_2, ..., x_n]$, where n denotes the number of unique tokens in the vocabulary used to represent the tweet. These features are then used as models inputs, optimizing their corresponding weights $w_i \in [w_1, w_2, ..., w_n]$, to predict the sentiment $\hat{y} \in [0, 1]$. Here, $\hat{y} = 1$ indicates a positive sentiment, and $\hat{y} = 0$ a negative sentiment. For multi-class classification problems, Multinomial logistic regression extends this framework to classify text into multiple classes, with $\hat{y} \in \{0, 1, ..., m\}$, where m denotes the total number of classes.

The logistic function for a binary classification with multiple features is defined as [85]:

$$z = \sum_{i=1}^{n} w_i x_i + b = \mathbf{w}.\mathbf{x} + b$$

where,

- where z is the linear combination of the feature values x_i and their respective importance denoted by weights w_i
- b represents the model's bias,
- n denotes the number of unique features to represent an instance
- and $\mathbf{w}.\mathbf{x}$ is the dot product of all weights w_i and features x_i of an instance

To constraint the classifier output to be in the probability range $[0, 1]$, z is further passed into a sigmoid function $\sigma(z)$. This function produces the probability $P(y|X_i)$ of a particular class y given the feature vector $X_i = [x_0, x_1, ..., x_n]$ of an instance [85]:

$$P(y|X_i) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

In case of binary classification,a threshold t determines the predicted class $\hat{y}$ of an instance [85]:

$$\hat{y} = \begin{cases} 1, & if P(y|X_i) > t \\ 0, & \text{otherwise} \end{cases}$$

The threshold t typically ranges from [0,1], with 0.5 being a common default, signifying equal probability of belonging to either class.

For multi-class classification, a multinomial logistic regression model outputs a set of probabilities indicating the likelihood that a given observation belongs to each class y_i. For each instance X_i, probabilities $P(y_i|X_i)$ are calculated for all classes y_i, where $y_i \in [y_1, ...y_m]$ and m is the total number of classes. One approach is to apply the softmax function to these probabilities to ensure they sum to one. The class with the highest probability is then assigned to each observation [147].

A.3 Decision Trees

Decision Tress are a non-parametric supervised machine learning model that can be used for both classification (for categorical variables) and regression tasks (for continuous variables) [155]. The model aims to learn if-then rules from the data, and applies them to predict the target class or value. The model's decision path follows a tree-like structure which begins at the root node containing all the samples and then splitting the node based on some binary decision, that results in two nodes containing homogeneous sets of samples in each [155]. This process is done recursively splitting each of the subsequent nodes based on some binary decision rules, ultimately resulting in leaf nodes containing samples belonging to a specific class. To prevent the model from over-fitting the training samples the tree is often pruned by removing the nodes that provide less information [155].

In order to build an efficient decision tree, the nodes need to be split based on input features that are related to the target variable such as the class [155]. To identify the most important features used to split the nodes, methods such as entropy, Gini index, classification error, information gain, or gain ratio are computed to analyze node purity. Node purity can be found by computing the probability $p(c_i)$ of class c_i in a node (i.e., the fraction of nodes belonging to the target class). High purity nodes consists high proportion of samples that belong to one specific class and thus are pure. Pure nodes have high information gain or low entropy (randomness) therefore leading to more confident predictions [155].

The formulas for each method is as follows [47]:

$$\text{Gini} = 1 - \sum_{i=1}^{n} p^2(c_i)$$

$$\text{Entropy} = \sum_{i=1}^{n} -p(c_i) log_2(p(c_i))$$

where, $p(c_i)$ is the probability/percentage of class c_i in a node

Information Gain (IG):

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N} I(D_{left}) - \frac{N_{right}}{N} I(D_{right})$$

where,

f: feature split on

D_p: dataset of the parent node

D_{left}: dataset of the left child node

D_{right}: dataset of the right child node

I: impurity criterion (e.g. Gini Index or Entropy)

N: total number of samples

N_{left}: number of samples at the left child node

N_{right}: number of samples at the right child node

A.4 Ensemble Model

Ensemble models aim to combine the predictions of several base classifiers in order to improve the generalizability and predictive performance over a single classifier. The two fundamental ensemble techniques are Bagging (also known as Bootstrap Aggregating) and Boosting. While bagging methods train multiple model independently taking the majority vote as the final prediction, boosting involves sequential ensemble algorithms that iteratively refine models by adjusting their weights based on previous prediction errors [16, 39]. Here we describe two widely used ensemble methods: Random Forest (RF), a bagging ensemble, and Extreme Gradient Boosting (XGBoost), a boosting algorithm.

- **Random Forest**

  Random forest models have demonstrated strong performance on text classification tasks across

various domains, and are notably effective for handling imbalanced datasets [148, 80, 11]. It is an example of bagging ensemble technique, developed by Brieman in 2001 [12]. It involves training multiple decision trees in parallel on a randomly chosen subset of data and features. These subsets are created through bootstrap sampling, which means randomly selecting samples from the training dataset with replacement [72]. The predictions from each tree are aggregated for the final prediction, using majority voting for classification or averaging for regression tasks. Since each model is trained independently, bagging ensemble methods are less prone to overfitting to the noise in the training data. The variance of the Random Forest estimator, can be further reduced by selecting a random subset of features for node splitting during tree construction, rather than using all features. Typically, the size of this random subset is the square root of the total number of features [72]. Since, individual decision trees tend to overfit, the randomness from data and feature sampling result in trees with less correlated errors, reducing the variance, sometimes at the cost of a slight increase in bias [12]. In practice, reducing variance often leads to significant improvements, resulting in a generally better model.

- **Extreme Gradient Boosting**

  Gradient Boosting Machines (GBMs), are boosting ensemble techniques that sequentially fit new models to provide a more accurate estimate of the target variable [121]. At each particular iteration, a new weak base-learner model is trained with respect to the error of the whole ensemble learnt so far [121]. The error denotes the gradient of the loss function with respect to the model prediction, and thus can be generalized, allowing optimization of an arbitrary differentiable loss function [49]. The new models are added to the ensemble such that they are maximally correlated with the negative gradient of the loss function associated with the ensemble [121].

  Extreme Gradient Boosting (XGBoost) is an improved and scalable extension of GBMs that uses Gradient Boosting Trees and was first proposed by Tianqi Chen and Carlos Guestrin in 2016 [33]. It has been applied to solve many classification and regression problems in diverse fields. To prevent overfitting a regularization term is added to the loss function. The XGBoost objective function at the t-th iteration of the gradient boosting process, denoted as $\mathcal{L}^{(t)}$, is defined as follows:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

  where:

- n is the number of instances in the training dataset,
- $l(y_i, \hat{y}_i^{(t-1)})$ is the differentiable convex loss function that measures the difference between the true value y_i and the predicted value $\hat{y}_i$ from the previous iteration $(t-1)$,
- $f_t(\mathbf{x}_i)$ is the prediction of the new tree model (i.e with improved tree structure and leaf weights) on the feature vector $\mathbf{x}_i$,
- $\Omega(f_t)$ represents the regularization term which penalizes the complexity of the model to prevent overfitting.

