Enhancing Safety and Human Reliability through Data-Driven and NLP Innovations

Karl Johnson

SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING

UNIVERSITY OF STRATHCLYDE

Glasgow, United Kingdom 2024

Declaration of Authenticity and Author's Rights

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Karl Johnson

Date: 10/07/2024

Statement of Co-Authorship

The following people and institutions contributed to publications which are part of this thesis:

Candidate:

Karl Johnson, University of Strathclyde, Glasgow

Co-Authors:

Edoardo Patelli, University of Strathclyde, Glasgow (Primary Supervisor)

Caroline Morais, Agency for Petroleum, Natural Gas and Biofuels (ANP), Rio de Janeiro, Brazil

Lesley Walls, University of Strathclyde, Glasgow

Ka Lai Yung, University of Toronto, Toronto, Canada

Raphael Moura, Agency for Petroleum, Natural Gas and Biofuels (ANP), Rio de Janeiro, Brazil

Michael Beer, Leibniz University, Hannover, Germany

Paper 1 [P1]: Caroline Morais, Ka Lai Yung, Karl Johnson, Raphael Moura, Michael Beer, Edoardo Patelli. Identification of Human Errors and Influencing Factors: A Machine Learning Approach. Safety Science 146, (2022). `https://doi.org/10.1016/j.ssci.2021.105528`.

Caroline Morais: Problem formulation, research design, conceptualization, visualization, writing (original draft preparation), review & editing.

Ka Lai Yung: Conceptualization, initial software development, review & editing

Karl Johnson: Software development & refinement, visualization, testing & validation, some writing, review & editing.
Raphael Moura: Conceptualization, review & editing
Michael Beer: Conceptualization, review & editing
Edoardo Patelli: Conceptualization, review & editing
All authors discussed the outcomes and contributed to the final manuscript.
Chapter 3 is based on this article.

Paper 2 [P2]: Karl Johnson, Caroline Morais, Edoardo Patelli. Enhancing Procedure Quality: Advanced Language Tools for Identifying Ambiguity and High-Potential Violation Triggers. Reliability Engineering & System Safety 264, Part A, (2025). https://doi.org/10.1016/j.ress.2025.111308
Karl Johnson: Problem formulation, research design, conceptualization, software development & refinement, visualization, writing (original draft preparation), review & editing.
Caroline Morais: Problem formulation, conceptualization, review & editing
Edoardo Patelli: Conceptualization, review & editing
All authors discussed the outcomes and contributed to the final manuscript.
Chapter 5 is based on this article.

The work presented here has also been included in multiple conference proceedings, including;
Conference Paper 1 [C1]: Karl Johnson, Caroline Morais, Lesley Walls, Edoardo Patelli. A Data Driven Approach to Elicit Causal Links between Performance Shaping Factors and Human Failure Events. 22nd European Safety and Reliability Conference 2022 (ESREL 2022), Dublin, Ireland. http://dx.doi.org/10.3850/978-981-18-5183-4_R12-12-372.

Conference Paper 2 [C2]: Karl Johnson, Caroline Morais, Edoardo Patelli. AI Tools for Human Reliability Analysis. 5th ECCOMAS Thematic Conference on Uncertainty Quantification in Computational Sciences and Engineering (UNCECOMP 2023), Athens, Greece. http://dx.doi.org/10.7712/120223.10348.20039.

Conference Paper 3 [C3]: Karl Johnson, Caroline Morais, Edoardo Patelli. Natural Language Processing Tool for Identifying Influencing Factors in Human Reliability Analysis and Summarizing Accident Reports. 23rd European Safety and Reliability Conference 2023 (ESREL 2023), Southampton, UK. http://dx.doi.org/10.3850/978-981-18-8071-1_P294-cd.

Conference Paper 4 [C4]: Karl Johnson, Caroline Morais, Edoardo Patelli. Identifying Ambiguity and Potential Violations in Standard Operating Procedures using Natural Language Processing Tools. 24th European Safety and Reliability Conference 2024 (ESREL 2024), Cracow, Poland.

Table of Contributions

Tool	Papers	Methodology	Chapter	Application
Virtual Human Factors Classifier (HF Classifier)	[P1] [C2]	Bag-of-Words, Support Vector Machine	3	Data collection for human reliability analysis.
Virtual Human Factors Classifier - 2nd Generation (HF Classifier 2.0)	[C2] [C3]	BERT	4	Data collection for human reliability analysis.
Human-Centric Summarizer	[C3]	BART	4	Support explainability of classifier. Provide human-focused summaries of the accidents.
Ambiguity Identifier	[P2] [C4]	Part of Speech Tagging (PoS), Dependency Parsing, Regular Expression (Regex), Word Embeddings	5	Improve the quality and clarity of procedure guides by identifying ambiguous and unclear language and phrasing.
High-Potential Violation Trigger Identification Tool (Violation Trigger tool)	[P2] [C4]	BERT, Transfer Learning	5	Identify procedure steps that when violated may contribute to incidents, based on past incident data.
Human Factors Causal Relationships Tool (HF Relationships tool)	[C1] [C2]	Structure Learning Algorithms for Bayesian Networks	6	Identify dependency relationships between performance shaping factors and human errors based on data, reducing reliance on expert opinion.

Table 1: Summary of Tools, Methodologies, and Applications

Acknowledgements

I would like to express my deepest gratitude to all those who supported me throughout the completion of this thesis.

First, I would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for sponsoring my work and research (grant EP/T517938/1).

I am profoundly grateful to my supervisors. My deepest thanks goes to Edoardo Patelli for the opportunity, guidance, support, and encouragement. To Caroline Morais, whose expertise in safety, risk, and HRA, as well as her advice on surviving a PhD, have been invaluable, I extend my heartfelt gratitude.

I would also like to thank all past and current members of the research group for setting a high standard, providing support, ideas, and helping to build my confidence. Thanks to all the professors and researchers I've encountered in the department, at the university, conferences, and other events; you have helped motivate me and fueled my desire to achieve.

Now for some more personal acknowledgments. Special thanks to the Strathclyde Men's Basketball Club and all the guys, who provided me with much-needed breaks from my work. Thank you to all my friends for keeping in contact with me when I was buried in papers and code, and for providing something to look forward to.

To my partner, Parna, the most important result of my time in Glasgow, thank you for going through this journey with me, for all the support, the laughter, and for listening to me talk about my work endlessly as I tried to figure out a problem.

Of course, special thanks to my family, who have supported me every second. To my dad,

Karl, and my Gran, Terry, thank you for the daily messages of encouragement.

My sisters, Molly and Rosie, thank you for keeping my life full of fun and always giving me a reason to smile.

And finally, thank you to my mum, Sharon. Without your support, none of this would be possible. Thank you for working so hard to support me and my sisters, for reading my writing and listening to my ideas, and for always pushing me to work hard and do my best.

Abstract

This thesis explores the development and application of innovative algorithmic, data-driven, machine learning, and natural language processing tools designed to enhance human reliability analysis in critical sectors such as nuclear power, aviation, and oil and gas. Motivated by identified challenges and opportunities in these industries, a suite of advanced tools was created to address key aspects of safety analysis and management.

Presented in this work are six tools, the first- and second- generation *Virtual Human Factors Classifiers*, the *Human-Centric Summarizer*, the *High-Potential Violation Trigger Identification tool*, the *Ambiguity Identifier* and finally the *Human Factors Causal Relationships tool*.

The *Virtual Human Factors Classifiers* were designed to automatically read analyze accident reports to classify the contributing factors. The primary motivation for this development was the expansion of a human reliability analysis database (MATA-D, Multi-attribute Technological Accidents Dataset), to provide the additional data necessary to address the issue of missing information and reduce the uncertainty of human error probability models. The tools have also demonstrated additional applications such as aiding assessors in their reviews of accidents and informing the procedure design process.

Complimentarily the *Human-Centric Summarizer* was developed to distill lengthy accident reports into high-quality concise summaries, that emphasize the human role in the incident. The summarizer serves a dual purpose. Firstly, it aids researchers and safety professionals in rapidly grasping each report, as well as any models based on the incident, without delving into the pages of detailed reports. Secondly, it assists in maintaining and updating the MATA-D. The summaries generated provide a quick reference to the key points of each incident, facilitating easier analysis and review of performance shaping factor classification.

In high-risk industrial environments, the clarity and accuracy of standard operating procedures are critical for ensuring safety and regulatory compliance. The presence of ambiguities in standard operating procedures can lead to misunderstandings, errors, and increased risks. While violations of procedural directives can significantly contribute to catastrophic outcomes.

To address these issues, two additional tools are introduced that leverage both rule-based and machine learning methodologies in natural language processing to evaluate the quality of standard operating procedure documents. The *High-Potential Violation Trigger Identification tool* identifies directives within procedural guides that when violated pose a high-risk potential. And the *Ambiguity Identifier* has been designed to detect various types of ambiguities and misleading steps within procedure guides.

By addressing these linguistic and procedural discrepancies, the tools aim to enhance the clarity and applicability of standard operating procedures, ultimately improving adherence and reducing risks in complex operational settings.

The final tool presented in this work is the *Human Factors Causal Relationships Tool*. It leverages data collected through the MATA-D to identify causal relationships among performance shaping factors. This tool is designed to reduce reliance on expert judgment in the development of human error models, thereby helping to mitigate concerns related to subjectivity and bias.

Case studies are presented for each tool, demonstrating their real-world utility and effectiveness in critical industry contexts. This thesis highlights the potential of data-driven and natural language processing approaches to revolutionize human reliability analysis practices, ultimately enhancing safety across critical industries.

Contents

List of Figures

List of Tables

Abbreviations

AI - Artificial Intelligence

ANP - Agência Nacional do Petróleo, Gás Natural e Biocombustíveis (Brazilian Oil & Gas regulator)

ASEP - Accident Sequence Evaluation Program

ATHEANA - A Technique for Human Event Analysis

BART - Bidirectional and Auto-Regressive Transformers

BERT - Bidirectional Encoder Representations from Transformers

BLEU - Bilingual Evaluation Understudy

BN - Bayesian Network

BoW - Bag of Words

CDSM - Cidade de São Mateus

CPT - Conditional Probability Table

CRAE - Conditional Relative Average Entropy

CREAM - Cognitive Reliability and Error Analysis Method

CSB - US Chemical Safety and Hazard Investigation Board

DAG - DIrected Acyclic Graph

FNN - Feed-Forward Neural Networks

FPSO - Floating Production, Storage and Offloading

HCR - Human Cognitive Reliability

HEART - Human Error Assessment and Reduction Technique

HEP - Human Error Probability

HF - Human Factor

HRA - Human Reliability Analysis

IAEA - International Atomic Energy Agency

IOGP - International Oil and Gas Producers

ISO - International Organization for Standardization

LCS - Longest Common Subsequence

LDA - Latent Dirichlet Allocation

LLM - Large Language Model

LOPA - Layers of Protection Analysis

LSMA - Semantic Analysis

MATA-D - Multi-Attribute Technological Accidents Dataset

MCAS - Maneuvering Characteristics Augmentation System

METEOR - Metric for Evaluation of Translation with Explicit ORdering

MI - Mutail Information

ML - Machine Learning

MLM - Masked Language Modeling

MSG - Maintenance Steering Group

NLP - Natural Language Processing

NLTK - Natural Language Toolkit

NPC - Necessary Path Condition

NRC - US Nuclear Regulatory Commission

NTSB - US National Transportation Safety Board

OCR - Optical Character Recognition

OOD - Out-Of-Distribution

OOV - Out-Of-Vocabulary

OREDA - Offshore and Onshore Reliability Data

PIGs - Pipeline Inspection Gauges

PoS - Part-of-Speech

PSF - Performance Shaping Factor

ReLU - Rectified Linear Unit

ROUGE - Recall-Oriented Understudy for Gisting Evaluation

SOP - Standard Operating Procedure

SPAR-H - Standardized Plant Analysis Risk-Human Reliability Analysis Method

STM - Structural Topic Model

SVM - Support Vector Machines

TF-IDF - Term Frequency - Inverse Document Frequency

THERP - Technique for Human Error Rate Prediction

TP, TN, FP, FN - True Positive, True Negative, False Positive, False Negative

Chapter 1

Introduction

In today's interconnected and highly complex industrial landscape, safety and risk management are paramount. Industries such as nuclear power, aviation, oil and gas, and chemical manufacturing operate under conditions where human error can lead to catastrophic consequences. In these high-stakes environments, Human Reliability Analysis (HRA) provides a systematic approach to identify, evaluate, and mitigate human errors, thereby enhancing overall safety and system reliability.

HRA methodologies help in understanding how human actions impact system safety, facilitating the development of strategies to reduce error rates. Human errors can manifest in various forms, including lapses, slips, and mistakes, all of which compromise safety (Moura 2023). Recognizing this, authoritative bodies such as the International Atomic Energy Agency (IAEA), the US Nuclear Regulatory Commission (NRC), and the International Organization for Standardization (ISO) advocate for the use of HRA as an essential tool for improving safety in critical operations (IAEA 1996; NRC 2006; ISO 2018).

The significance of HRA cannot be overstated. It provides a structured approach to understanding human behavior in operational contexts, enabling the identification of potential errors before they result in incidents. By incorporating HRA into safety management systems, industries can enhance their resilience and ensure safer operational practices. Moreover, HRA facilitates the design of better training programs and the development of more effective human-machine interfaces (Gunda and Singh 2023).

Human Error Probability (HEP) is a key metric in HRA, representing the likelihood of error during task execution. As a quantifiable estimate, HEP plays a critical role in system safety assessments by indicating the probability of human error in specific operational contexts (Sun et al. 2012). By determining HEP, industries can identify high-risk tasks and implement measures to mitigate these risks, thereby reducing the potential for accidents.

Determining HEP involves several steps:

1. Task Analysis: Breaking down tasks into smaller, manageable components to understand the specific actions required and where errors might occur.
2. Data Collection: Gathering data from various sources such as historical accident reports, operational data, and expert judgments to inform the probability calculations.
3. Modeling Techniques: Using probabilistic models, such as the Technique for Human Error Rate Prediction (THERP) (Swain and Guttmann 1983) or the Human Cognitive Reliability (HCR) (Hannaman et al. 1985) model, to estimate error probabilities based on collected data and defined tasks.
4. Validation and Calibration: Comparing the model predictions with actual observed data to ensure accuracy and to adjust the models as necessary to reflect real-world conditions accurately (Sun et al. 2012).

HEP provides a quantitative basis for assessing human reliability and is integral to designing interventions that enhance safety by reducing the likelihood of human error.

Various Performance Shaping Factors (PSFs) influence HEP, including environmental, organizational, technological, and personal factors (Hollnagel 1998). PSFs are the conditions and influences that affect human performance and can either increase or decrease the likelihood of errors. Evaluating PSFs requires both qualitative and quantitative methods. Qualitative approaches, such as structured expert judgment elicitation and scenario

analysis, help interpret complex interactions and contextual factors that are difficult to quantify (Kirwan 1994). These methods are complemented by quantitative techniques involving statistical models and empirical data.