XGBoost aims to minimize this objective function, sequentially adding a new model f_t, improving the predictions by focusing on instances that were poorly predicted in the previous iteration. The inclusion of the regularization term $\Omega(f_t)$ helps to improve the generalization ability of the model. In literature, XGBoost models often demonstrates superior performance compared to other ML models such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Naive Bayes in text classification tasks, across various text feature representations, including TF-IDF and BERT [134, 25, 167].

Appendix B

Chap 4: Learning-to-Rank (LTR) Model

B.1 Learning to Rank (LTR)

With the rapid growth of the Web, Information Retrieval (IR) has become crucial, driving the need for efficient search engines (i.e. IR systems) to help users navigate and utilize the vast amount of data. Many IR problems such as document retrieval, collaborative filtering, sentiment analysis, and product rating [105] are by nature ranking problems, where the goal is to sort documents by relevance given a topic or query, ensuring the most relevant ones are ranked highest.

In IR, 'learning-to-rank' (LTR) refers to the task of training a machine learning model to sort query results by their relevance, preference, or importance using discriminative learning methods and extensive training data [105]. LTR is a kind of supervised learning, that requires a training set and corresponding relevance judgements (i.e. labels).

Over the years several "learning-to-rank" models have been proposed, including various techniques such as gradient descent, gradient boosting, as well as query-independent learning using multi-class or binary classifiers for ranking [105, 8, 17, 20, 54, 98, 119, 135, 180]. These algorithms can be broadly categorized into three approaches: pointwise, pairwise, and listwise [105]. Below, we provide explanations of each approach as described by Liu et al. [105]:

1. **Pointwise**: In the pointwise approach, each document is represented by a feature vector, and the objective is to predict the relevance degree of individual documents independently. The primary limitation of this approach is that it does not account for the relationships between

documents nor does it consider the relative positions of documents in the final ranking list. It also overlooks the fact the some documents are associated with the same query [105].

2. **Pairwise**: The pairwise approach compares pairs of documents to determine their relative order. Each pair is represented by their respective feature vectors, and a bi-variate function predicts which document in the pair is more relevant. The loss function here assesses the inconsistency between the prediction pairs against the ground truth pair of labels. However, this method only evaluates pairs in isolation and does not directly infer the final order of a list of documents, nor does it acknowledge that pairs may relate to the same query. Some examples of pairwise ranking algorithms include RankNet and FRank, RankBoost, and Ranking SVM [105].

3. **Listwise**: Unlike the previous approaches, the listwise approach considers an entire set of documents related to a specific query. This method utilizes multivariate functions to predict either the relevance of each document as a binary outcome or to directly rank the entire list. This approach is more effective compared to pointwise and pairwise methods since it accounts for the positions of all documents in a list associated with a specific query, resulting in a more comprehensive evaluation of document relevancy and order [105]. Examples of listwise ranking algorithms include SoftRank, AdaRank, ListNet, ListMLE, and LambdaRank [105, 18]

B.2 LTR with Tree-Based Ensembles

Ensemble models combine multiple learners, typically using majority voting to make the final classification [42]. These methods often outperform any single classifier, yielding more accurate results, particularly in text classification tasks [4, 165, 104]. Furthermore, ensemble models are capable of scaling effectively as the volume of data increases [4].

Li et al. proposed the use of ensemble boosting tree-based methods for ranking [98]. Boosting techniques uses the weighted average of a collection of weak learners [141] (i.e. shallow trees with depth of 1 or 2) to construct a strong learner by sequentially combining their predictions. Gradient Boosting Machines (GBM) are one of the most common techniques of the boosting ensemble [4].

LambdaMART [19], a tree boosting method for ranking [33], combines the MART algorithm [56] with the listwise ranking model LambdaRank [18], which is based on RankNet [19, 159]. This method learns the pairwise preferences of documents, while considering the NDCG gain derived from swapping document ranks in any given pair [159]. This makes it a listwise algorithm where the

cost is dependent on the sorted list of documents [159]. LambdaMART achieved state-of-the-art on benchmark datasets in LTR tasks, outperforming other algorithms such as RankNet [24, 105], and was the winning submission in the Yahoo! learning-to-rank challenge [159].

B.3 LTR using Query-Dependent Rankers

The goal of 'learning-to-rank' (LTR) using query-dependent models is to predict a relevance score for each document with respect to a given query q. The bipartite ranking problem involves learning a function f that assigns higher scores to severe positive instances over non-severe negative instances given a dataset D with n instances, arranging them in a ranked order [65, 120]. Thus, the ranking problem can be defined as follows:

Input: Given a query q and a set of n documents $D = \{d_1, d_2, \ldots, d_n\}$, each pair $x_i = (q, d_i)$ forms an input to the ranking model, where x_i represents the tuple of the query and the i-th document.

Ranker: The ranker is a ML model, such as a Decision Tree or Neural Network, trained to optimize a selected rank-based loss function (i.e. can be pairwise, listwise, or pointwise). The model predicts a relevance score s_i for each document d_i based on the input x_i, such that $s_i \in \mathbb{R}$. The choice of the loss function is crucial as it assesses the accuracy of the predicted scores s_i against the actual relevance scores y_i, with the objective of minimizing the difference in ranks. The model can be represented as $s_i = f(x_i)$, where f is the scoring function learned during training.

Output: The output consists of a ranked list of documents sorted by the predicted scores s_i, in descending order. This ranked output is then evaluated against the true binary relevance labels y_i, which indicate the severity of each incident (e.g. severe or non-severe). These labels are used to compute rank based performance metrics that assess the model's effectiveness in prioritizing documents according to their true relevance.

B.4 LTR using Query-Independent Classifiers

In IR, several studies utilized query-independent classification models for relevance ranking, such as [36, 58, 119, 20, 187]. Unlike query-dependent LTR, these models distinguish between classes directly from the input features without considering any query/topic-related information.

In case of binary relevance ranking, the output probability scores of the positive class from the classifiers can be used as the rank scores. The process of using binary classifier model to rank documents can be defined as follows:

Input: Given a set of n documents $D = \{d_1, d_2, \ldots, d_n\}$, each instance $x_i = d_i$ forms an input to the classifier model, where x_i represents the feature set of the i-th document.

Classifier: The classifier is a ML model, such as a Decision Tree or Neural Network, trained to optimize a probabilistic loss function (e.g., binary logistic regression loss [93]). In computes the probability $s_i = P(\hat{y}_i = 1|x_i)$, of each instance x_i belonging to the positive class (indicating relevance or severity), with $s_i \in [0, 1] \subset \mathbb{R}$. The loss function assesses the accuracy of these predictions against the actual relevance label y_i, aiming to minimize mis-classification.

Output: The model outputs a list of documents ranked in descending order based on the probability scores s_i of the positive class. This list is evaluated against the binary relevance labels y_i, denoting incident severity, to compute performance metrics that assess how effectively the model prioritizes documents by their true relevance.

B.5 Ranker Specific Hyper-parameters

Among the query-dependent ranking models, some parameters are specific to ranking-based objectives and significantly impact model performance. The XGBoost library [33, 40], provides three ranking loss functions: “rank:ndcg”, and “rank:map” and “rank:pairwise”. The “rank:ndcg” and “rank:map” use the LambdaMART [18] algorithm, optimizing for Normalized Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP) metrics respectively (for details on metrics see section 3.8.6). However, the “rank:pairwise” objective uses the original RankNet [17] loss without scaling. We tune models on these ranking objectives, selecting the most effective loss function for each feature representation as they are variations within the same LTR algorithm family. In contrast, the LGBM library [89, 37] implements only the LambdaRank [179] algorithm, which was introduced before LambdaMART.

Additionally, XGBoost’s “lambdarank_pair_method” offers two strategies for pair generation: “mean” and “topk”. The “mean” strategy randomly samples m pairs per document within a query group, where m is defined by the “lambdarank_num_pair_per_sample” hyperparameter [40]. Conversely, “topk” focuses on the top k documents, which improves efficiency in large datasets

but may result in fewer effective pairs for training. We tune both strategies alongside "lambdarank_num_pair_per_sample" to ensure optimal performance.

Appendix C

Chap 5: Active Learning Strategies

C.1 AL Selection Strategies

This section defines the specifies the algorithms we used for Active Learning (AL) selection strategy, including three uncertainty-based methods: Least Confidence, Margin, and Entropy; one representativeness-based method, Information Density Cosine selection; and one Random selection strategy that serves as a baseline for comparison with the other approaches. For background on AL selection strategies, refer to Section 2.2.2.

For all subsequent formulas, let $\mathbf{P}$ be the matrix of output probabilities from a model θ for both labeled and unlabeled samples, where each row corresponds to a sample and each column to the predicted probability of a class. Let n be the number of uncertain samples to select, and let $\mathbf{U}$ be the set of indices for the unlabeled samples.

1. **Least Confidence:** Is a simple uncertainty selection strategy that queries the instance x whose predicted output is the least confident. The uncertainty measure for each sample x can be written as:

$$U(x) = 1 - P_\theta(\hat{y}|x) \tag{C.1}$$

where $\hat{y} = \underset{y}{\operatorname{argmax}}\ P_\theta(y|x)$ is the class prediction with the highest posterior probability for sample x under the model θ.

The least confidence selection strategy selects the n samples with the highest uncertainty,

which can be formalized as:

$$X^*_{LC} = \underset{x \in \mathbf{U}}{\text{argsort}}\ U(x)[:n] \tag{C.2}$$

Here, argsort orders the samples x according to the uncertainty measure $U(x)$ in *descending* order, and $[:n]$ indicates selecting the first n samples from this sorted list, such that all indices selected fall within the unlabeled set $\mathbf{U}$.

2. **Margin:** The Margin method is an uncertainty selection strategy that extends the concept of least confidence. Unlike only using the highest probable class, this approach focuses on the difference between the output probabilities of the model's two most likely predictions. Specifically, the margin is defined as the difference between the probabilities of the first and second most likely class predictions. The smaller this difference, the greater the uncertainty associated with a sample. The margin for each sample x is given by:

$$M(x) = P_\theta(\hat{y_1}|x) - P_\theta(\hat{y_2}|x) \tag{C.3}$$

where $\hat{y_1}$ and $\hat{y_2}$ are the first and second highest class prediction probabilities under the model θ, respectively.

Thus, margin sampling will select n samples for which this margin is smallest. This can be formulated as:

$$X^*_M = \underset{x \in \mathbf{U}}{\text{argsort}}\ M(x)[:n] \tag{C.4}$$

Unlike least confidence, here argsort orders the samples x according to the margin $M(x)$ in *ascending* order, selecting the first n samples from this sorted list, such that all indices selected fall within the unlabeled set $\mathbf{U}$.

3. **Entropy:** In information theory, entropy is a measure of the uncertainty of an event's outcome and is one of the most common uncertainty sampling methods [149, 145]. Entropy is high, when an event has multiple equally likely outcomes indicating the difficulty in predicting the true outcome. Conversely, if one outcome is much more likely than the others, the entropy is low because there is less uncertainty. Entropy of an instance x can be computed as follows:

$$H(x) = -\sum_y P_\theta(y|x) \log P_\theta(y|x), \tag{C.5}$$

where $P_\theta(y|x)$ is the probability of label y given the sample x under the model parameters θ, and the summation is performed over all possible labels. The negative sign ensures that the entropy is a non-negative quantity.

Since the most uncertain samples are the ones with highest entropy $H(x)$, the sampling will be done as follows:

$$X_E^* = \underset{x \in \mathbf{U}}{\text{argsort}}\ H(x)[:n] \tag{C.6}$$

To get the top n samples with highest entropy values $H(x)$, argsort sorts the samples x in *descending* order, to select the first n samples from this sorted list, ensuring all indices belong to the unlabeled set $\mathbf{U}$.

4. **Information Density Cosine:** Uncertainty selection strategies rely on probability outputs of the classifier, and therefore does not take into account the structure of the data or input features [145]. Information Density selection addresses this by querying instances that are representative of the input space distribution. Information density sampling weighs the information of an instance x by its average similarity to all other instances in the input data using similarity measures such as cosine similarity or Euclidean distance [145]. It is used in conjunction with a base sampling strategy such as Least Confidence, querying instances that are both informative and representative of the input space distribution. In this study, we adopt Cosine similarity for Information Density since it is more suitable for text data. Cosine similarity measures the angle between input vectors, effectively capturing the content similarity independant of document length, unlike Euclidean distance [76]. For an unlabeled dataset $\mathbf{U}$, with a base selection strategy ϕ, the information density of an instance x can be calculated as:

$$I(x) = \phi(x) \times \left(\frac{1}{|\mathbf{U}|} \sum_{x' \in X} \text{sim}(x, x') \right), \tag{C.7}$$

where sim(x, x') is the cosine similarity function and $\phi(x)$ is the informativeness measure of x computed using a base AL selection strategy.