Key PSFs include:

- Environmental Factors: Conditions such as lighting, noise, temperature, and workspace layout that can affect an individual's ability to perform tasks accurately.
- Organizational Factors: Elements like management practices, communication systems, and safety culture that impact how tasks are performed and supervised.
- Technological Factors: The design and functionality of equipment, tools, and interfaces that workers use, which can facilitate or hinder task performance.
- Personal Factors: Individual characteristics such as experience, training, stress levels, and cognitive abilities that influence how tasks are executed (Hollnagel 1998).

Evaluating PSFs is vital for accurate HRA and effective risk management. By understanding how these factors interact and influence human performance, industries can design better systems, processes, and training programs to reduce the probability of errors (Moura et al. 2016). This comprehensive approach ensures a more reliable and safer operational environment.

However, implementing HRA comes with several challenges and limitations. There is a growing demand for more quantitative, evidence-based methods to support analysis and decision-making across various fields. This demand relies on accurate and comprehensive data, which is often unavailable. As collecting the required data is labor-intensive and time-consuming, requiring meticulous analysis to ensure all relevant factors are identified (Moura et al. 2016). Consequently, HRA heavily depends on expert judgments, which can introduce subjective biases and inconsistencies in the analysis.

Given the complexity of the systems and the numerous factors influencing human performance, it can be difficult to explain the rationale behind conclusions and decisions reached. This lack of explainability can hinder the acceptance and implementation of HRA recommendations by stakeholders (French et al. 2011).

The field increasingly desires to incorporate more data-driven and computational ap-

proaches, leveraging technological advances to enhance HRA. By utilizing data, advanced analytics, and machine learning (ML) in various and innovative ways, industries can benefit all processes and improve overall safety.

Inspired by this demand and technological advancements, this thesis discusses the development of several computational tools, integrating ML and Natural Language Processing (NLP) techniques, to tackle various challenges in HRA.

1.0.1 Motivation, Goals and Contributions

The motivation for this project stems from the need to modernize HRA by integrating data-driven approaches. Traditional HRA techniques often depend on expert judgment and qualitative assessments, which can limit consistency and scalability. This thesis presents a suite of tools designed to support key aspects of HRA processes through natural language processing (NLP) and machine learning (ML). Together, the contributions aim to enhance the precision, efficiency, and practical utility of HRA in safety critical contexts.

To increase the utilization of data and ML in HRA, the first challenge to address is data availability. Collecting HRA relevant information, particularly from real-world incidents, is a labor-intensive and time-consuming process. Developing more efficient data collection methods is essential to ensure the availability of comprehensive and accurate data for analysis. Improving this process not only accelerates the identification of key factors and trends in human error but also supports more informed modeling and evidence based safety interventions. While faster access to insights can improve responsiveness, the ultimate goal is to enable deeper, data-supported understanding that underpins robust and deliberate safety decisions.

The first tool developed to improve the efficiency of HRA data collection, while effective, lacks the precision and contextual understanding required for unsupervised data collec-

tion with confidence. Therefore, there is a need to develop a second-generation tool that utilizes advanced ML models capable of collecting data that is more reflective of real-world scenarios. This improvement would lead to increased confidence in the data, and to better-informed decision-making and more effective risk mitigation strategies.

The first tool presented is a classifier, developed to improve the efficiency of HRA data collection. While effective in reducing manual effort, it lacks the precision and contextual awareness needed for fully unsupervised operation. To address this a second-generation classifier has been developed. This tool leverages advanced ML techniques to improve the accuracy and contextual relevance of collected data, making it more representative of real-world scenarios. These improvements increase confidence in the data and support better informed decision-making and more effective risk mitigation strategies.

Effective safety management also depends on the clear and context-rich communication of insights. Accident reports typically contain extensive and complex narratives, which can obscure critical information related to human involvement. The third tool presented is a summarization tool developed to distill these reports into concise, accessible summaries that highlight human elements. These summaries are designed to support comprehensive analysis by making key insights easier to interpret and communicate across stakeholders. With enhanced data availability and improved interpretation, it becomes feasible to uncover and analyze the complex dependencies and interactions between various PSFs. A fourth tool (presented last in this thesis) addresses this need by applying machine learning techniques to identify and model the relationships between PSFs based on collected human reliability data. By learning from real-world patterns, the tool generates structured representations of how different factors influence each other and contribute to human error. These data-driven models allow for the visualization and support the quantitative estimation of HEP.

Building on this analytical foundation, the final tools presented in this thesis focus on enhancing the clarity, consistency, and safety impact of standard operating procedures (SOPs). SOPs play a critical role in guiding human actions in complex and high-risk environments, but they can often suffer from ambiguity, inconsistent phrasing, or poor

alignment with operational context.

The fifth tool analyzes SOP text to identify vague language, syntactic complexity, and structural inconsistencies that may contribute to misinterpretation or non-compliance. The sixth tool builds on this by identifying high-risk directives that are critical to safety. By flagging these elements, the tool enables focused review and targeted intervention, helping organizations ensure that critical instructions are not only clear but also reliably followed.

Together, these tools support a more systematic and data-informed approach to procedural design, contributing to reduced human error, improved compliance, and stronger overall safety outcomes.

While this work emphasizes improving the efficiency of data handling, interpretation, and communication, its deeper purpose is to enhance the quality of safety-related decision-making. The tools are designed not to replace expert judgment, but to support it, providing timely, structured, and relevant insights that help ensure safety interventions are grounded in careful, comprehensive understanding.

Based upon these motivations this thesis sets out the following main goals;

- (G1) Overcome data collection constraints by developing a more efficient approach for learning data from accident reports.
- (G2) Increase confidence in the automated extraction and collection of data.
- (G3) Support more efficient decision making, while maintaining understanding and explainability when reviewing the human role in incidents.
- (G4) Reduce reliance on expert opinion in HRA model construction.
- (G5) Improve the clarity and accuracy of SOPs, particularly focusing on ambiguity and directives with high-risk potential if violated.

1.0.1.1 Contributions

This thesis' main contribution is the development of a suite of tools that aim to tackle each of the goals, thereby enhancing HRA and improving safety outcomes across various high-risk industries.

To address (G1) the a automated classification tool (presented in Chapter 3 has been development, designed to automate the identification of influencing factors and human errors in accident reports, enabling more efficient collection of HRA data.

(G2) is to address the limitations encountered in the initial response to (G1). To increase confidence in automated data collection, and utilise developments in NLP, a second-generation classifier was designed (presented in Chapter 4). This tool leverages a large language model (LLM), specifically BERT.

To meet (G3), together with the classifier tools, a tool for summarizing the human role within accident reports has been created, introduced in Chapter 4. This tool provides evidence and explainability for the classification, in a concise format that facilitates a more structured and accessible evaluation process. This supports faster understanding, with the goal of enabling informed and carefully considered decisions.

To reduce the reliance on expert opinion in model construction (G4), a data-driven tool to support the identification of the causal relationships between PSFs has been developed. The tool leverages established approaches in Bayesian network (BN) construction, with the option of integrating expert opinion (Chapter 6, Section 6.4).

Finally, to address (G5), two distinct tools were developed that combine linguistic rules, NLP, and ML techniques. The first of these tools is designed to identify ambiguous wording, structure, and terminology in SOPs. The second leverages insights from past incidents to flag directives with high-risk potential if violated. Both SOP focused tools are presented in Chapter 5.

1.0.2 Thesis Structure

In the following chapters, the development, implementation, and case study applications of the tools, designed to address the identified motivations and achieve the intended impacts, are presented.

The second chapter provides the background on HRA, data, and ML methods. It details the relevant and related studies for each of the identified challenges. This chapter further supports the motivations behind each of the developed tools.

Chapter 3 introduces the *Virtual Human Factors Classifier*, designed to automate the identification of influencing factors and human errors in accident reports, enabling more efficient collection of HRA data. This chapter is based on a journal paper co-authored by the candidate, paper [P1], titled "Identification of Human Errors and Influencing Factors: A Machine Learning Approach," published in Safety Science Journal - Volume 146 in February 2022 (Morais et al. 2022a).

Chapter 4 begins with a discussion of the limitations of the classifier from Chapter 3 and other challenges associated with the use of data and ML, particularly regarding explainability. This is followed by the introduction of the second-generation *Virtual Human Factors Classifier*, which leverages advancements in NLP and LLM. Additionally, a tool for summarizing the human role within accident reports is introduced (*Humane-Centric Summarizer*), providing evidence and explainability for the classification and enabling quicker evaluation and decision-making. This chapter is based upon work presented at the European Safety and Reliability Conference (ESREL 2023) in the paper, [C3] titled "Natural Language Processing Tool for Identifying Influencing Factors in Human Reliability Analysis and Summarizing Accident Reports" (Johnson et al. 2023b).

Chapter 5 introduces the development of two distinct tools designed to help improve the quality, clarity and accuracy of SOPs. These tools leverage a mixture of linguistic rules, NLP and ML techniques to identify potentially ambiguous wording, structure and terminology, and to identify directives with the SOP that when violated or circumvented can lead to significant increase in risk potential. The chapter is based upon, [P2], "Enhancing Procedure Quality: Advanced Language Tools for Identifying Ambiguity and

High-Potential Violation Triggers" submitted to Reliability Engineering & System Safety journal. The work was also presented at the European Safety and Reliability Conference (ESREL 2024), [C4], titled "Identifying Ambiguity And Potential Violations In Standard Operating Procedures Using Natural Language Processing Tools" (Johnson et al. 2024).
Chapter 6 consists of the mathematical/statistical background and technical implementation of the tools, in part based on work presented at UNCECOMP 2023 - 5th ECCOMAS Thematic Conference on Uncertainty Quantification in Computational Sciences and Engineering, [C2], titled "AI Tools for Human Reliability Analysis" (Johnson et al. 2023a).
Chapter 6 also discusses one final tool (*Human Factors Causal Relationships tool*), presented at ESREL 2022, [C1], "A Data Driven Approach to Elicit Causal Links between Performance Shaping Factors and Human Failure Events" (Johnson et al. 2022)).
Finally, Chapter 7 presents its the main contributions, implications and recommendations for the future.
All data supporting this thesis is available at `https://datacat.liverpool.ac.uk/1018/`. (Moura et al. 2016), and `https://safetyzone.iogp.org/` (IOGP 2024).
Additional methodological details and supplementary code are provided in Appendix A. The full source code for all developed tools is available at the following repositories,

- `https://github.com/VirtualRaphael/Human-Factors-Classifier-1.0`
- `https://github.com/VirtualRaphael/Human-Factors-Classifier-2.0`
- `https://github.com/VirtualRaphael/Human-Centric-Summarizer`
- `https://github.com/VirtualRaphael/Ambiguity-Identifier`
- `https://github.com/VirtualRaphael/Violation-Trigger-Tool`
- `https://github.com/VirtualRaphael/Human-Factors-Relationships-Tool`

Chapter 2

Background and Existing Studies

2.1 Human Reliability Analysis & Data

HRA involves the evaluation, assessment and estimation of HEPs, and the factors that influence it, in various operational contexts. To carry out quantitative HRA and calculate HEPs, a range of data sources are often utilized, each with distinct advantages and limitations. These sources include expert judgments, data from simulators, real operations, near misses, and accidents (Kirwan 1994).

Expert elicitation involves gathering insights and predictions from experienced practitioners in the field. This data source is useful for scenarios where empirical data is scarce or difficult to obtain. However, expert judgments can be subject to biases such as overconfidence and anchoring (Mosleh et al. 1988). Despite these limitations, expert elicitation remains an essential component of analysis, providing contextual understanding that complements quantitative data sources.(Laumann 2020).

Simulators are used to mimic real-world systems and environments, allowing operators to practice and be evaluated in a controlled setting. This data is valuable for studying human-machine interactions, especially in high-risk industries like nuclear power, aviation, and chemical processing. Data from simulators can often be limited to human-machine interfaces in control rooms and may not fully capture all PSFs due to the de-contextualized nature of the tasks studied. Other limitations include the artificial nature of the envir-

onment, operators know that their actions have no real consequences and that they are being observed, which can influence their behavior (Park et al. 2004).

Real operation data is typically considered the most reliable for estimating human error probabilities as it reflects real-world events and conditions. It provides direct insights from actual operational environments, capturing genuine human behaviors and error rates under real-world conditions (James et al. 2014). This authenticity ensures that the data reflects true operational stresses, constraints, and complexities. Real operation data encompasses a wide range of information, from routine performance logs to detailed incident reports. This breadth of data provides a holistic view of human performance and error patterns across different scenarios.

Real operation data includes near-miss reports. These are incidents that could have resulted in an accident but did not, often due to timely intervention or luck. By analyzing near-misses, organizations can identify potential hazards and error patterns before they result in significant consequences and implement preventive measures to mitigate risks proactively.

Accident reports are a critical component of real operational data, offering several unique advantages. These reports provide a detailed analysis of incidents, including a thorough examination of the interplay between human, machine, and organizational factors leading to failures. This in-depth analysis helps uncover root causes and contributing factors that may not be evident in other data sources.

One significant advantage of accident reports is their inclusion of root cause analyses. These analyses explore the underlying reasons for failures, offering a comprehensive understanding essential for developing effective strategies to prevent future incidents. Learning from past incidents is crucial for improving safety and risk mitigation, as past events offer valuable insights and recommendations for preventing future incidents.

The availability and accessibility of many accident reports provide a wealth of information for researchers and practitioners. Public access to these reports ensures that valuable lessons from past incidents can be widely disseminated and applied across different indus-

tries. This openness enhances the collective understanding of safety and risk management practices. Accident reports are invaluable for their detailed analyses, root cause investigations, accessibility, and the actionable insights they provide. By learning from these reports, organizations can enhance their safety practices and prevent future incidents.

2.1.1 MATA-D

The Multi-Attribute Technological Accidents Dataset (MATA-D) encompasses a substantial collection of 238 major accidents sourced from a variety of complex industrial sectors such as aviation, chemicals, oil & gas, nuclear, and waste treatment. This dataset provides a unique opportunity for cross-sectorial learning and analysis of major industrial accidents, focusing on human, organizational, and technological factors that contribute to such incidents (Moura et al. 2016)

MATA-D is compiled from detailed accident investigation reports, each thoroughly analyzed to distill contributing PSFs. These reports are primarily in English and averaging two hundred pages in length, offer rich insights into diverse accidents across multiple industrial settings. Of these, 110 reports are publicly accessible and have been crucial in both training and testing data analysis tools (Morais et al. 2020).

To classify the contributing human factors in these accidents, the Cognitive Reliability and Error Analysis Method (CREAM) was employed. Developed by (Hollnagel 1998), this framework categorizes errors and PSFs into organizational, technological, and individual elements. This taxonomy not only aids in accident analysis but also enhances risk assessment capabilities by pinpointing prevalent factors in accidents.

The dataset categorization involved a meticulous manual coding process, referencing the CREAM taxonomy to classify each report with Boolean values, indicating the presence or absence of specific factors (Moura et al. 2016). This process required a substantial time commitment, however ensured a high level of detail and accuracy. This is necessary for developing reliable human reliability models with confidence. This binary classification allows for the application of straightforward statistical methods or sophisticated models

to identify patterns and predict future risks. MATA-D, coded in Excel, provides an easy-to-use interface for researchers and practitioners to access and analyze data, available for download at `https://datacat.liverpool.ac.uk/1018/`.

The MATA-D capitalizes on reports from a wide range of sectors to support cross-industry learning, each characterized by its unique format and vocabulary. The diversity in report format is not only evident in the varying number of pages but also in the reproducibility of sections within the corpus (Morais et al. 2022a). Similarly, the vocabulary differs significantly, reflecting the specific jargon and taxonomy relevant to each industry's investigative methodologies. Despite the additional challenges presented by the varied structures and vocabulary, these allow the MATA-D to provide insights into the specific conditions and factors contributing to accidents across various industries but also supports the development of robust analytical models capable of interpreting complex and diverse data effectively.

Within the MATA-D there are two significant subsets of reports. First, it contains a collection of 57 reports from the US Chemical Safety and Hazard Investigation Board (CSB), focusing on industrial chemical accidents. The second subset is comprised of 20 reports from the US National Transportation Safety Board (NTSB), this subset focuses on transportation accidents (Moura et al. 2016).

MATA-D is a valuable resource for advancing HRA, offering comprehensive, cross-industry data on human factors in major accidents. Its detailed classification of human errors and PSFs, combined with ease of access, makes it an essential tool for enhancing predictive models and developing effective strategies to reduce human error and improve safety across various industries.

2.2 Machine Learning, Natural Language Processing in HRA

The availability of data has led to increased interest in data-driven decision-making and ML modeling across various industries. For instance, in aviation, HRA models have been developed to predict maintenance needs and enhance safety protocols, employing techniques such as the Human Error Assessment and Reduction Technique (HEART) and BN models to evaluate and manage human errors (Yazgan and E. 2024). In the healthcare sector, HRA is utilized to improve patient safety by systematically analyzing incident reports to identify and mitigate human errors (Sujan et al. 2020).

However, for these models to be reliable and reflect a broader range of possible situations, more data is required. Increasing the volume and variety of data enhances the models' robustness and helps in capturing more nuanced patterns that may not be apparent in smaller datasets.

ML and NLP have been identified as effective means to automate the collection of data, particularly through the classification of textual data.

2.2.1 Automated Classification

As might be expected, fields and industries that generate substantial textual data have always shown the most interest in ways to process and analyze this information.

At the beginning of this project, studies that used ML strategies to classify textual narratives into safety and risk features were identified. The sample also focused in industries with similar level of organizational and technological complexity as found in MATA-D, as well as those that have investigated at least one human factor as one of the features, such as aviation (Robinson et al. 2015), railway (Hughes et al. 2016; Heidarysafa and Brown 2018), oil & gas (Ribeiro et al. 2020), civil construction (Goh and Ubeynarayana 2017) and maritime industries (Grech and Smith 2002). A comprehensive review of the applica-

tion of ML techniques in occupational accident analysis, mixing multiple industries with lower level of complexity is provided by Sarkar and Maiti (2020).

Despite large research and application of ML approaches, gaps and needs for risk and reliability analysis remain. Previous studies have not classified full accident reports into a human reliability taxonomy, nor have any attempts been made to expand databases of human reliability with the support of ML, or within multiple industry sectors. This led to the first development of this thesis, the first-generation classifier (Morais et al. 2022a). Since the beginning of this project, interest in NLP has increased substantially, leading to a surge in the automation of text and report analysis and the inclusion of more advanced techniques.

The healthcare sector, with its daily influx of incident and patient safety reports, has become a focal point for the application of NLP methods and tools. A comprehensive review of such research is presented by Trinh et al. (2023), concerned particularly with the risk of falls.Within this domain, an approach focused on categorizing factors from patient safety event reports employs a methodology similar to the first-generation classifier. It utilizes Bag of Words (BoW) representations and machine learning models, enhanced by a systematic selection of information-rich sentences, which significantly improves categorization performance (Tabaie et al. 2023).

More recently developed approaches, using technologies such as BERT (Macêdo et al. 2022) and Contextual Word Embeddings (Macedo et al. 2023), have been leveraged to identify the causes of accidents across various industry sectors. However, these works have predominantly concentrated on specific industry domains, such as aviation (Jing et al. 2023) or Oil & Gas (Macêdo et al. 2022) rather than adopting a cross-industry perspective. Moreover, they tend to identify a limited array of specific factors, in contrast to this research's aim of comprehensively mapping the full spectrum of factors. As well as this, much of the existing literature on accidents is geared towards categorizing the severity of incidents, rather than uncovering their root causes (Ramos et al. 2022; Oliaee et al. 2023).

2.2.2 Summarization

The growing complexity and volume of data necessitate effective summarization techniques to make sense of extensive textual information and support efficient understanding and explainability of other automated analyses.

There are two main approaches to automated summarization, extractive and abstractive. Extractive summarization involves targeting and extracting key phrases, sentences, or segments directly from the source text, and then combining these to form a summary. Extractive summarizations consist of portions of the original text verbatim and is therefore generally very accurate in terms of factual correctness. However, these may lack coherence, be less fluent, and may miss out on conveying the overall essence of the original text. Abstractive summarization involves aiming to understand the main ideas of the source text and then expressing them in a new and concise way. These can produce more coherent and fluent summaries, that often capture the overall essence of the text more effectively than extractive methods. With this increased creativity there is a higher risk of inaccuracies or distortions of the original text's meaning, as the process involves paraphrasing and interpretation (Zhu et al. 2021). Inconsistency in the text structure of accident reports may prevent extractive summarization algorithm's ability to extract the key points, and variation in the written styles can affect the algorithm's ability to consistently identify the most relevant information. The success of extractive algorithms relies heavily on the quality and clarity of the source text, poorly written texts with industry specific language, ambiguous phrasing, and/or complex sentence structures can impede the model's ability to extract meaningful content. Extensive comparison of the two types of summarizations can be found in (Bhargav et al. 2022).

When it comes to abstractive summarization there are two main approaches, one is to fine-tune your model for your specific task, the other is to leverage the pre-trained summarization models on specifically targeted text. The main challenge with fine-tuning summarization tools is the data requirements. Fine-tuning often requires large, high-quality datasets that contain both the original texts and corresponding summaries (El-Kassas 2011). Gathering and preparing such datasets would be extremely time-consuming and resource-

intensive, comparable with the original construction of the MATA-D, if not greater due to the required quality of such summaries, as they should be accurate, coherent, and effectively capture the main points of the original text.

Recent developments of more sophisticated models has significantly enhanced the capabilities of text summarization tools. LLMs use deep learning techniques to understand and generate human-like text, enable them to create summaries that are not only accurate but also contextually rich and coherent with remarkable nuance and relevance (Liu et al. 2023).

There have been limited applications of such automated summarization approaches within the domain of accident reports. One approach, concerned with summarizing coal mine accident reports, specifically targets sections before summarizing/rearranged to optimize clarity (Zhao et al. 2020). The algorithmic approach, using the TextRank and Word2Vec packages, is designed to generate summaries of the entire reports, leveraging the consistent structure these reports typically follow. However, this contrasts with the objectives of this work, which has a dual focus: firstly, to summarize incidents, focusing specifically on human behavior and errors, as well as the influencing factors, and secondly, to be versatile enough for application across industry where variations in report quality and structure encountered are significant.

Our aim in developing the summarization tool is to create a more adaptable method that can effectively handle diverse reporting formats and content complexities, while harnessing the identified strengths of both summarization types.

2.2.3 HEP Modelling – Bayesian Networks

HRA involves understanding and modelling the factors that influence human performance and error. These factors are often interdependent, necessitating the consideration of their interrelationships (Groth and Mosleh 2012). This requirement has led to the development of causal models that explicitly capture these dependencies. Such models not only calculate HEP but also elucidate why errors occur and how they can be prevented.

BNs have become a popular choice for building these models due to their graphical structure, which clearly illustrates the causal links between PSFs and events. This visual representation makes the relationships easily understandable even to those not directly involved in the model's construction. The influence of these factors on each other can be extracted from the conditional probability tables (CPTs) included in the model (Groth and Mosleh 2012).

The use of BNs addresses several challenges within HRA, including the shift towards data-driven models. Analysts can combine information from various sources, such as empirical data (e.g. MATA-D) and expert opinions, to build these models. Successful applications of BNs to model HEP with differing levels of complexity can be found in (Groth and Mosleh 2011; Fan et al. 2022; Mkrtchyan et al. 2015; Podofillini and Dang 2013).

However, while empirical data is often used to estimate the conditional probabilities, the network structure is typically derived from expert judgment. This reliance on expert opinion to identify causal links can introduce biases, potentially overlooking some causal relationships between factors (Mkrtchyan et al. 2015).

2.3 Standard Operating Procedures

SOPs are pivotal in various sectors for enhancing consistent quality, safety, and reducing miscommunications. The primary benefits of SOPs include error prevention, facilitating knowledge transfer, and ensuring consistent guidelines are available (Gough and Hamrell 2009). The development of effective SOPs must involve those who perform the tasks, in a 'walk-through/talk-through' process, not only to encourage ownership and adherence of the end user, but also to enable the analyst or SOP writer to see if procedures are accurate reflection of how things are really done (Institute 2020). This process is usually known in human factors community as 'work as done versus work as imagined, and benefits from the "Plan-Do-Check-Act" cycle, a dynamic model that promotes continuous improvement through iterative updates and feedback (Amare 2012, Institute 2020).

Moreover, SOPs are instrumental in regulatory compliance and efficient business operations. They serve as a first line of defense during inspections and are legally binding documents, underscoring their importance in maintaining consistent, quality outputs across all operational processes. The literature highlights challenges when SOPs are outdated, contradictory, or not reflective of actual practices, which necessitates regular updates and internal compliance to align with true operational procedures (Gough and Hamrell 2009). This alignment is also crucial as SOPs and quality management systems are interdependent, evolving with the business to continuously improve processes and ensure product integrity.

Clear and high-quality design and writing are essential for effective SOPs. Hollman and Nechyporenko (2020) provided "Ten simple rules on how to write a standard operating procedure," which include identifying when an SOP is necessary, defining the purpose and scope and reviewing and approval of the SOP.

Responsibilities for writing, reviewing, and approving the SOP should be clearly assigned, and the SOP should be tested with colleagues, specifically the 'end users', to ensure its effectiveness. Regular reviews and updates are crucial to adapt the SOP to procedural changes and to improve the document continually (Ahmed et al. 2020).

Despite the critical role of SOPs in ensuring safety and compliance, their effectiveness can be undermined by ambiguity (Steen-Tveit et al. 2024). The MATA-D is a collection of major accident reports drawn from various industrial sectors such as aviation, chemicals, oil & gas, nuclear, and waste treatment (Moura et al. 2016). This dataset has used a framework categorizes errors and performance-shaping factors into organizational, technological, and individual elements. In this study, particular attention is given to the factor "Inadequate procedure". Analysis of the MATA-D entries reveals that 45% of the accidents were influenced by "Inadequate procedure" as a contributing factor.

Kim et al. (2022) conducted an empirical study on HEP related to procedure-extraneous behaviors, which are commission errors occurring when operators engage in actions not explicitly guided by procedures. Using data from a nuclear power plant simulator, the study estimated error probabilities. The findings highlight the importance of accurate and clear SOPs, as errors stemming from deviations or misinterpretations can lead to operational failures.

Ultimately, SOPs are relevant influencing factors that influence human performance when executing a task. The majority of HRA techniques cited by the review from Health and Safety Executive UK, account for procedure quality as a performance influencing factors (Bell and Holroyd 2009). Including, Technique for THERP, CREAM, HEART, Standardized Plant Analysis Risk-Human Reliability Analysis Method (SPAR-H), Accident Sequence Evaluation Program (ASEP), A Technique for Human Event Analysis (ATHEANA).

2.3.1 Identifying Ambiguity

The first challenge is how to automatically detect and identify steps and instructions that are ambiguous in some manner. This is a challenging problem, that has grown in significance with the increased popularity and development of research around NLP. This has necessitated the development of sophisticated tools to identify and measure ambiguity in natural language texts, a crucial aspect for enhancing communication, interpretation and understanding. Various strategies and tools have been proposed in recent research to tackle the identification and resolution of ambiguities in text.

Ambiguity in language can manifest in multiple forms, each type poses unique challenges in detection. Several studies have proposed different methodologies for addressing these. Ceccata et al. (2005) developed a tool for aiding writers by identifying ambiguous phrases in document. The tool utilizes lexical databases, like WordNet, to assess the potential meanings of each word and how the word's context within a sentence might affect its interpretation. This process aims to pinpoint words that could be understood in multiple ways, thus contributing to textual ambiguity. Once potential ambiguities are identified, the tool measures the extent of the ambiguity, based on a quantifiable scale derived from the number of possible interpretations a word or phrase may have. Similarly, later Kiyavitskaya et al. presented a two-step tool approach for software requirements specifications (Kiyavitskaya et al. 2008). The first tool identifies sentences that could be understood in

more than one way based on measures defined within the tool. While the second elucidates these ambiguities, leaving the resolution to human analysts.

The advent of computational linguistics has further aided the automation of ambiguity detection. The study presented by Gleich and Kof (2010) also discusses a tool that automates the detection of ambiguous statements in requirements documents. The tool uses computational linguistics techniques, specifically part-of-speech (PoS) tagging, to analyze texts for ambiguity. It categorizes ambiguities into different types based on existing frameworks like the Ambiguity Handbook (Berry et al. 2003). The tool aims to not only detect ambiguities but also to educate the users about potential sources of ambiguity in their documents.

A recent exploration into the use of AI (artificial intelligence) tools, like ChatGPT, versus traditional rule-based methods provides insightful comparisons (Fantechi et al. 2023). This study evaluates the effectiveness of AI-driven approaches against deterministic rule-based systems in detecting ambiguities within software requirements documents. Findings suggest that while AI tools show promise, they require significant further enhancement to match the consistency and reliability of rule-based approaches. The lack of pre-labeled or annotated corpus of documents poses a significant challenge for training AI models to analyze texts effectively. NLP tools, which rely on ML, greatly benefit from large-scale, example-driven data for training (Valcamonico et al. 2024). Creating a sufficiently annotated dataset to train a purely AI-driven tool would be both difficult and time-consuming.

2.3.2 Procedure Violations

For procedure guides to be an effective tool in safety management, they not only need to be clear and unambiguous, but also must truly reflect real-world operations. Violations refer to instances where individuals or groups deviate from established safety protocols or procedures, typically in a non-malicious manner (Dougherty 1995), and are known to increase the risk of accidents in industrial contexts. These deviations can be intentional and often vary based on their circumstances and perceived necessity.