The higher the information density, the more similar the given instance is to the rest of the data. Thus we sort instances in *descending* order, such that the first n samples are the ones with the highest information density:

$$X_I^* = \underset{x \in \mathbf{U}}{\text{argsort}}\ I(x)[:n] \tag{C.8}$$

5. **Random:** To compare the above selection strategies against a baseline we implement random selection that aims to randomly select instances from the unlabelled set for labelling. Formally, the random selection method can be defined as:

$$X_R^* = \text{RandomSample}(\mathbf{U}, n) \tag{C.9}$$

where $\text{RandomSample}(\mathbf{U}, n)$ is a function that selects n distinct indices uniformly at random from the unlabelled set $\mathbf{U}$, without replacement.

Bibliography

[1] Armen Aghajanyan et al. "Better fine-tuning by reducing representational collapse". In: *arXiv preprint arXiv:2008.03156* (2020).

[2] Akiko Aizawa. "An information-theoretic perspective of tf–idf measures". In: *Information Processing & Management* 39.1 (2003), pp. 45–65.

[3] Sara Albolino et al. "Patient safety and incident reporting: survey of Italian healthcare workers". In: *BMJ Quality & Safety* 19.Suppl 3 (2010), pp. i8–i12.

[4] Fatimah Alzamzami, Mohamad Hoda, and Abdulmotaleb El Saddik. "Light gradient boosting machine for general sentiment classification on short texts: a comparative evaluation". In: *IEEE access* 8 (2020), pp. 101840–101858.

[5] Asma Ammari et al. "Deep-active-learning approach towards accurate right ventricular segmentation using a two-level uncertainty estimation". In: *Computerized Medical Imaging and Graphics* 104 (2023), p. 102168.

[6] Les Atlas, David Cohn, and Richard Ladner. "Training connectionist networks with queries and selective sampling". In: *Advances in neural information processing systems* 2 (1989).

[7] Josh Attenberg and Şeyda Ertekin. "Class imbalance and active learning". In: *Imbalanced Learning: Foundations, Algorithms, and Applications* (2013), pp. 101–149.

[8] Brian Bartell, Garrison W Cottrell, and Richard Belew. "Learning to retrieve information". In: *Proceedings of the Swedish Conference on Connectionism*. Citeseer. 1995, p. 27.

[9] Marco Basaldella et al. "COMETA: A corpus for medical entity linking in the social media". In: *arXiv preprint arXiv:2010.03295* (2020).

[10] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. "A comparative analysis of gradient boosting algorithms". In: *Artificial Intelligence Review* 54 (2021), pp. 1937–1967.

[11] Ameni Bouaziz et al. “Short text classification using semantic random forest”. In: *Data Warehousing and Knowledge Discovery: 16th International Conference, DaWaK 2014, Munich, Germany, September 2-4, 2014. Proceedings 16*. Springer. 2014, pp. 288–299.

[12] Leo Breiman. “Random forests”. In: *Machine learning* 45 (2001), pp. 5–32.

[13] Keno K Bressem et al. “Highly accurate classification of chest radiographic reports using a deep learning natural language model pre-trained on 3.8 million text reports”. In: *Bioinformatics* 36.21 (2020), pp. 5255–5261.

[14] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[15] Jeffrey R Brubacher et al. “Barriers to and incentives for safety event reporting in emergency departments”. In: *Healthc Q* 14.3 (2011), pp. 57–65.

[16] Peter Bühlmann. “Bagging, boosting and ensemble methods”. In: *Handbook of computational statistics: Concepts and methods* (2012), pp. 985–1022.

[17] Chris Burges et al. “Learning to rank using gradient descent”. In: *Proceedings of the 22nd international conference on Machine learning*. 2005, pp. 89–96.

[18] Christopher Burges, Robert Ragno, and Quoc Le. “Learning to rank with nonsmooth cost functions”. In: *Advances in neural information processing systems* 19 (2006).

[19] Christopher JC Burges. “From ranknet to lambdarank to lambdamart: An overview”. In: *Learning* 11.23-581 (2010), p. 81.

[20] Zhe Cao et al. “Learning to rank: from pairwise approach to listwise approach”. In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 129–136.

[21] Siw Carlfjord, Annica Öhrn, and Anna Gunnarsson. “Experiences from ten years of incident reporting in health care: A qualitative study among department managers and coordinators”. In: *BMC Health Services Research* 18 (1 Feb. 2018). ISSN: 14726963. DOI: 10.1186/s12913-018-2876-5.

[22] Ilias Chalkidis et al. “An empirical study on large-scale multi-label text classification including few and zero-shot labels”. In: *arXiv preprint arXiv:2010.01653* (2020).

[23] Kyu Hyun Chang. “Explaining Active Learning Queries”. PhD thesis. 2017.

[24] O Chapelle, Y Chang, and TY Liu. “The Yahoo! learning to rank challenge”. In: *Learning to Rank Challenge* (2010).

[25] Ashish Chaturvedi et al. “Comparative Multinomial Text Classification Analysis of Naïve Bayes and XGBoost with SMOTE on Imbalanced Dataset”. In: *Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2021*. Springer. 2022, pp. 339–349.

[26] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. “Special issue on learning from imbalanced data sets”. In: *ACM SIGKDD explorations newsletter* 6.1 (2004), pp. 1–6.

[27] Nitesh V Chawla et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[28] DeHua Chen, Li Zhang, and Chuang Ma. “A multimodal diagnosis predictive model of alzheimer’s disease with few-shot learning”. In: *2020 International Conference on Public Health and Data Science (ICPHDS)*. IEEE. 2020, pp. 273–277.

[29] Hongbo Chen et al. “A Machine Learning Approach with Human-AI Collaboration for Automated Classification of Patient Safety Event Reports: Algorithm Development and Validation Study”. In: *JMIR Human Factors* 11.1 (2024), e53378.

[30] Hongbo Chen et al. “Improving Patient Safety Event Report Classification with Machine Learning and Contextual Text Representation”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 67. 1. SAGE Publications Sage CA: Los Angeles, CA. 2023, pp. 1063–1069.

[31] Jianlv Chen et al. “Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation”. In: *arXiv preprint arXiv:2402.03216* (2024).

[32] Jiaoyan Chen et al. “Zero-shot and few-shot learning with knowledge graphs: A comprehensive survey”. In: *Proceedings of the IEEE* 111.6 (2023), pp. 653–685.

[33] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[34] Tianqi Chen et al. “Xgboost: extreme gradient boosting”. In: *R package version 0.4-2* 1.4 (2015), pp. 1–4.

[35] D Christopher, Prabhakar Raghavan, Hinrich Schütze, et al. “Scoring term weighting and the vector space model”. In: *Introduction to information retrieval* 100 (2008), pp. 2–4.

[36] William S Cooper, Fredric C Gey, and Daniel P Dabney. "Probabilistic retrieval based on staged logistic regression". In: *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*. 1992, pp. 198–210.