There are three main types of violations (Reason et al. 1998). Routine violations are deliberate deviations where individuals bypass established procedures, that they perceive as inconvenient or unnecessary. This type of violation is often habitual and occurs when the consequences of non-compliance are minimal or unenforced (Lawton 1998). An example of this would be in a manufacturing setting, an operator might routinely skip a machine's shutdown process to save time, assuming it poses no immediate risk. There are also situation violations, these occur when individuals or groups break rules or norms due to specific circumstances or situations. These violations are often influenced by external pressures or constraints within a particular context (Lawton 1998). These are typically not premeditated and occur because of factors like stress, lack of resources, or unforeseen circumstances that compel individuals to deviate from expected behaviors. For example, a worker might ignore safety protocols to meet a critical deadline. Then there are exceptional Violations, these are rare and occur under extreme conditions or in high-pressure scenarios where the usual protocols appear insufficient. These are emergency-driven decisions where individuals take extraordinary steps to mitigate perceived threats (Lawton 1998). During a critical system failure at a nuclear facility, engineers might bypass certain safety checks to initiate a faster shutdown, believing it is the only way to prevent a meltdown.

Violations can undermine safety culture, promoting a careless attitude towards regulations and increasing the risk of accidents. While some violations may appear to save time or resources initially, they often lead to greater inefficiencies, such as equipment damage and increased maintenance needs, due to improper use.

The systematic reviews by Alper and Karsh (2009) and Boskeljon-Horst et al. (2024) explore the empirical causes and systemic solutions to violations. Their study provides an exhaustive examination of the intentional, though not malevolent, breaches of safety norms in various industries. The review identifies key causes of such violations through a comprehensive literature search focusing on sectors including healthcare, aviation, mining, railroads, and construction. The works identifies potential causes, including,

- Individual Characteristics: Attributes such as age, experience, and personal attitudes towards safety compliance.

- Information/Education/Training: The role of adequate training and the dissemination of information in preventing violations.
- Design to Support Worker Needs: How workplace design can influence the occurrence of safety violations.
- Safety Climate: The impact of organizational culture on safety practices.
- Competing Goals: Workplace demands that might conflict with safety compliance.
- Problems with Rules: The clarity and applicability of safety rules and their enforcement.

The review advocates for viewing violations not just as failures of individual workers but as indicators of broader systemic issues within organizational safety protocols (Alper and Karsh 2009). When tasks in a SOP are not possible to execute, due to physical constraints or to poor quality written procedures, there is high potential that the end users are not going do the work as imagined by the SOP writer (Institute 2020).

Focusing on Norwegian offshore supply bases, Boskeljon-Horst et al. (2024) address the challenge of aligning formal safety procedures with actual workplace practices, emphasizing that procedural violations are often symptomatic of larger systemic issues rather than mere non-compliance by workers. The study demonstrates that simplifying procedures and involving workers in procedural design significantly enhances compliance and safety perception. It stresses the importance of closing the gap between 'work as imagined' and 'work as actually done'.

These studies underscore the importance of systemic approaches to managing safety violations, suggesting that addressing these issues requires a multifaceted strategy that includes improving procedural clarity, worker training, and safety culture. By adopting such approaches, industries can enhance their overall safety standards and reduce the incidence of accidents caused by violations. This alignment between theoretical safety procedures and practical application on the ground is crucial for building safer work environments.

Although violations are known to disrupt the integrity of procedure documentation and completed safety analyses, it is also known that impractical rules encourage violation (Health and (HSE) 1995). Incorporating feedback from ground-level operations into procedural documentation is crucial for making safety protocols practically applicable and

helping safety analysts identify risks more accurately and develop effective mitigation strategies (Hollnagel 2017).

Violations of certain procedural steps, especially at critical times, can significantly increase risk and lead to accidents. Some steps are more prone to violations, and when these steps have a high likelihood of being violated and a significant impact on risk, it becomes a major concern. Therefore, identifying such procedural steps is therefore crucial to minimizing risk and preventing potential accidents.

Through the critical examination of SOPs and their pivotal role in ensuring safety and compliance, key benefits and potential pitfalls of SOPs in various industries have been identified. The recurring issue of outdated, ambiguous, or misaligned procedures points to a significant challenge in maintaining operational integrity, especially when procedures are not reflective of actual practices. These insights serve as the foundational motivation for the project, which seeks to develop tools to detect ambiguities and identify steps prone to non-malevolent violations in procedural documentation. While the aim is to support more efficient and accurate communication of safety-critical information, it is equally important that any changes to SOPs or implementation of safety measures are based on a deep understanding of operational contexts. SOPs must be carefully analyzed and thoughtfully updated to ensure they accommodate the full range of real-world scenarios they are intended to govern. Fast communication should support and not replace rigorous and evidence-based decision-making when it comes to safety interventions.

Chapter 3

Machine Learning Approach to Automated Human Reliability Data Collection

One of the most acknowledged ways to prevent design errors in complex industries is to conduct risk assessment, where multi-disciplinary teams revise a design according to information from past accidents, components, and human reliability. There are industrial recommended practices on how companies should use lessons learnt from past accidents (CCPS 2010), research on how they are actually using it (Drupsteen et al. 2013) or how it could be used (Moura et al. 2017b, Moura et al. 2017a). The lessons learnt encompass not only hazards but also their frequency of occurrence, which are used to quantify risks in probabilistic risk analysis, or to estimate order of magnitude in semi-quantitative analysis (e.g. Layers of Protection Analysis (LOPA)) and qualitative analysis when risk ranking is required (Baybutt 2015).

Regarding frequency, component failure databases play a central role in quantitative risk analysis, where data is majorly provided by components manufacturers and sometimes shared within groups of industry operators, such as the Maintenance Steering Group (MSG-3) in aviation (Gonçalves and Trabasso 2018) and the Offshore and Onshore Reliability Data (OREDA) in upstream oil & gas (Lima et al. 2019). However, there is still plenty of space for the development of databases to support system safety, which should

be able to include systems and installations rather than only components' parts, as well as the interaction between human, organizational and technological factors (Leveson 2020). To fill this information gap, the MATA-D (Section 2.1.1) was created (Moura et al. 2016). Although it is already possible to use it for HRA (Morais et al. 2022b), it is desirable to reduce its uncertainty, leading to more precise risk estimates. To understand how to decrease its uncertainty, it is important to understand the different representation of the uncertainties within the dataset: aleatoric to model uncontrollable events, e.g. impairments and cognitive bias, or epistemic/reducible uncertainty due to missing data and theoretically reducible (Patelli 2016). It is acknowledged in the human reliability field that human behavior is dependent on the context, varying according to organizational and technological factors Hollnagel 1998. The lack of information on these factors' interactions (seldomly observed and reported) is the major contribution to the epistemic uncertainty. Thus, to reduce epistemic uncertainty it would be desirable to expand the database, by collecting more accident reports and classifying them in order to increase the chance of describing more human-machine-organization interactions.

However, collecting empirical data is time-consuming and expensive, especially in human reliability field, where data collection and classification are usually done by other humans (experts in their fields). MATA-D database have been constructed through extensive reading and classifying 238 accident investigation reports (Moura et al. 2016), a task that have taken around one year to be completed. The classification also required specialized knowledge, as the assessors had to be minimally trained on the taxonomy used to pursue the classification.

It is therefore proposed to enlarge the human reliability dataset by replacing (or supporting) human coding by automated classification of accident reports from any industrial sector using a pre-defined human factor's taxonomy. In order to absorb lessons learnt from different industry sectors, the objective is to continually add to the dataset reports only from industries with the same level of complexity regarding the interaction of organizational structure, technology and humans (Moura et al. 2016). The aim is not only to expand MATA-D, but to do it faster and timely. The use of a ML strategy for text recognition and classification is herein proposed, as an experienced expert takes around 3 days to read and classify one accident report, which contains about two hundred pages,

whereas a ML approach would take less than one minute. Thus, the development of a computer support, that could support risk specialists, or directly collect and update the database for every new accident report of interest, is proposed. Caution would be needed on the acceptance criteria of this new data, as depending on the sample quality the uncertainty might increase (Siegrist 2012). Therefore, a central research question of this is whether a ML approach is capable of both accelerating the expansion of a human reliability database and maintaining the same data quality offered by human experts.

The approach, here named *Virtual Human Factors Classifier* (and *HF Classifier* going forwards) might be useful in other ways. For instance, it may be used to improve human reliability models (Morais et al. 2022b) or to support cross-learning from different industry sectors. It can also support incident investigators in an unbiased fashion to consider possible PSFs, which might have triggered human errors (instead of focusing only on human errors). On the original aim of expanding MATA-D, risk assessors should benefit from the provision of more data, providing more possible combinations between PSFs and human errors, and minimizing missing data problem in probabilistic approaches.

This chapter has been divided into three parts. The first section will examine the machine-learning strategies, the second section discusses the methodology and implementation. The third section is concerned with evaluating the performance of the developed tool, before section four explores two case studies based on accident reports from aviation (Boeing 737 MAX) and oil & gas industry (FPSO CDSM, Cidade de Sao Mateus floating production storage and offloading unit). The chapter concludes with a brief conclusion and some discussion of any limitations.

3.1 Automated Text Analysis

Before classifying a document, the text features need to be extracted to generate a representation of the document, capturing the properties that are important for further classification (Goldberg 2017). There are many feature extraction methods available, some popular options at the time of the tools development were BoW, Term Frequency - Inverse Document Frequency (TF-IDF) and word2vec (Waykole and Thakare 2018).

A BoW model extracts features from the text, specifically the vocabulary of known words and their frequency of occurrence. The reason the model is called a 'bag' of words is that it does not consider any information about the order or structure of words. To use it on a set of documents, data is collected from text files and organized into a list, forming a vocabulary. To improve results and save computational time and memory the model ignores case, punctuation, and other frequent words that do not contain relevant information, such as stop words (e.g. "*a*", "*the*", "*of*"). To score the known words in each file (i.e. document), their presence is marked as Boolean values (0 and 1) – thus, using the list of words previously prepared, each new file is analyzed and converted into a binary vector. To extract features from files, the order of words is discarded (Brownlee 2020).

Bag-of-bigrams is a special case of feature combinations that counts consecutive word sequences of a given length, which proves to be more powerful than BoW, as word-bigrams are more informative than individual words. However, it is difficult to know a-priori which bigrams will be useful for a specific task, thus the modeler should assign the less important combinations previously with low weights. Bag of trigrams are also common, differently from 4-grams and 5-grams that are sometimes used for letters, but rarely for words due to sparsity issues (Goldberg 2017).

TF-IDF accounts for the frequency of each word in a set of documents and its useful to give higher scores to domain specific words, something that is considered a drawback for BoW (as domain specific words which does not have higher frequency within a document may be ignored). TF-IDF reduces the score of frequent words in a document that are also frequent among all the documents, highlighting the words that are unique (Hughes et al. 2016, Waykole and Thakare 2018).

Word2vec assumes that words that occur in the same contexts tend to have similar meanings (Kim et al. 2020), thus models constructed by word2vec algorithms will place words with common contexts next to each other in a vector space (Heidarysafa and Brown 2018, Waykole and Thakare 2018). Word2vec models are two-layer neural networks, and depending on their architecture they are able to consider nearby context words more heavily than words with distant context (i.e. continuous skip gram), or to not account for context at all (i.e. continuous bag-of-words) (Waykole and Thakare 2018).

3.1.1 Classifying text features

After the text relevant features are captured from the document and represented in a model, they are ready to be classified by a machine-learning technique. At the time of the tools first development, some of the most known and broadly tested techniques for automated text classification were the dictionary method, Naïve Bayes, support vector machines (SVM), latent Dirichlet allocation (LDA), latent semantic analysis (SMA), structural topic model (STM) (Kim et al. 2020). Aside from the dictionary method, they can be mostly divided into supervised and unsupervised learning methods (some authors further distinguish semi-supervised approaches, in which the training set contains a small amount of data with known categories and a large amount of data with unknown categories (Ratsaby and Venkatesh 1995). The method selection might be based on how texts are going to be classified, and if some documents have been previously classified by humans (allowing their use as examples to train the machine) (Goldberg 2017, Kim et al. 2020). As the classification categories are known and predefined, a supervised learning approach, a dictionary-based or rule-guided method, is appropriate for training the model for the *HF Classifier*. This approach enables the algorithm to learn direct associations between textual inputs and their corresponding HF categories, leveraging labeled data to optimize classification accuracy and reliability.

In dictionary-based methods, the machine uses predefined set of words to infer particular features of a text, relying on the user defined dictionary. In such methods, the categories of

interest are represented by single words, which are searched by an algorithm through large bodies of text (Kim et al. 2020, Iliev et al. 2014). In the classification of organizational factors in accidents, it would be equivalent to define into the algorithm that every time the words and expressions work shift, jetlag, lack of sleep, circadian rhythm are found in the text the algorithm should classify it as the organizational factor of irregular working hours.

Naïve Bayes and SVM are popular supervised learning methods for text classification. Naïve Bayes is a simple Bayesian classifier which assumes that all attributes are independent of each other, thus independent of the word context and position in the document (Zubrinic et al. 2013, McCallum and Nigam 1998). Naïve Bayes classifiers are reported to have better resilience to missing data than SVM classifiers (Shi and Liu 2011), which potentially makes naïve Bayes better to analyze fragments of texts (e.g. few paragraphs) and SVM to classify whole documents (Goh and Ubeynarayana 2017, Wang and Manning 2012)

SVMs are one of the most popular supervised ML algorithms, due to its little need for adjustments, and to due to their excellent prediction and generalization capabilities (Goh and Ubeynarayana 2017, Arrieta et al. 2020). They can be used for classification, regression, or other tasks such as outlier detection (Arrieta et al. 2020). The SVM algorithm constructs a hyper-plane (or a set of them) in a high-dimensional space, so that a good separation between classes is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (Arrieta et al. 2020). The simplest case, when data have only two classes, a SVM classifies data by finding the maximum-margin hyperplane which separates the data points of one class from those of the second class. The support vectors cross the data points that are closest to the hyperplane that separate the classes. As SVM is a supervised learning model, it has to be trained before it cross-validates the classifier. Only then, the trained machine can be used to predict or classify new data. SVM is usually suggested if features' interaction might be important for classification, similar to a semantic space, as learned hyperplane separates documents belonging to different topics in the input space (Zubrinic et al. 2013). While the literature often suggests that more complex problems may benefit from using alternative SVM

kernel functions to improve predictive accuracy, previous studies indicate that this is not always the case. For example, the linear kernel has been shown to outperform non-linear polynomial kernels in certain contexts, including multi-word classification tasks where contextual information of individual words is taken into account (Zhang et al. 2008).

3.2 Methodology

For the development of the *HF Classifier*, SVM is proposed to automatically evaluate and classify accident reports into potential human factors, with the support of BoW model for data extraction. The model was trained and tested using data from MATA-D. This section better describes the procedures applied to train and test the models.

3.2.1 Dataset

The decision to choose and focus the work on the MATA-D (detailed in Section 2.1.1) was based on its conceptual advantages, specifically the potential for cross-learning lessons from accidents across different sectors. Additionally, it offers two technical advantages regarding machine-learning application. Firstly, the majority of accident reports were available for training and testing the machine-learning model against expert-classified opinions. Secondly, the dataset has a specific taxonomy, simplifying the decision on the automated text technique to use.

The reports in the MATA-D dataset cover various industry sectors, they present different formats and vocabularies. The format changes not only in terms of the number of pages but also in terms of reproducible sections in a corpus. The vocabularies vary not only due to the specificity of the different industrial sectors but also due to the taxonomy applied, usually connected to the investigation methodology. This variety in vocabulary trains the

tools to handle a wide range of accident reports from different industries.