[37] Microsoft Corporation. *lightgbm.LGBMRanker - LightGBM 4.3.0.99 documentation*. Accessed: 2024-04-18. 2024. URL: https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRanker.html.

[38] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[39] Pádraig Cunningham. "Ensemble techniques". In: *Techn own Port Scan Dataset as shown in Figure* 6 (2007).

[40] xgboost developers. *Learning to Rank - xgboost 2.1.0-dev documentation*. Accessed: 2024-04-18. 2022. URL: https://xgboost.readthedocs.io/en/latest/tutorials/learning_to_rank.html#references.

[41] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[42] Thomas G Dietterich. "Ensemble methods in machine learning". In: *International workshop on multiple classifier systems*. Springer. 2000, pp. 1–15.

[43] Christos Dimitrakakis and Christian Savu-Krohn. "Cost-minimising strategies for data labelling: optimal stopping and active learning". In: *International Symposium on Foundations of Information and Knowledge Systems*. Springer. 2008, pp. 96–111.

[44] Molla S Donaldson, Janet M Corrigan, and Linda T Kohn. "To err is human: building a safer health system". In: (2000).

[45] Liat Ein Dor et al. "Active learning for BERT: An empirical study". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 7949–7962.

[46] Shari M Erickson et al. "Patient safety: achieving a new standard for care". In: (2003).

[47] Erin B Evangelista, Fabiana Guajardo, and Taikang Ning. "Classification of Abnormal Heart Sounds with Machine Learning; Classification of Abnormal Heart Sounds with Machine Learning". In: *2020 15th IEEE International Conference on Signal Processing (ICSP)* 1 (2020). DOI: 10.1109/ICSP48669.2020.9320916. URL: https://physionet.org.

[48] Huw Prosser Evans et al. “Automated classification of primary care patient safety incident report content and severity using supervised machine learning (ML) approaches”. In: *Health informatics journal* 26.4 (2020), pp. 3123–3139.

[49] Stefanos Fafalios, Pavlos Charonyktakis, and Ioannis Tsamardinos. “Gradient Boosting Trees”. In: *Gnosis Data Analysis PC* (2020), pp. 1–3.

[50] Alberto Fernández et al. *Learning from imbalanced data sets*. Vol. 10. Springer, 2018.

[51] Allan Fong, A Zachary Hettinger, and Raj M Ratwani. “Exploring methods for identifying related patient safety events using structured and unstructured data”. In: *Journal of biomedical informatics* 58 (2015), pp. 89–95.

[52] Allan Fong et al. “A machine learning approach to reclassifying miscellaneous patient safety event reports”. In: *Journal of Patient Safety* 17.8 (2021), e829–e833.

[53] Allan Fong et al. “Using active learning to identify health information technology related patient safety events”. In: *Applied clinical informatics* 26.01 (2017), pp. 35–46.

[54] Yoav Freund et al. “An efficient boosting algorithm for combining preferences”. In: *Journal of machine learning research* 4.Nov (2003), pp. 933–969.

[55] Jeffrey EF Friedl. *Mastering regular expressions*. ” O’Reilly Media, Inc.”, 2006.

[56] Jerome H Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* (2001), pp. 1189–1232.

[57] Omar Galal, Ahmed H Abdel-Gawad, and Mona Farouk. “Federated Freeze BERT for text classification”. In: *Journal of Big Data* 11.1 (2024), p. 28.

[58] Fredric C Gey. “Inferring probability of relevance using the method of logistic regression”. In: *SIGIR’94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*. Springer. 1994, pp. 222–231.

[59] Madzarov Gjorgji, Gjorgjevikj Dejan, and Chorbev Ivan. “A Multiclass SVM Classifier Utilizing Binary Decision Tree”. In: *Informatica* 33.2 (2009), pp. 233–241.

[60] Elif Ceren Gök and Mehmet Onur Olgun. “SMOTE-NC and gradient boosting imputation based random forest classifier for predicting severity level of covid-19 patients with blood samples”. In: *Neural Computing and Applications* 33.22 (2021), pp. 15693–15707.

[61] Mohamed Goudjil et al. “A novel active learning method using SVM for text classification”. In: *International Journal of Automation and Computing* 15 (2018), pp. 290–298.

[62] Mohamed Goudjil et al. “A Novel Active Learning Method Using SVM for Text Classification”. In: *International Journal of Automation and Computing* 15 (3 June 2018), pp. 290–298. ISSN: 17518520. DOI: 10.1007/S11633-015-0912-Z/METRICS. URL: https://link.springer.com/article/10.1007/s11633-015-0912-z.

[63] Mattyws F Grawe, Claudia A Martins, and Andreia G Bonfante. “Automated patent classification using word embedding”. In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2017, pp. 408–411.

[64] Yu Gu et al. “Domain-specific language model pretraining for biomedical natural language processing”. In: *ACM Transactions on Computing for Healthcare (HEALTH)* 3.1 (2021), pp. 1–23.

[65] H Altay Güvenir and Murat Kurtcephe. “Ranking instances by maximizing the area under ROC curve”. In: *IEEE Transactions on Knowledge and Data Engineering* 25.10 (2012), pp. 2356–2366.

[66] James A Hanley and Barbara J McNeil. “The meaning and use of the area under a receiver operating characteristic (ROC) curve.” In: *Radiology* 143.1 (1982), pp. 29–36.

[67] Tawfiq Hasanin et al. “Examining characteristics of predictive models with imbalanced big data”. In: *Journal of Big Data* 6.1 (2019), pp. 1–21.

[68] T Hasegawa and S Fujita. “Patient Safety Policies: experiences, effects and priorities; lessons from OECD member states”. In: *Tokyo (JPN): Ministry of Health, Labour and Welfare* (2018).

[69] Haibo He and Edwardo A Garcia. “Learning from imbalanced data”. In: *IEEE Transactions on knowledge and data engineering* 21.9 (2009), pp. 1263–1284.

[70] Pengcheng He et al. “Deberta: Decoding-enhanced bert with disentangled attention”. In: *arXiv preprint arXiv:2006.03654* (2020).

[71] Tianxu He et al. “An active learning approach with uncertainty, representativeness, and diversity”. In: *The Scientific World Journal* 2014 (2014).

[72] Simon Hegelich. “Decision trees and random forests: Machine learning techniques to classify rare events”. In: *European Policy Analysis* 2.1 (2016), pp. 98–120.

[73] Kurt R Herzer et al. “Patient safety reporting systems: sustained quality improvement using a multidisciplinary team and “good catch” awards”. In: *The Joint Commission Journal on Quality and Patient Safety* 38.8 (2012), 339–AP1.

[74] Jonathan Hillman and Toby Dylan Hocking. "Optimizing roc curves with a sort-based surrogate loss function for binary classification and changepoint detection". In: *arXiv preprint arXiv:2107.01285* (2021).

[75] Peiyun Hu et al. "Active learning with partial feedback". In: *arXiv preprint arXiv:1802.07427* (2018).

[76] Anna Huang et al. "Similarity measures for text document clustering". In: *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*. Vol. 4. 2008, pp. 9–56.

[77] Jin Huang and Charles X Ling. "Using AUC and accuracy in evaluating learning algorithms". In: *IEEE Transactions on knowledge and Data Engineering* 17.3 (2005), pp. 299–310.

[78] Ronda Hughes. "Patient safety and quality: An evidence-based handbook for nurses". In: (2008).

[79] Shehnaz Islam et al. "Evaluating Active Learning Strategies for Automated Classification of Patient Safety Event Reports in Hospitals". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. SAGE Publications Sage CA: Los Angeles, CA. 2024, p. 10711813241260676.

[80] Nasir Jalal et al. "A novel improved random forest for text classification using feature ranking and optimal number of trees". In: *Journal of King Saud University-Computer and Information Sciences* 34.6 (2022), pp. 2733–2742.

[81] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[82] Kalervo Järvelin and Jaana Kekäläinen. "IR evaluation methods for retrieving highly relevant documents". In: *ACM SIGIR Forum*. Vol. 51. ACM New York, NY, USA. 2017, pp. 243–250.

[83] Shaoxiong Ji, Matti Hölttä, and Pekka Marttinen. "Does the magic of BERT apply to medical code assignment? A quantitative study". In: *Computers in biology and medicine* 139 (2021), p. 104998.

[84] Ronghui Ju et al. "An efficient method for document categorization based on word2vec and latent semantic analysis". In: *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*. IEEE. 2015, pp. 2276–2283.

[85] Daniel Jurafsky and James H Martin. "Chapter 5-Logistic Regression". In: *Speech and Language Processing,* (2023), pp. 11–17.

[86] Md Abdul Kader et al. "Contextual embedding for distributed representations of entities in a text corpus". In: *Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*. PMLR. 2016, pp. 35–50.

[87] Rohit Kumar Kaliyar. "A multi-layer bidirectional transformer encoder for pre-trained word embedding: a survey of bert". In: *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE. 2020, pp. 336–340.

[88] Denys Katerenchuk and Andrew Rosenberg. "RankDCG: Rank-ordering evaluation measure". In: *arXiv preprint arXiv:1803.00719* (2018).

[89] Guolin Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems* 30 (2017).

[90] Faria Khandaker et al. "Transformer models for mining intents and predicting activities from emails in knowledge-intensive processes". In: *Engineering Applications of Artificial Intelligence* 128 (2024), p. 107450.

[91] Ji-Yoon Kim, Hun-Seok Lee, and Jin-Seok Oh. "Study on prediction of ship's power using light GBM and XGBoost". In: *Journal of the Korean Society of Marine Engineering* 44.2 (2020), pp. 174–180.

[92] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[93] David G Kleinbaum, Mitchel Klein, and Erica Rihl Pryor. *Logistic regression: a self-learning text*. Vol. 94. Springer, 2002.

[94] Daisuke Koike et al. "Implementation strategies for the patient safety reporting system using Consolidated Framework for Implementation Research: a retrospective mixed-method analysis". In: *BMC health services research* 22.1 (2022), p. 409.

[95] Ananya Kumar et al. "Fine-tuning can distort pretrained features and underperform out-of-distribution". In: *arXiv preprint arXiv:2202.10054* (2022).

[96] Jinhyuk Lee et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* 36.4 (2020), pp. 1234–1240.

[97] Dongyuan Li et al. "A Survey on Deep Active Learning: Recent Advances and New Frontiers". In: *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[98] Ping Li, Qiang Wu, and Christopher Burges. "Mcrank: Learning to rank using multiple classification and gradient boosting". In: *Advances in neural information processing systems* 20 (2007).

[99] Rumeng Li et al. "Detection of bleeding events in electronic health record notes using convolutional neural network models enhanced with recurrent neural network autoencoders: deep learning approach". In: *JMIR medical informatics* 7.1 (2019), e10788.

[100] Yiming Li et al. "Artificial intelligence-powered pharmacovigilance: A review of machine and deep learning in clinical text-based adverse drug event detection for benchmark datasets". In: *Journal of Biomedical Informatics* (2024), p. 104621.

[101] Yiming Li et al. "Improving Entity Recognition Using Ensembles of Deep Learning and Fine-tuned Large Language Models: A Case Study on Adverse Event Extraction from Multiple Sources". In: *arXiv preprint arXiv:2406.18049* (2024).

[102] Chen Liang and Yang Gong. "Automated classification of multi-labeled patient safety reports: a shift from quantity to quality measure". In: *MEDINFO 2017: Precision Healthcare through Informatics*. IOS Press, 2017, pp. 1070–1074.

[103] Petro Liashchynskyi and Pavlo Liashchynskyi. "Grid search, random search, genetic algorithm: a big comparison for NAS". In: *arXiv preprint arXiv:1912.06059* (2019).

[104] Jimmy Lin and Alek Kolcz. "Large-scale machine learning at twitter". In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. 2012, pp. 793–804.

[105] Tie-Yan Liu et al. "Learning to rank for information retrieval". In: *Foundations and Trends® in Information Retrieval* 3.3 (2009), pp. 225–331.

[106] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[107] David Lowell, Zachary C Lipton, and Byron C Wallace. "Practical obstacles to deploying active learning". In: *arXiv preprint arXiv:1807.04801* (2018).

[108] Yuxing Lu, Xukai Zhao, and Jinzhuo Wang. "Medical knowledge-enhanced prompt learning for diagnosis classification from clinical text". In: *Proceedings of the 5th Clinical Natural Language Processing Workshop*. 2023, pp. 278–288.

[109] Hengyu Luo. *Prompt-learning and Zero-shot Text Classification with Domain-specific Textual Data*. 2023.

[110] Renqian Luo et al. "BioGPT: generative pre-trained transformer for biomedical text generation and mining". In: *Briefings in bioinformatics* 23.6 (2022), bbac409.

[111] Farah Magrabi et al. "Clinical safety of England's national programme for IT: a retrospective analysis of all reported safety events 2005 to 2011". In: *International journal of medical informatics* 84.3 (2015), pp. 198–206.

[112] Martin A Makary and Michael Daniel. "Medical error—the third leading cause of death in the US". In: *Bmj* 353 (2016).

[113] Christopher D Manning. *Introduction to information retrieval*. Syngress Publishing, 2008.

[114] Christopher McMaster et al. "Developing a deep learning natural language processing algorithm for automated reporting of adverse drug reactions". In: *Journal of biomedical informatics* 137 (2023), p. 104265.

[115] Prem Melville and Raymond J Mooney. "Diverse ensembles for active learning". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 74.