It is important to note that although the MATA-D dataset contains information on how 238 accident reports have been labeled against the CREAM taxonomy, only the publicly available reports were used to train and test the *HF Classifier* in the presented version.

3.2.2 Machine Technique

Since the classification categories are known (i.e., predefined taxonomy) and the dataset was previously labeled by experts, a supervised learning method is the most appropriate. This narrowed the decision down to either Naïve Bayes or SVM. Naïve Bayes classifiers have been shown to perform better with missing data (Shi and Liu 2011), making them a good choice for identifying human factors interactions in major accidents, which are considered rare and uncertain events (Morais et al. 2020). However, SVM has the potential to better capture feature interactions and classify larger documents more effectively (Zubrinic et al. 2013, Wang and Manning 2012). Given that interaction patterns have been observed between MATA-D factors (Moura et al. 2017a) and the goal is to apply the tool to accident reports averaging 200 pages, an SVM model with a linear kernel was chosen for classification.

BoW was selected as the feature extraction tool to pre-process the features to be classified by SVM. The choice was not only due to its recognized simplicity and flexibility (Waykole and Thakare 2018), but also because the intention to classify accident reports with no specific sector or domain suggested that it was better not to use models that capture too much the context from the training set into account – to avoid giving much higher importance to sector specific words or set of words (Goldberg 2017). A simplified workflow of the proposed approach is shown in Figure 3.1.

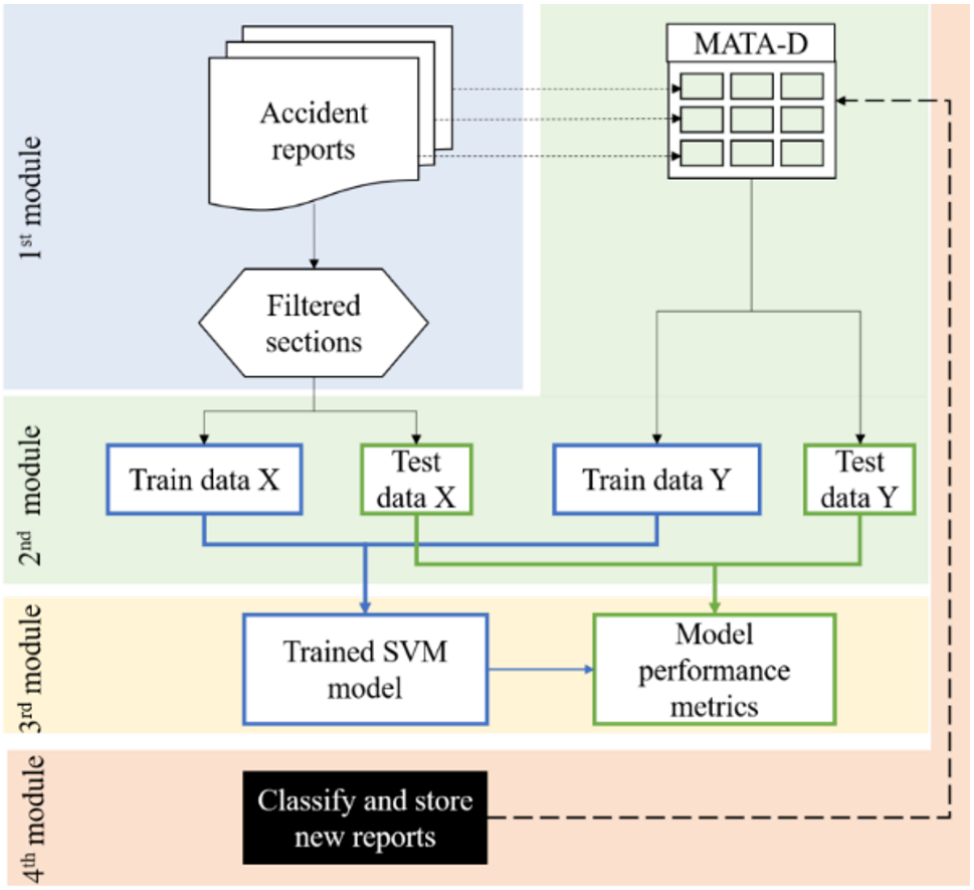

Figure 3.1: Simplified Workflow - HF Classifier

In the first module, accident investigation reports are analyzed. The documents in portable document format (i.e., files with PDF extension) were processed to check if the text in files was recognized by the machine and, if not, an optical character recognition software (OCR) was used to convert them to text files, an important step for relatively old accident reports.

After this pre-treatment, the tool scanned the accident reports, and their texts were sent to the next module. The most likely start and end of the targeted sections, recommendation and lessons learned, are identified by a confidence scoring system (detailed in Section A.1.2), and these sections are the output into the next module. Finally, the text was pre-processed to clean punctuation, stop words, and reduce words to their stem (e.g., "*testing*" was reduced to "*test*").

In the second module the tool took each accident report's file name and found the corresponding entry in the MATA-D. For this reason, the accident reports had equally assigned

names in dataset and correspondent PDF file. This gave the machine-learning component the desired output for each accident report, which was a combination of selected section texts and their known human factors.

Then, the selected text was converted into BoW objects (X in Figure 3.1 forming the input of the model), and the factors extracted from the MATA-D (Y in Figure 3.1 served as the output of the model). The module partitioned the data into a training set (80% of total) and a testing set (20% of total).

In the third module, the model based on SVM was trained and tested using data input from the previous two modules. Finally, the parameters of the classifier were recorded and overall performance metrics (accuracy, precision, recall and F1-score) were calculated based on test sets in all categories. Only then, the tool was prepared to be used in the next module.

The fourth and final module of the tool allowed users to add a new report that was not yet part of the MATA-D. The result was a list of the human reliability factors identified by the tool (an array of the predicted positive factors), a small table with all positives and negatives predictions (the 53 factors of the chosen taxonomy), and a word cloud of the most relevant words in the report.

3.2.3 Implementation

All the computational work was carried out using MATLAB software, and supported by the `text analytics toolbox`, which used the BoW model to extract text strings from files and prepare data for the ML algorithm. The MATLAB `statistics and the machine-learning toolbox` was used to transform text inputs into binary classification adopting the SVM. Data was extracted from the Excel based MATA-Dataset, while the accident report were in portable document format (i.e., PDF extension). The text recognition software embedded in Adobe Acrobat Pro was used to convert text-images to text-strings in cases where original reports had been saved as images (e.g. relatively old

accident reports, such as the Public Inquiry into the Piper Alpha Disaster (Cullen 1993). The dataset MATA-D with labeled classifications of each report is available at: `https://doi.org/10.17638/datacat.liverpool.ac.uk/1018` (Moura et al. 2016). Detailed discussion on the tools technical background is included in the Appendix, Section A.1.

3.3 Evaluating Performance

For evaluating the performance of classification models, first the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) need to be identified. These terms represent the different possible outcomes of the predictions made by a classification model when compared to the actual labels in the dataset (Goh and Ubeynarayana 2017). TPs refer to the instances where the models correctly predicts the positive class. For example, in this case if Missing Information was classified as a factor in the accident by the tool and by the expert in the MATA-D, this would a TP. Whereas a True Negative (TN) is where the model correctly predicts the negative class, or in this case the absence of an influencing factor.

FPs, also known as Type I errors, occur when the model incorrectly predicts the positive class, so if the model were to predict Inadequate plan as a factor in the accident and the expert did not in the MATA-D this would be a FP. FNs, or Type II errors, happen when the model fails to detect a positive condition (a present factor).

Different errors have distinct consequences depending on the domain of use. For example, in the field of safety management and in accident analysis, a FN (failing to identify a PSF leading to an accident) is typically more severe than a FP (incorrectly identifying a factor as contributing to an accident). This is because a FN could result in overlooking critical factors, potentially leading to unmitigated hazards in future situations. Conversely, a FP in this context, while potentially less dangerous, may cause unnecessary allocation of resources towards non-critical issues, which could divert attention and resources away from more significant safety threats.

However, informing trend analysis and policy decisions, FPs can be more detrimental. If irrelevant factors are consistently misclassified as contributors across a large dataset, they may skew statistical analysis and risk assessments, leading to misplaced priorities and ineffective interventions. In such cases, the cumulative effect of FPs can create systemic misdirection, while occasional FNs may have a lesser impact on long term safety strategy. These concepts can also be used to define several key performance metrics, that each serve to quantify different aspects of the model's performance.

Accuracy is a straightforward and intuitive measure of a model's overall correctness, as it represents the ratio of correctly predicted observations, both positive and negative, to the total number of observations (Shung 2018).

$$\text{Accuracy} = \frac{TP + FP}{TP + TN + FP + FN} \tag{3.1}$$

However, its usefulness can be limited in scenarios where there is an imbalance in class distribution. In such situations, a model could still achieve high accuracy by predominantly predicting the majority class, but this would not necessarily reflect its effectiveness in identifying the minority class, which could often be more critical.

The MATA-D is an example of an unbalanced dataset, as a typical accident entry in the dataset is attributed with only a few factors (an average of 46 negatives out of the 53 factors were identified per incident) (Morais et al. 2022b). So, when evaluating the model's performance other metrics are essential.

Precision measures only the correctness achieved in positive prediction. It indicates the proportion of positive identifications that were correct and is particularly important in scenarios where false positives are a significant concern (Shung 2018).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.2}$$

Whereas Recall assesses a model's ability to identify all relevant instances within a dataset. Recall is crucial in situations where failing to detect positives can have severe consequences (Shung 2018).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.3}$$

Finally, F1-score represents the harmonic mean of precision and recall, providing a single metric that balances both the concerns of precision and recall (Shung 2018). It is especially useful when the classes are imbalanced and both error types are of concern.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

The performance of a classification model cannot be adequately assessed by a single metric. A comprehensive evaluation requires considering accuracy, precision, recall, and the F1-score, along with the context and requirements of the model.

In the construction of the *HF Classifier* models, 80% of the available data was used for training purposes, and the remaining 20% was withheld to serve as a test set to evaluate the model's performance. The discussed metrics, accuracy, precision, recall and F1-score, were calculated for each of the models during the iterative training process. As each factor, and therefore model, are of equal importance in the overall assessment of the *HF Classifier* the macro-average was then calculated.

$$\text{Macro Average Metric} = \frac{1}{N} \sum_{i=1}^{N} \text{Metric}_i \tag{3.5}$$

Where N is the number of models, and Metric_i is the calculated metric for the i-th model (Leung 2022.

3.3.1 HF Classifier Performance

To measure the performance of the *HF classifier*, the binary classifications available in MATA-D were used as target classes. The performance metrics achieved are given in Table 3.1.

Metric	HF Classifier
Accuracy	86%
Precision	60%
Recall	46%
F1-Score	52%

Table 3.1: HF Classifier Performance Metrics

In this study, the collective SVM models trained with all public reports have achieved an accuracy of 86%, precision of 60%, recall of 46%, and F1-score of 52%. The results obtained have performed similarly to the benchmarked studies, which have presented accuracies from 44 to 90% (Robinson et al. 2015, Heidarysafa and Brown 2018), Ribeiro et al. 2020); precision from 22 to 84% (Goh and Ubeynarayana 2017, Grech and Smith 2002, Robinson et al. 2015); recall from 63 to 89% (Goh and Ubeynarayana 2017, Grech and Smith 2002); and F1-scores from 33% to 71% (Heidarysafa and Brown 2018, Ribeiro et al. 2020, Goh and Ubeynarayana 2017)

It has been noted, that performance benchmarking has to consider not only the numbers, but the descriptions from authors from previous works. For example, when more reports have been tested in (Grech and Smith 2002) the precision of 84% dropped to 48%, and the recall, which before achieved 89%, has not been calculable (due to zero TPs and FNs). To provide a visual aid, a word cloud has been generated to inspect the BoW contents used in the training process Figure 3.2. As the BoW representation is based on stemmed tokens, the words appear in their post-stemming form. The size of each word in the cloud corresponds to its frequency within the training text. Another important type of performance is the training time required by the ML algorithm. The elapsed time taken for the linear SVM to train and test with all reports was approximately one minute, once the models are trained these can be stored and a new report can be evaluated in around 30 seconds.

Utilizing this approach can drastically reduce the time required for task completion com-

Figure 3.2: Word Cloud of Report Training Text Data (after word stemming)

pared to relying solely on an expert. While users may be hesitant to fully trust the system initially, it provides significant support, enabling them to focus their evaluation efforts more effectively. This allows users to either expedite their work or use it to check their work, ultimately enhancing overall efficiency and accuracy.

3.4 Case Studies

In order to test the model on new accident reports (not yet in the MATA-D), two investigation reports from different industry sectors (aviation and oil & gas) were chosen to be analyzed. These new reports were classified by the same expert that classified the reports in the MATA-D. The results of the automated classification were not shown to him before the task, to avoid any bias. The results shown in Table 3.2 and Table 3.3 present the results when the tool analyzed the full report.

3.4.1 Aviation case study – 2018 Boeing 737 MAX 8 Aircraft final accident report

On October 2018, an accident with a Lion Airline aircraft, led to 189 fatalities ()KNKT 2019. Five months later, in 2019, an Ethiopian Airlines plane crashed minutes after take-off, killing all 157 onboard (Marks and Dahir 2020). The fact that both accidents involved the same aircraft model, a Boeing 737-8 MAX, had concerned civil society and safety regulators about the possible common flaws, which resulted in all 387 planes with same model grounded globally (BBC 2019). The two events have been famously known by the potential design flaws of the Maneuvering Characteristics Augmentation System (MCAS) which might have mislead the pilots' actions (Chronopoulos and Guzman 2020).

The *HF Classifier* is tested here on the final accident report of the Lion Air Aircraft flight, issued on October 2019, approximately one year after the accident (KNKT 2019). The final accident report was previously classified by the same experts which have classified MATA-D within the CREAM human factors taxonomy, in order to compare their similarity in new reports. Table 3.2 shows the comparison between the human factors classifications obtained with human coding and the *HF Classifier*.

The table has been color coded according to the legend below to help the reader understand how the model prediction metrics were calculated.