[116] George Michalopoulos et al. "ICDBigBird: a contextual embedding model for ICD code classification". In: *arXiv preprint arXiv:2204.10408* (2022).

[117] Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. "On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines". In: *arXiv preprint arXiv:2006.04884* (2020).

[118] Eduardo Mosqueira-Rey et al. "Human-in-the-loop machine learning: a state of the art". In: *Artificial Intelligence Review 2022 56:4* 56 (4 Aug. 2022), pp. 3005–3054. ISSN: 1573-7462. DOI: 10.1007/S10462-022-10246-W. URL: https://link.springer.com/article/10.1007/s10462-022-10246-w.

[119] Ramesh Nallapati. "Discriminative models for information retrieval". In: *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. 2004, pp. 64–71.

[120] Harikrishna Narasimhan and Shivani Agarwal. "On the relationship between binary classification, bipartite ranking, and binary class probability estimation". In: *Advances in neural information processing systems* 26 (2013).

[121] Alexey Natekin and Alois Knoll. "Gradient boosting machines, a tutorial". In: *Frontiers in neurorobotics* 7 (2013), p. 21.

[122] Mei-Sing Ong, Farah Magrabi, and Enrico Coiera. "Automated categorisation of clinical incident reports using statistical text classification". In: *Quality and Safety in Health Care* 19.6 (2010), e55–e55.

[123] Mei-Sing Ong, Farah Magrabi, and Enrico Coiera. "Automated identification of extreme-risk events in clinical incident reports". In: *Journal of the American Medical Informatics Association* 19.e1 (2012), e110–e118.

[124] World Health Organization et al. *Global patient safety action plan 2021-2030: towards eliminating avoidable harm in health care*. World Health Organization, 2021.

[125] Arti Patle and Deepak Singh Chouhan. "SVM kernel functions for classification". In: *2013 International conference on advances in technology and engineering (ICATE)*. IEEE. 2013, pp. 1–9.

[126] Yifan Peng, Shankai Yan, and Zhiyong Lu. "Transfer learning in biomedical natural language processing: an evaluation of BERT and ELMo on ten benchmarking datasets". In: *arXiv preprint arXiv:1906.05474* (2019).

[127] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[128] Matthias Perkonigg, Johannes Hofmanninger, and Georg Langs. "Continual active learning for efficient adaptation of machine learning models to changing image acquisition". In: *International Conference on Information Processing in Medical Imaging*. Springer. 2021, pp. 649–660.

[129] Matthew E Peters, Sebastian Ruder, and Noah A Smith. "To tune or not to tune? adapting pretrained representations to diverse tasks". In: *arXiv preprint arXiv:1903.05987* (2019).

[130] Matthew E Peters et al. "Deep contextualized word representations. CoRR abs/1802.05365 (2018)". In: *arXiv preprint arXiv:1802.05365* 42 (1802).

[131] Derek A Pisner and David M Schnyer. "Support vector machine". In: *Machine learning*. Elsevier, 2020, pp. 101–121.

[132] Kedar Potdar, Taher S Pardawala, and Chinmay D Pai. "A comparative study of categorical variable encoding techniques for neural network classifiers". In: *International journal of computer applications* 175.4 (2017), pp. 7–9.

[133] Wisam A Qader, Musa M Ameen, and Bilal I Ahmed. "An overview of bag of words; importance, implementation, applications, and challenges". In: *2019 international engineering conference (IEC)*. IEEE. 2019, pp. 200–204.

[134] Zhang Qi. "The text classification of theft crime based on TF-IDF and XGBoost model". In: *2020 IEEE International conference on artificial intelligence and computer applications (ICAICA)*. IEEE. 2020, pp. 1241–1246.

[135] Tao Qin et al. "Query-level loss functions for information retrieval". In: *Information Processing & Management* 44.2 (2008), pp. 838–855.

[136] Natasha Rafter et al. "Adverse events in healthcare: learning from mistakes". In: *QJM: An International Journal of Medicine* 108.4 (2015), pp. 273–277.

[137] Laila Rasmy et al. "Med-BERT: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction". In: *NPJ digital medicine* 4.1 (2021), p. 86.

[138] Emma Richard and Bhargava Reddy. "Text Classification for Clinical Trial Operations: Evaluation and Comparison of Natural Language Processing Techniques". In: *Therapeutic Innovation and Regulatory Science* 55 (2 Mar. 2021), pp. 447–453. ISSN: 21684804. DOI: 10.1007/S43441-020-00236-X/FIGURES/5. URL: https://link.springer.com/article/10.1007/s43441-020-00236-x.

[139] Anthony Rios and Ramakanth Kavuluru. "Few-shot and zero-shot multi-label learning for structured label spaces". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*. Vol. 2018. NIH Public Access. 2018, p. 3132.

[140] ECM Santiago Gonzalez-Carvajal. "Comparing BERT against traditional machine". In: *Retrieved* 5.17 (2020), p. 2020.

[141] Robert E Schapire. "The strength of weak learnability". In: *Machine learning* 5 (1990), pp. 197–227.

[142] Tim Schopf, Daniel Braun, and Florian Matthes. "Evaluating unsupervised text classification: zero-shot and similarity-based approaches". In: *Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval*. 2022, pp. 6–15.

[143] S Selva Birunda and R Kanniga Devi. "A review on word embedding techniques for text classification". In: *Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2020* (2021), pp. 267–281.

[144] Burr Settles. "Active learning literature survey". In: (2009).

[145] Burr Settles. "Synthesis lectures on artificial intelligence and machine learning: active learning". In: *Morgan & Claypool Publishers, San Rafael* (2011). URL: https://link.springer.com/book/10.1007/978-3-031-01560-1.

[146] Burr Settles and Mark Craven. "An analysis of active learning strategies for sequence labeling tasks". In: *proceedings of the 2008 conference on empirical methods in natural language processing*. 2008, pp. 1070–1079.

[147] Kanish Shah et al. "A Comparative Analysis of Logistic Regression, Random Forest and KNN Models for the Text Classification". In: *Augmented Human Research 2020 5:1* 5 (1 Mar. 2020), pp. 1–16. ISSN: 2365-4325. DOI: 10.1007/S41133-020-00032-0. URL: https://link.springer.com/article/10.1007/s41133-020-00032-0.

[148] Kanish Shah et al. "A comparative analysis of logistic regression, random forest and KNN models for the text classification". In: *Augmented Human Research* 5 (2020), pp. 1–16.

[149] Claude Elwood Shannon. "A mathematical theory of communication". In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[150] Shashank Shekhar, Adesh Bansode, and Asif Salim. "A comparative study of hyper-parameter optimization tools". In: *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*. IEEE. 2021, pp. 1–6.

[151] Alex Sherstinsky. "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network". In: *Physica D: Nonlinear Phenomena* 404 (2020), p. 132306.