True positives: dark green (expert classified as '1' and machine predicted correctly as '1')

True negatives: light green (expert classified as '0' and machine predicted correctly as '0')

False negatives: dark red (expert classified as '1', but machine wrongly predicted as '0')

False positives: red (expert classified as '0', but machine wrongly predicted as '1')

				Expert Classification	HF Classifier
HUMAN	Action	Execution (Error Modes)	Wrong Time	1	0
			Wrong Type	0	0
			Wrong Object	0	0
			Wrong Place	1	1
	Specific Cognitive Functions	Observation	Observation Missed	0	0
			False Observation	0	0
			Wrong Identification	0	0
		Interpretation	Faulty diagnosis	1	1
			Wrong reasoning	0	0
			Decision error	0	0
			Delayed interpretation	1	0
			Incorrect prediction	0	0
		Planning	Inadequate plan	1	0
			Priority error	1	0
	Temporary Person Related Functions		Memory failure	0	0
			Fear	0	0
			Distraction	1	0
			Fatigue	0	0
			Performance Variability	0	0

Continued on next page

			Expert Classification	HF Classifier
		Inattention	0	0
		Physiological stress	0	0
		Psychological stress	0	1
	Permanent Person Related Functions	Functional impairment	0	0
		Cognitive style	0	0
		Cognitive bias	0	0
TECHNOLOGY	Equipment	Equipment failure	1	1
		Software fault	0	0
	Procedures	Inadequate procedure	1	1
	Temporary Interface	Access limitations	0	0
		Ambiguous information	1	0
		Incomplete information	1	0
	Permanent Interface	Access problems	0	0
		Mislabeling	0	0
ORGANIZATION	Communication	Communication failure	1	0
		Missing information	1	1
	Organization	Maintenance failure	1	1
		Inadequate quality control	1	1
		Management problem	1	0
		Design failure	1	1

Continued on next page

		Expert Classification	HF Classifier
	Inadequate task allocation	1	1
	Social pressure	0	0
Training	Insufficient skills	1	1
	Insufficient knowledge	1	1
Ambient Conditions	Temperature	0	0
	Sound	0	0
	Humidity	0	0
	Illumination	0	0
	Other	0	0
	Adverse ambient conditions	0	0
Working Conditions	Excessive demand	1	0
	Inadequate work place layout	0	0
	Inadequate team support	1	0
	Irregular working hours	0	0
		Sum of true positives	11
		Sum of true negativies	30
		Sum of false positives	1

Continued on next page

	Expert Classification	HF Classifier
	Sum of false negatives	11
	Accuracy	77%
	Precision	92%
	Recall	50%
	F1-Score	65%

Table 3.2: Virtual expert vs. Expert classification for Lion Airline accident report (Boeing 737-8MAX)

The following factors were observed by the classifier in the Lion Air accident operating with the Boeing 737 MAX: human error of execution of wrong place (i.e. action out of sequence); the cognitive function failure of faulty diagnosis; the technological factors of equipment failure and inadequate procedure; the organizational factors of missing information, maintenance failure, inadequate quality control, design failure, inadequate task allocation, insufficient skills, insufficient knowledge.

The confusion matrix heatmap in Figure 3.3 visualizes the classifier's performance in identifying contributing factors to the Lion Air accident. It shows that while the model correctly recognized several relevant causes and dismissed many irrelevant ones, it also missed a significant number of actual contributing factors, indicating areas where the model's ability could be improved.

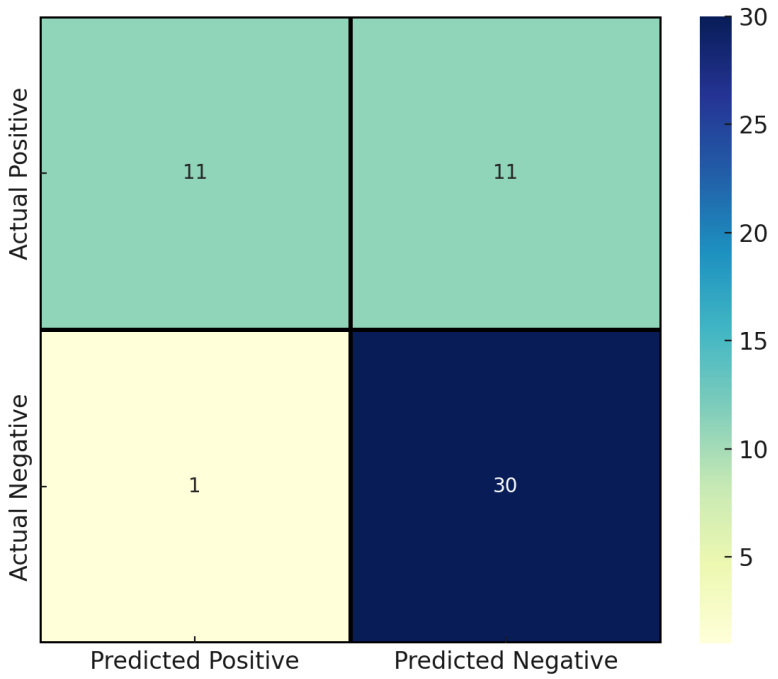

Figure 3.3: Confusion Matrix Heatmap - Lions Air accident

The word cloud is included as it serves as an additional support for the user to check if the information in the report is being correctly extracted or if there are problems that deserve any intervention to improve the prediction performance. It is also interesting to visually compare the differences between the word clouds obtained with the new report, Figure 3.4, with the word cloud of the training set, Figure 3.2.

Figure 3.4: Word Cloud for the Boeing 737 MAX accident report

3.4.2 Oil & Gas case study: FPSO CDSM accident report

On February 2015, an explosion onboard FPSO CDSM killed nine, injured 26 workers, as well as caused damage to the installation, and production halt of two gas production fields. The Brazilian Oil & Gas regulator (ANP) included in their investigation report root causes from the design phase to the emergency response. The FPSO (floating production, storage and offloading unit) was operated by BW Offshore in gas fields under concession to Petróleo Brasileiro S.A (Petrobras) in Brazilian waters (ANP 2020).

The FPSO CDSM accident report was also classified by the same experts as the MATA-D and the Lion Airline report. Error! Reference source not found. shows the comparison between human factors classifications obtained with human coding and the *HF Classifier*.

				Expert Classification	HF Classifier
HUMAN	Action	Execution (Error Modes)	Wrong Time	0	0
			Wrong Type	0	0
			Wrong Object	0	0
			Wrong Place	1	0
	Specific Cognitive Functions	Observation	Observation Missed	1	0
			False Observation	0	0
			Wrong Identification	0	0
		Interpretation	Faulty diagnosis	1	0
			Wrong reasoning	1	0
			Decision error	0	0
			Delayed interpretation	0	0
			Incorrect prediction	0	0
		Planning	Inadequate plan	1	0
			Priority error	0	0
	Temporary Person Related Functions		Memory failure	0	0
			Fear	0	0
			Distraction	0	0
			Fatigue	0	0
			Performance Variability	0	0

Continued on next page

			Expert Classification	HF Classifier
		Inattention	0	0
		Physiological stress	0	0
		Psychological stress	0	0
	Permanent Person Related Functions	Functional impairment	0	0
		Cognitive style	0	0
		Cognitive bias	1	0
TECHNOLOGY	Equipment	Equipment failure	0	0
		Software fault	0	0
	Procedures	Inadequate procedure	1	1
	Temporary Interface	Access limitations	0	0
		Ambiguous information	0	0
		Incomplete information	1	0
	Permanent Interface	Access problems	0	0
		Mislabeling	0	0
ORGANIZATION	Communication	Communication failure	1	0
		Missing information	1	0
	Organization	Maintenance failure	1	1
		Inadequate quality control	1	1
		Management problem	0	0
		Design failure	1	1

Continued on next page

			Expert Classification	HF Classifier
		Inadequate task allocation	1	1
		Social pressure	1	0
	Training	Insufficient skills	1	0
		Insufficient knowledge	1	0
	Ambient Conditions	Temperature	0	0
		Sound	0	0
		Humidity	0	0
		Illumination	0	0
		Other	0	0
		Adverse ambient conditions	0	0
	Working Conditions	Excessive demand	1	0
		Inadequate work place layout	0	0
		Inadequate team support	0	0
		Irregular working hours	0	0
			Sum of true positives	5
			Sum of true negativies	35
			Sum of false positives	0

Continued on next page

	Expert Classification	HF Classifier
	Sum of false negatives	13
	Accuracy	75%
	Precision	100%
	Recall	28%
	F1-Score	43%

Table 3.3: Virtual expert x expert classification for FPSO CDSM accident report

The *HF Classifier* identified the following factors from the FPSO CDSM report: the technological factor of inadequate procedure, and the organizational factors of maintenance failure, inadequate quality control, design failure and inadequate task allocation.

Figure 3.5 illustrates the performance of the *HF Classifier* on the FPSO CDSM report. It shows a strong overall result, largely driven by the high number of TNs. While the classifier made no FP errors and correctly flagged some relevant factors, the relatively low number of TPs and the presence of FNs suggest that its good accuracy is due to the dominance of irrelevant factors in the dataset rather than consistently accurate detection of contributing causes.

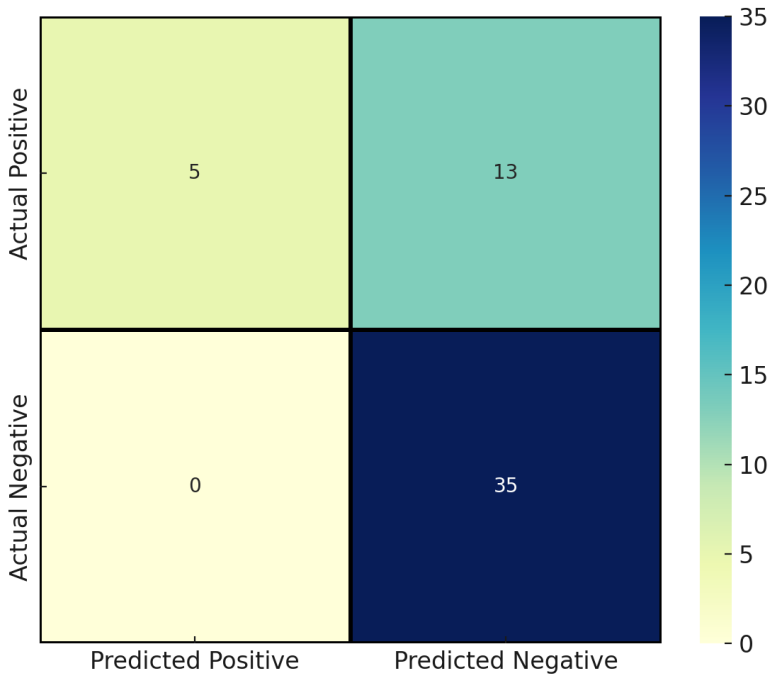

Figure 3.5: Confusion Matrix Heatmap - FPSO CDSM accident

The word cloud in Figure 3.6 shows that the text extracted from the full report had some frequent words that were not related to any accident cause of human factor. The words "*Brazilian*", "*agency*", "*biofuel*", and "*ssm*", are related to the name of the investigation body that was repeated at the footnote in every page – therefore, in future developments this should be automatically removed.

Figure 3.6: Word Cloud for the full FPSO CDSM accident report

3.4.3 Discussion and Suggestions

This work set out to explore the potential of automatically classifying accident reports using a trained machine, with the main goal of supporting human reliability data collection. The results suggest some utility in this approach, but also reveal notable limitations that warrant careful consideration. Most significantly, the classifier's overall performance, especially in terms of recall (46%), highlights the current challenges in reliably detecting all relevant human errors and PSFs. In safety-critical domains, the cost of missing such factors, and consequently failing to allocate mitigation resources appropriately, makes recall an especially important metric. While high precision ensures the categories identified are usually correct and useful for appropriate resource allocation, the model's limited recall means a number of critical items may go undetected. The performance and balance between precision and recall requires further optimization.

Although the testing set provides the performance metrics needed to benchmark the tool against previous studies, the case studies offer additional contextualized insights into how the model performs in practice. These case studies demonstrate the applicability of the model in two different industry sectors (aviation and Oil & Gas) and help to better understand how positive and neg-

ative classes impact the model's performance. The performance metrics from the case studies show slight variations compared to the testing set results, particularly in terms of precision and recall. Notably, in the Oil & Gas case study, the classifier achieved perfect precision (100%), but its low recall (28%) indicates it is far from comprehensively identifying critical factors. While the predictions made are reliable, the limited scope of detected factors could lead to significant omissions in risk management efforts.

Several factors contribute to the model's performance limitations. Accident reports are often written in varied narrative styles, differ in structure across organizations, and contain technical terminology or implicit references that pose challenges for standard NLP models. In addition, the dataset used for training is highly imbalanced: some contributing factors appear frequently, while others are rare. This skewed distribution makes it difficult for the model to learn to identify less common but still important PSFs.

The case study results indicate that the categories detected by the ML approach align with the 26 most significant contributing factors identified by Moura et al. 2017a. This finding suggests that the model could potentially be trained using only the more frequently occurring categories, which might lead to improved performance. Nevertheless, the tool was also able to detect an infrequent category, namely psychological stress, which appears in only 3% of the 238 investigation reports. Although this category was misclassified when compared to expert human classification, its identification demonstrates that the model is capable of recognizing less common but relevant PSFs.

Further testing was conducted by limiting the training to the 13 most frequent categories. The results showed no significant changes in overall performance, though there was a slight improvement of approximately 5% in precision, recall, and F1-score, with accuracy remaining largely stable. Based on these findings, the decision was made to continue investigating all categories. This approach supports the expansion of the dataset and helps reduce epistemic uncertainties related to rare combinations of human error and PSFs.

To improve the accuracy of predictions made by automated classifiers, several adjustments to the format of accident investigation reports are recommended. Reports should adopt a standardized structure, with consistent chapter titles and sequence. Text should not be embedded in image format, as this hinders machine readability. Furthermore, reports should be made publicly available online, if not in English, any other language is acceptable, given the continued advancement of translation tools. Less critical recommendations include ensuring chapter numbers are used consistently and only in the summary and body text, clearly distinguishing sections that

describe the normal characteristics of the installation, ideally placing these at the beginning or end of the report (for example, as an appendix), avoiding the inclusion of causal information within these descriptive sections, and refraining from repeating the name of the investigative body in the header or footer on every page.

3.5 Conclusions

The *HF Classifier* provides automatic classification of accident reports involving human error. This approach demonstrates potential for efficiently expanding existing human reliability databases (MATA-D) based on accident reports analyzed by a ML algorithm. It has the potential ability to support and serve as a check for classification tasks typically conducted by human experts. The objective is to accelerate the work of human experts, as reading full accident reports to classify them into a specific taxonomy is a time-consuming process that can take weeks, depending on the complexity of the event and the number of reports or inquiries available. While this work emphasizes the value of accelerating data processing and communication, it does not suggest that timeliness should override the careful analysis required for implementing safety measures.

The developed tool offers nearly real-time classification capabilities. These findings are of interest to risk assessors in any industry sector who need to learn more efficiently from past major accidents, as automated text analysis can help them expand their datasets.

The proposed tool can be easily adapted for use with other human reliability taxonomies or applied to components' reliability data, provided a labeled dataset is available along with the text sources.

While the tool holds promise for supporting human experts, it is important to acknowledge its current limitations. The classifier's performance metrics, particularly its recall (46%), highlight areas that need improvement to ensure comprehensive detection of human errors and PSFs. The trade-off between precision and recall must also be carefully managed, as undetected risks can undermine the effectiveness of automated classification. These limitations suggest that, at this stage, the tool is best used as a complementary aid to human expertise rather than as a standalone solution. However, the performance metrics achieved, when compared to other previous

studies, indicate that the automated text techniques chosen are satisfactory and relevant.