[152] Pavel Sirotkin. "On search engine evaluation metrics". In: *arXiv preprint arXiv:1302.2318* (2013).

[153] Sonish Sivarajkumar and Yanshan Wang. "HealthPrompt: a zero-shot learning paradigm for clinical natural language processing". In: *AMIA Annual Symposium Proceedings*. Vol. 2022. American Medical Informatics Association. 2022, p. 972.

[154] Congzheng Song et al. "Generalized zero-shot text classification for ICD coding". In: *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 2021, pp. 4018–4024.

[155] Yan Yan Song and Ying Lu. "Decision tree methods: applications for classification and prediction". In: *Shanghai Archives of Psychiatry* 27 (2 Apr. 2015), p. 130. ISSN: 10020829. DOI: 10.11919/J.ISSN.1002-0829.215044. URL: /pmc/articles/PMC4466856//pmc/articles/PMC4466856/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC4466856/.

[156] Alvin Subakti, Hendri Murfi, and Nora Hariadi. "The performance of BERT as data representation of text clustering". In: *Journal of big Data* 9.1 (2022), p. 15.

[157] S Suryavardan et al. "Findings of factify 2: multimodal fake news detection". In: *arXiv preprint arXiv:2307.10475* (2023).

[158] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems* 27 (2014).

[159] Krysta M Svore, Maksims N Volkovs, and Christopher JC Burges. "Learning to rank with multiple objective functions". In: *Proceedings of the 20th international conference on World wide web*. 2011, pp. 367–376.

[160] Thanakorn Thaminkaew, Piyawat Lertvittayakumjorn, and Peerapon Vateekul. "Prompt-Based Label-Aware Framework for Few-Shot Multi-Label Text Classification". In: *IEEE Access* (2024).

[161] C Throop and C Stockmeier. "SEC & SSER patient safety measurement system for healthcare". In: *Healthcare Performance Improvement (HPI) White Paper Series* (2011).

[162] Fouad Trad and Ali Chehab. "Prompt engineering or fine-tuning? a case study on phishing detection with large language models". In: *Machine Learning and Knowledge Extraction* 6.1 (2024), pp. 367–384.

[163] Lewis Tunstall et al. "Efficient few-shot learning without prompts". In: *arXiv preprint arXiv:2209.11055* (2022).

[164] Hamit Taner Ünal and Fatih Başçiftçi. "Evolutionary design of neural network architectures: a review of three decades of research". In: *Artificial Intelligence Review* 55.3 (2022), pp. 1723–1802.

[165] Giorgio Valentini and Francesco Masulli. "Ensembles of learning machines". In: *Neural Nets: 13th Italian Workshop on Neural Nets, WIRN VIETRI 2002 Vietri sul Mare, Italy, May 30–June 1, 2002 Revised Papers 13*. Springer. 2002, pp. 3–20.

[166] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[167] V Viswanathan, S Sridevi, S Balachandran, et al. “MIMIC III Text classification with the generalization of BERT transformer model synergized with XGBoost classifier”. In: *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE. 2023, pp. 1–6.

[168] Ethan Wang, Hong Kang, and Yang Gong. “Generating a health information technology event database from FDA MAUDE reports”. In: *MEDINFO 2019: Health and Wellbeing e-Networks for All*. IOS Press, 2019, pp. 883–887.

[169] Jiaqi Wang et al. “Prompt engineering for healthcare: Methodologies and applications”. In: *arXiv preprint arXiv:2304.14670* (2023).

[170] Yanbo J Wang et al. “A Novel DeBERTa-based Model for Financial Question Answering Task”. In: *arXiv preprint arXiv:2207.05875* (2022).

[171] Ying Wang, Enrico Coiera, and Farah Magrabi. “Using convolutional neural networks to identify patient safety incident reports by type and severity”. In: *Journal of the American Medical Informatics Association* 26.12 (2019), pp. 1600–1608.

[172] Ying Wang et al. “Using multiclass classification to automate the identification of patient safety incident reports by type and severity”. In: *BMC medical informatics and decision making* 17 (2017), pp. 1–12.

[173] Yining Wang et al. “A theoretical analysis of NDCG type ranking measures”. In: *Conference on learning theory*. PMLR. 2013, pp. 25–54.

[174] Yu Wang et al. “Medical text classification based on the discriminative pre-training model and prompt-tuning”. In: *Digital Health* 9 (2023), p. 20552076231193213.

[175] Yuxuan Wang et al. “From static to dynamic word representations: a survey”. In: *International Journal of Machine Learning and Cybernetics* 11 (2020), pp. 1611–1630.

[176] Jason Wei et al. “Chain-of-thought prompting elicits reasoning in large language models”. In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[177] Sean J Welleck. “Efficient AUC optimization for information ranking applications”. In: *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38*. Springer. 2016, pp. 159–170.

[178] ZR Wolf and RG Hughes. "From Error reporting and disclosure: Patient Safety and Quality: An Evidence Based Handbook for Nurses". In: *Rockville. MD: AHRQ publication* (2008), pp. 1–47.

[179] Qiang Wu et al. "Adapting boosting for information retrieval measures". In: *Information Retrieval* 13 (2010), pp. 254–270.

[180] Fen Xia et al. "Listwise approach to learning to rank: theory and algorithm". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1192–1199.

[181] Ran Xu et al. "Knowledge-infused prompting: Assessing and advancing clinical text data generation with large language models". In: (2023).

[182] Leiming Yan, Yuhui Zheng, and Jie Cao. "Few-shot learning for short text classification". In: *Multimedia Tools and Applications* 77 (2018), pp. 29799–29810.

[183] Xi Yang et al. "A large language model for electronic health records". In: *NPJ Digital Medicine* 5.1 (2022), p. 194.

[184] Wang Yining et al. "A theoretical analysis of ndcg ranking measures". In: *Proceedings of the 26th annual conference on learning theory*. 2013.

[185] Ian James Bruce Young, Saturnino Luz, and Nazir Lone. "A systematic review of natural language processing for classification tasks in the field of incident reporting and adverse event analysis". In: *International journal of medical informatics* 132 (2019), p. 103971.

[186] Aston Zhang et al. "Dive into deep learning". In: *arXiv preprint arXiv:2106.11342* (2021). URL: `https://d2l.ai/chapter_natural-language-processing-pretraining/word2vec.html`.

[187] Harry Zhang and Jiang Su. "Naive bayesian classifiers for ranking". In: *European conference on machine learning*. Springer. 2004, pp. 501–512.

[188] Tianyu Zhao et al. "Aspect-Based Sentiment Analysis using Local Context Focus Mechanism with DeBERTa". In: *2023 5th International Conference on Data-driven Optimization of Complex Systems (DOCS)*. IEEE. 2023, pp. 1–6.

[189] Yukun Zhu et al. "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 19–27.