By training the tool with reports from diverse industry sectors and incorporating a wide range of human reliability categories, this study demonstrates the feasibility of cross-industry lesson learning. The classifier shows promise mapping information from different sectors onto a common human reliability classification scheme, enabling the integration of lessons learned across industries. However, its limited recall and sensitivity to less frequent categories highlight the need for expanding the dataset and refining the classification approach to reduce epistemic uncertainties.

In conclusion, while the *HF Classifier* showcases the potential for ML to improve empirical data collection for HRA, its current performance metrics reflect both strengths and limitations. The tool represents a step forward in leveraging automation for HFs analysis, but further work is required to enhance its recall, address variability across contexts, and ensure comprehensive and accurate classifications. These findings underscore the value of ML in supporting HRA while highlighting the importance of continued development to realize its full potential.

Chapter 4

Natural Language Processing for Human Reliability Analysis - Supporting Data Collection, Explainability and Expanding Applications

The simple models used in the development of the *HF Classifier* imposed limitations in its ability to capture context, nuanced meanings of text, handle the variations present in natural language These limitations impacted its performance metrics, particularly accuracy and recall. Consequently, this affects user confidence in the tool. Moreover, the ability to use the tool to directly expand the MATA-D is constrained, as users cannot rely on its classifications without extensive manual verification. This underscores the need for more advanced models that can better handle the complexities of natural language and understand context, to improve performance metrics and bolster user trust.

NLP is branch of AI concerned with enabling computers to understand, interpret, and respond to human language, with various applications such as speech recognition, machine translation, and chatbots (Chowdhary 2020). A key advancement in NLP is the development of LLMs, which are sophisticated algorithms trained on vast datasets of text. These models exhibit a remarkable ability to understand text due to their extensive training on diverse datasets. Employing deep learning techniques, they have demonstrated remarkable proficiency in generating coherent and

contextually relevant text. They are instrumental in powering complex tasks such as question-answering systems, text summarization, and language generation tasks.

To address the limitations and enhance the performance of the methodology employed by the *HF Classifier*, a popular LLM named BERT (Bidirectional Encoder Representations from Transformers), (Devlin et al. 2018), has been selected as the foundation for the newly developed classification tool, termed the *Second-Generation Virtual HF Classifier* (*HF Classifier 2.0*). This tool retains the core idea of the original approach, wherein a classification layer is fine-tuned using the labeled accident reports from the MATA-D, thereby leveraging BERT's advanced understanding of context and language nuances to improve classification accuracy.

In addition to classification, the development of a secondary tool that offers a summarization capability leveraging BART (Bidirectional and Auto-Regressive Transformers) (Lewis et al. 2019) to be known as, *Human-Centric Summarizer*, is presented (Johnson et al. 2023b). This feature is designed to distill lengthy accident reports into concise, informative summaries focusing particularly on the human role in each accident. The leveraging of BART allows abstractive summarizations, that are not simple truncations of the original text but rather overviews that highlight the key elements related to human factors and system interactions.

The tool's summarization functionality serves multiple purposes. The developed tool aids researchers and safety professionals in grasping each report, as well as any models based on the incident, without delving into the pages of detailed reports, which is vital for accurate decision-making and effective safety management.

Additionally, the summarization tool aids in understanding the outputs of the classification tool, boosting confidence in the results and related analysis, and facilitating updates to the MATA-D. The generated summaries offer a reference to the key aspects of each incident, making it easier to analyze and review the PSF classifications. By capturing critical moments and situations that may have led to human errors and accidents, the summarization tool serves as an essential explainability layer for the classification tool. It provides the evidence behind each report's classification, highlighting the human role in incidents and uncovering underlying causes of errors. This transparency enhances the analysis's comprehensibility and supports more efficient data use and processing in HRA.

By enhancing the use and applications of NLP in the field of HRA, these tools underscore the potential of advanced computational techniques to transform the way data collection and processing is conducted. Ultimately contributing to more data-driven decisions and the development of safer operational practices across various complex industries.

Beyond facilitating the expansion of the MATA-D, the developed tools offer significant real-world applications. However, they are intended to support, not replace, expert analysis, serving as valuable aids in focusing initial evaluations, helping identify patterns, and providing direction for deeper investigations. This is especially useful when reviewing several older incidents to inform preliminary ideas and approaches.

The classification tool enables safety professionals to efficiently analyze accident reports, pinpointing organizational, technological, and individual contributors to incidents. In some cases, it has been proven capable of identifying factors that experts have overlooked during manual analysis. The summarizer complements such insights by providing additional human focused evidence from the original report. While the tools do not eliminate the need for expert input, the tools can enhance the accuracy of identifying potential risks, providing a reliable starting point for further evaluation.

Additionally, these tools support compliance with safety regulations by facilitating the systematic documentation and analysis of relevant human error data and accident insights. This structured approach promotes continuous improvement in safety protocols and operational procedures, ultimately contributing to the creation of safer industrial environments. Safety changes, by their nature, take time and careful consideration. These tools are not a shortcut to immediate solutions but are intended to streamline the initial steps of analysis, helping professionals focus their efforts and accelerate the review process.

This chapter will first focus on the development of the *HF Classifier 2.0*, introducing BERT before discussing its implementation and performance. The next section will move onto the *Human-Centric Summarizer*, where the development of the extractive and abstractive (leveraging BART) summarization algorithm is discussed and evaluated. Having introduced the tools the next section explores some case studies that demonstrate the usability of the developed tools. This is followed by some discussion about the limitations and scope for future development of the tools, before the chapter closes with a brief conclusion. Discussion of past related studies and different approaches/methodologies is given in Secion 2.2.

4.1 Dataset

As with the development of the first-generation *HF Classifier*, the second-generation tool also uses the MATA-D as its target training and validation data.

4.2 Classification Tool

Building on the predecessor, insights from previous studies, and mainly the advancements in NLP, the *HF Classifier 2.0*, a new classification tool, has been developed to overcome the limitations of its predecessor and deliver superior performance. Central to this advancement is BERT, a LLM whose functionality and capabilities are discussed. In this section the development of the tool, how BERT's technology is leveraged and fine-tuned to meet our specific objectives is detailed. The discussion concludes with a comparative analysis of the performance of both generations, highlighting the enhancements achieved with the integration of BERT.

4.2.1 BERT

BERT represented a breakthrough in the field of NLP (Devlin et al. 2018). BERT employs the Transformer architecture, primarily its encoder component, to better understand contextual relationships between words in a text. Unlike traditional models that process words sequentially (either left-to-right or right-to-left), BERT examines text bidirectionally, providing a deeper understanding of language context and semantics. The core innovation of BERT lies in its use of the Transformer's attention mechanism, which allows the model to weigh the importance of different words relative to others in a sentence, regardless of their position (Lin et al. 2022). This means it can simultaneously consider the context of a word from both the left and the right sides of it within a sentence, effectively understanding language in a way that mirrors human comprehension more closely.

BERT is pre-trained on a massive corpus, including the BooksCorpus and English Wikipedia, which encompasses over 3 billion words (Devlin et al. 2018). This pre-training involves learning by masking some words in the text and predicting them based only on their context, a method known as Masked Language Modeling (MLM) (Salazar et al. 2020). This unsupervised learning allows BERT to develop a sophisticated sense of language structure and word relationships before it's ever used for specific tasks.

BERT is used for classification by adding a classification layer on top of the pre-trained BERT model, which is fine-tuned using a labeled dataset specific to the task at hand. Here, the final hidden state corresponding to the classification token is used to predict the class labels. This approach leverages BERT's pre-trained contextual mappings, significantly enhancing its efficiency and accuracy in class-specific tasks. When compared to traditional ML classifiers, BERT offers superior performance by effectively capturing complex language nuances that are often missed by more conventional methods (Garrido-Merchan and Gonzalez-Carvajal 2023).

BERT utilizes WordPiece tokenization, which breaks words down into meaningful sub-units, allowing the model to effectively handle out-of-vocabulary (OOV) words by processing them as sequences of sub-words. This capability is particularly useful in dealing with specialized vocabularies and unusual terms, which are common in domain-specific texts (Sennrich et al. 2015). The architecture comprises multiple layers of attention and feed-forward networks, which allow it to capture and analyze complex relationships and dependencies in text. Each layer computes representations of the input data with increasing levels of abstraction and complexity, which enhances BERT's ability to classify texts with nuanced meanings or intricate structures effectively (Wang et al. 2024).

In the context of accident report analysis, the ability of BERT to handle diverse and complex text structures is particularly valuable. Given the variable formats and specialized vocabularies of industrial accident reports, BERT's robust pre-training allows it to adapt to this domain with relatively little additional training. However, it's noteworthy that BERT is only capable of handling sequences up to a 512 tokens in length, the model requires tailored preprocessing steps for all documents to effectively manage longer texts typical of accident reports. This preprocessing involves segmenting texts and removing less informative parts to reduce the amount of segmenting necessary due to the token limit, ensuring that the most relevant information is

retained for classification.

This model's capacity to integrate and understand the contextual relationships in text makes it an advisable tool for the automated classification and analysis of extensive accident reports, paving the way for more insightful and actionable safety analytics.

4.2.2 Second-Generation Virtual HF Classifier

The *HF Classifier 2.0* follows a similar initial process and logic to that of the first-generation tool.

First, all incoming accident reports are initially processed and converted to text files. If the document is not machine-readable, OCR software is used to convert them into text files, ensuring that the data input into the system is standardized and accessible for further processing. These text files are then scanned for specific sections of interest, such as "*recommendations*" and "*lessons learned*". If found these specific sections are extracted and used in place of the complete report, if not found the entire report is used. Utilizing the WordPiece tokenization scheme, the text data is broken down into manageable pieces. This tokenization, identical to the pre-training scheme of BERT, allows for effective handling of various word forms and enhances the model's ability to process and understand complex vocabulary.

The tokenized texts are combined with predefined labels from the MATA-D dataset to create the training data. At the core of the *HF Classifier 2.0* is a pre-trained BERT model, imported from the Hugging Face Transformers library (Face 2023b). To adapt BERT for the classification task, a fully connected classification layer is appended to its final hidden layer. This layer uses the output embedding of the special `[CLS]` token—which captures the overall meaning of the input sequence—as input, and produces a probability score for each class. The entire model is trained using stochastic gradient descent with a binary cross-entropy loss function. During training, hyperparameters are fine-tuned based on the model's performance.

To address BERT's limitation with processing long text sequences (maximum 512 tokens), documents exceeding this threshold are divided into smaller chunks of 500 tokens. During both training and inference, each chunk is processed independently through the model. Rather than assigning labels to each chunk individually, the model outputs (logits) from all chunks of a document are averaged to form a single, aggregated representation. This aggregated output is then

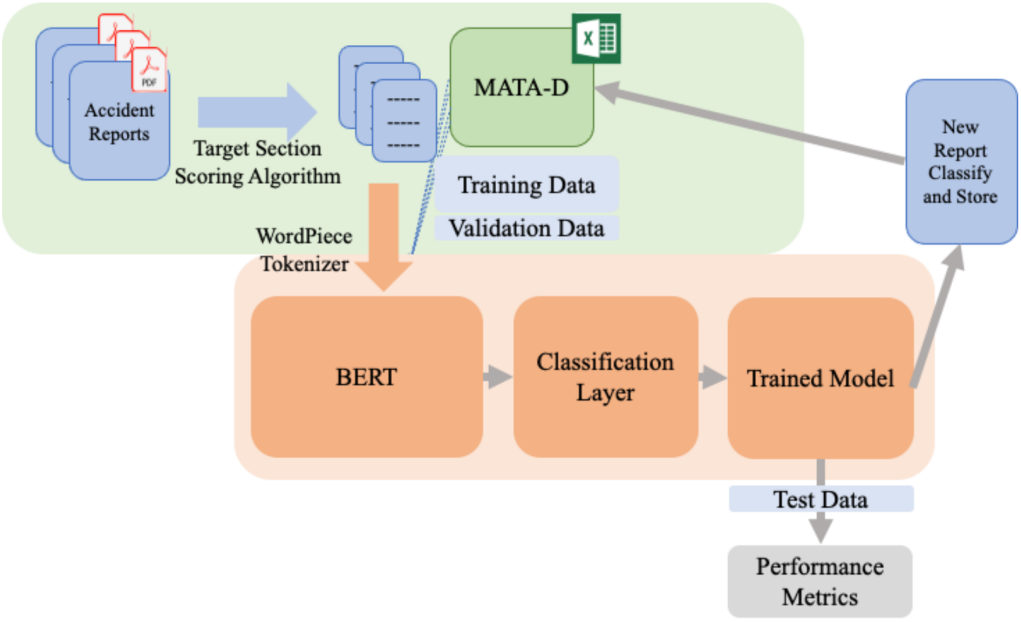

Figure 4.1: Simplified Workflow of the HF Classifier 2.0

used to compute the final classification, ensuring that the model learns from the entire extracted text context and reducing label noise introduced by chunk-level supervision.

The trained model classifies the accident reports, identifying the factors involved in each incident. The output is a binary array that represents the presence or absence of each of the 53 factors outlined in the MATA-D. This facilitates easy updating and expansion of the dataset with new incidents. Post-training, the model is saved for future use. This ensures that new reports can be classified efficiently using the established protocol. The saved model streamlines the process for future accident report evaluations, maintaining consistency and accuracy in classifications.

The process from data input through to output generation is outlined in Figure 4.1. Further details regarding the tools background can be found in Section 6.1 and Section A.2.

4.2.3 Classification Tool Performance

This tool is evaluated using the same metrics used to evaluate the first-generation tool in Section 3.3.

- Accuracy: The proportion of true results (both TPs and TNs) among the total number of cases examined.
- Precision: The ratio of TPs to the sum of true and FPs, indicating the correctness of positive predictions.
- Recall: The ratio of TPs to the sum of TPs and FNs, reflecting the ability to identify all relevant instances.
- F1-Score: The harmonic mean of precision and recall, providing a balance between the two when their values vary widely.

As with the first-generation tool to assess the effectiveness of the second-generation classifier, the binary classifications from the MATA-D dataset are used as the target classes. The performance metrics obtained by the first-generation serve as the benchmark for this version's performance. Given the large number of TNs typically observed in MATA-D, where an average of 46 out of 53 categories are identified as negatives across all reports, the dataset could be considered imbalanced. In such scenarios, the F1-score, which balances precision and recall, is often more reflective of the model's performance than accuracy alone.The performance metrics for the *HF Classifier 2.0* are given in Table 4.1.

Metric	**HF Classifier 2.0**
Accuracy	91%
Precision	82%
Recall	74%
F1-Score	78%

Table 4.1: HF Classifier 2.0 Performance Metrics

The model, trained with all available reports in the MATA-D dataset, achieved an accuracy of 91%, precision of 82%, recall of 74 and an F1-score of 78%. These results represent a substantial improvement over the benchmark metrics from the previous version, which reported 86% accuracy, 60% precision, 46% recall, and 52% F1-score.

The enhanced performance demonstrates the tool's capability to accurately classify binary classes within the MATA-D dataset. The improvements in precision and recall indicate a stronger ability to manage the complexities of the dataset, including imbalanced classes, while maintaining reliability. Additionally, with the capacity to process reports at a rate of approximately one

minute per report, the tool offers improved efficiency compared to its manual assessment, along with significantly greater confidence in its results compared to its predecessor.

However, while the tool marks a meaningful advancement, it is important to recognize that it is not without limitations. Its recall of 74% and F1-score of 78% indicate there is still room for improvement in consistently capturing all relevant classifications, particularly in highly nuanced or edge cases. These performance metrics, while strong, reflect the inherent challenges of applying automated classification to complex, context-rich data.

The tool is designed primarily to support the collection and structuring of human reliability data from accident reports, helping to expand empirical databases used for safety analysis. Within risk analysis, it does not aim to replace human expertise but rather to enhance it by streamlining the initial stages, offering a solid foundation for further investigation. It serves as a powerful aid for safety professionals, helping focus their efforts and enabling efficient evaluations. At the same time, expert validation remains essential to address the subtleties and contextual nuances that are critical to understanding human factors and making informed safety recommendations.

4.2.4 Classification of Shorter Reports

One of the limitations of the predecessor model was its inability to effectively handle small reports. The SVM approach used in the first generation performs better with larger documents (Morais et al. 2022a). Testing of the first-generation tool on shorter reports, such as near miss summaries, confirmed the expected decline in performance. This limitation has been highlighted by industry stakeholders during demonstrations and testing, particularly in relation to shorter reports describing minor events. These reports often contain limited textual content, which reduces the the inclusion of keywords that the first-generation model relies on to identify relevant PSFs. As a result, the tool often returns no classifications or identifies irrelevant factors, limiting its usefulness for analyzing such cases.

In contrast, the performance of the second-generation classifier is not heavily dependent on the length of the accident report. To evaluate this, ten reports from the test set were selected: five reports ranging from 5 to 20 pages and five over 100 pages. The performance metrics for each group showed only slight variation, with the shorter reports experiencing a minor (<5%) decrease in accuracy, recall, and precision. This suggests that while there is a slight decrease in

performance due to the reduced context available in shorter reports, the performance remains satisfactory, supporting the tool's application to reports of various lengths.

Longer texts pose the challenge of requiring more splitting, which can lead to the fragmentation of context. This fragmentation makes it challenging for BERT to understand the full meaning if critical information is split across chunks. However, the additional context in longer texts can help BERT make more informed decisions. Conversely, shorter texts require less chunking, reducing the potential for information loss or misclassifications that might occur when the text is divided arbitrarily. This is an advantage of shorter texts that partly compensates for the reduced context.

Overall, the main factor influencing the tool's performance is the quality of the report. Shorter reports should still contain the necessary details to evaluate the accident and the PSFs. Extremely short reports, less than two pages long, remain a challenge for the tool as they are unlikely to provide all the necessary insights. This would be a challenge and limitation for any data-driven tool, and is an unrealistic expectation that could only be resolved with increased data from more comprehensive reporting. However, additional pre-training on domain-specific context could help improve the model's ability to extract meaningful information from shorter or less detailed reports.

Further details of the technical development of the *HF Classifer* is presented in Chapter 6, Section 6.1.1.

4.3 Summarization Tool

Following the development and demonstration of the capabilities of the advanced classification tool, another specialized tool added to our technological suite: the *Human-Centric Summarizer* is introduced. This tool utilizes BART to condense lengthy accident reports into concise summaries. Prioritizing the human elements of each incident, it captures complex interactions and significant details beyond simple truncation. These sections explore the functionality and strategic advancements of BART within this application. By combining BART's capabilities with an initial extractive summarization phase how the tool effectively produces comprehensive summaries is demonstrated.

Evaluating the performance of such summarizations presents unique challenges, therefore, multiple assessment methods have been implemented. This approach provides a thorough evaluation of the tool's effectiveness, ensuring that the summaries are both informative and reflective of the original reports.

4.3.1 BART

In the development of the *Human-Centric Summarizer*, the BART model was chosen for the abstractive summarization step. The accurate summarization of complex accident reports can support decision-making and analysis, however traditional extractive summarization methods often fall short in capturing the nuanced essence of such texts. However, BART, a sequence-to-sequence model renowned for its effectiveness in generating coherent and contextually relevant summaries, promises far greater understanding and interpretability.

BART integrates the Transformer architecture in a sequence-to-sequence framework (Lewis et al. 2019). Unlike BERT, which is primarily an encoder model used for understanding text, BART extends this with a decoder module, enabling it to generate new text based on the understood context. BART is pre-trained using a combination of denoising auto-encoding and sequence-to-sequence tasks. The denoising aspect involves corrupting the original text (via masking, deletion, or order shuffling) and training the model to reconstruct it, which equips BART with robust paraphrasing capabilities crucial for summarization (Lewis et al. 2019).

BART excels in abstractive summarization, where it generates paraphrased text that summarizes key points while possibly omitting redundant information. The seq2seq framework of BART, combined with its training on denoising and paraphrasing tasks, uniquely positions it for summarization. It can produce summaries that are not only context-aware but also maintain the essence and continuity of the original text (Venkataramana et al. 2022).

BART's effectiveness in summarization is underscored by its ability to handle complex document structures and generate summaries that are accurate and reflective of the original text's intent and factual content.

4.3.2 Summarization Tool Development

The summarization process used in the *Human Centric Summarizer* is highly effective for analyzing accident reports. It excels at extracting key sections and generating concise, informative summaries. This provides interested parties with further context and evidence, beyond the list of identified PSFs, without the significant task that is reading the entire report. The following steps and complete process is illustrated in Figure 4.2.

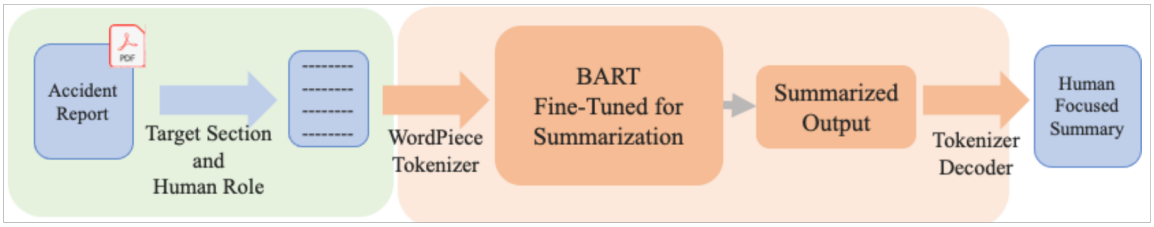

Figure 4.2: Simplified Workflow of the Human Centric Summarizer

The initial phase involves data cleaning to ensure the accuracy and relevance of the information fed into the model. This step includes the extraction of critical sections within the reports, specifically targeting 'recommendations' and 'lessons learned'.

Additionally, the model identifies and extracts sentences that contain pronouns or human-related nouns (such as "*user*", "*operator*", "*manager*") from the entire text. This capability is implemented using the Python package NLTK (Natural Language Toolkit), which enables NLP tasks (Bird et al. 2009). Our custom pipeline leverages these tools to accurately detect and extract sentences with pronouns and human-related nouns. This approach provides a flexible and robust solution capable of handling diverse and complex text data, enabling the precise extraction of sections detailing the human role in the incident, which are essential for generating a human-centered summary. These extracted parts are then aggregated and stored with the target sections for further processing.

If the word count of the combined text exceeds 1024 words, it is segmented into smaller sections to facilitate manageable summarization tasks. This segmentation ensures that the model can handle large texts efficiently. Each segment is summarized and then combined, if exceeding the word count again it automatically iterates through the tool until the entire summary is below

the target threshold (500 words).

The text is tokenized using the pre-trained BART tokenizer from the Hugging Face Transformers library (Face 2023a). This process converts the raw text into a format that the BART model can interpret, preparing it for the summarization phase. BART processes the tokenized text to generate multiple summary candidates, utilizing a controlled beam search technique. This method allows the model to explore several possible summaries simultaneously, ultimately selecting the summary that achieves the highest score based on language model probabilities (Lewis et al. 2019). Finally, the selected summary is converted back into human-readable text using the BART tokenizer's decoding method. This text represents the distilled essence of the original report, emphasizing crucial insights and actionable information.

Additional details of the *Human-Centric Summarizer* is presented in Chapter 6, Section 6.2.

4.3.3 Summarization Tool Performance

When assessing the performance of our summarization tool, some key factors need to be considered,

- Quality: The accuracy and readability of the summary.
- Relevance: The pertinence of the summary content to the original text.
- Length: The conciseness of the summary relative to the original content.
- Domain-Specific Performance: The tool's effectiveness in handling jargon and specifics of the domain.

There are several evaluation metrics that can also be used such as, ROUGE (Recall-Oriented Understudy for Gisting Evaluation), BLEU (Bilingual Evaluation Understudy), and METEOR (Metric for Evaluation of Translation with Explicit ORdering) to assess the quality of the generated summaries (Lin 2004, Papineni et al. 2002, Lavie and Agarwal 2007). ROUGE measures the overlap between the n-grams of the generated summary and reference text. ROUGE variants include ROUGE-N for n-gram overlap, ROUGE-L for sequence matching, ROUGE-S for capturing long-distance word pairs, and ROUGE-W for prioritizing longer sequences. ROUGE metrics generally evaluate two aspects. These are Recall, the fraction of the words from the

reference summaries that appear in the generated summary, and Precision, the fraction of the words in the generated summary that appear in the reference summaries (**line**). Similarly, to the classification performance metric, F1-score for the ROUGE measure, which combines recall and precision into a single metric using a harmonic mean, is reported.

While ROUGE can be useful, it has limitations as it does not account for semantic equivalence (different words with the same meaning), a significant drawback when using an abstractive summarization. The metric also does not evaluate the generated text for coherency or readability. Despite this ROUGE is still used in multiple studies and serves as a benchmark metric for comparison in this study (Maples 2017).

Given these challenges, human judgment is crucial for a thorough assessment of summary quality. Therefore, the tool's performance is not only evaluated through automated metrics but also by manually comparing the generated summaries with sections extracted from the original accident reports. This manual assessment focuses specifically on the readability and conciseness of the summaries.

Additionally, the co-development of the classification tool allows for both the summaries and the original texts to be used as inputs. This facilitates the calculation and comparison of classification performance metrics, providing insights into the information retained or lost during the summarization process. By integrating these various approaches, a comprehensive assessment of the *Human Centric Summarizer* can be achieved.

For this study, 30 reports were randomly selected to evaluate the performance of the summarization tool.

4.3.3.1 ROUGE Metrics

To calculate the ROUGE metrics, the final summarizations were compared to the text extracted directly from the original report, more specifically the critical sections "recommendations" and "lessons learned", and the sentences that contain pronouns or human-related nouns (such as "*user*", "*operator*", "*manager*"). This extracted text had an average length of approximately 3000 words. The F1-score for ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-S are given in Table 4.2.

ROUGE Metric	F1-Score
ROUGE-1	51%
ROUGE-2	40%
ROUGE-L	45%
ROUGE-S	32%

Table 4.2: ROUGE Performance Metrics for Human Centric Summarizer

The F1-score of 51% for ROUGE-1 suggests a moderate performance in terms of capturing relevant keywords from the original text in the summary. This metric is critical as it ensures that the summarization includes most of the crucial terms that appear in the reference text most frequently, thereby maintaining the core factual content. Within the reports there are industry and situation specific terms that are essential for understanding the context and implications of the report, this score indicates that the summarization tool it limited but has the ability to retain key information. This is particularly useful in scenarios where stakeholders need to understand the essential details without delving into the full document. The ROUGE-2 score at 40% is lower than ROUGE-1, which is expected because it measures the more challenging task of capturing consecutive word pairs (bigrams). This score is moderate and suggests that while the tool is reasonably effective at maintaining some of the original meaning, there is room for improvement in capturing more complex relationships within the text. This decrease reflects the inherent difficulty in compressing detailed content into a shorter form while preserving specific two-word phrases that may be critical for conveying nuanced meanings. In this context, losing some of these nuances can mean missing out on subtler context or technical descriptions, which could be significant for a full understanding but might be less critical for a high-level overview. With an F1-score of 45%, the ROUGE-L metric shows that the summarization tool has an adequate ability to maintain the order and structure of the content. This score reflects how well the summary preserves the longest common subsequences (LCS) within the reference, indicating the tool's effectiveness at capturing the flow and narrative structure of the original text. Although not as high as ROUGE-1, this level of performance is sufficient for ensuring that the overall coherence and logical sequence of ideas are maintained, which is vital for summaries intended to provide actionable insights.

The lowest score of 32% in ROUGE-S indicates a challenge in capturing skip-bigrams, which involve non-consecutive but contextually related word pairs. This metric highlights a limitation

of the tool in preserving the broader contextual connections within the text, which can be crucial for fully understanding complex interactions and sequences in accident scenarios. The relatively lower score suggests that while the tool captures direct content effectively, it struggles more with interconnected content spread across the original document. This aspect is particularly relevant where understanding the broader implications of specific actions or events within an accident report is necessary.

The ROUGE scores obtained from our tool demonstrate a reasonable level of accuracy in summarizing complex, human-centered content from lengthy accident reports. While the F1-scores are not near-perfect, it is crucial to consider the inherent complexity of summarizing detailed accident reports, which often contain nuanced information that is challenging to condense effectively. These scores are also comparable to those achieved in a similar study where BART was used to summarize long financial reports, with F1-scores of 45% for ROUGE-1, 32% for ROUGE-2, 44% for ROUGE-L and 37% for ROUGE-S (Zmandar et al. 2023).

The primary goal of our summarization tool is to highlight critical human-related insights rather than to replicate the original text verbatim. Thus, a lower score in some ROUGE metrics does not necessarily indicate poor performance but rather reflects the summarization tool's focus on extracting essential meanings rather than exact phrasings. However, the scores also highlight areas where the tool may struggle, such as preserving exact word sequences and the structure of the original text (as evidenced by the lower scores in ROUGE-2 and ROUGE-S).

The ROUGE metrics provide valuable insights into our overall assessment of the performance of our summarization tool, underscoring its effectiveness in capturing key content while also pointing out limitations in handling complex narrative structures. The utility of the tool, particularly in contexts where efficient assimilation of critical information is needed, such as in safety protocol development or accident response planning, remains significant. This is especially true when a multidisciplinary team is risk assessing a system and wants to find evidence of similar scenarios in accident reports.

4.3.3.2 Classification Tool Comparison

Additionally, the impact of summarization on the performance of the *HF Classifier 2.0*, specifically in terms of PSF classification, is evaluated. This comparison, outlined in Figure 4.3, provides insight into any potential information loss resulting from the summarization process.

Metric	Original Reports	Summarized Reports
Accuracy	91%	79%
Precision	82%	70%
Recall	74%	61%
F1-Score	78%	65%

Table 4.3: Original vs. Summarized Report - Classification Performance Comparison

These results show a decrease in all performance metrics when using summarized reports compared to the original reports. This reduction can be expected due to the compression of the content, where essential details might be lost or altered. Despite this, the summarized reports still maintain a relatively high performance, suggesting that the critical aspects of human roles and influencing factors in accidents are preserved effectively in the summaries.

One significant factor contributing to the decline in classification performance is the exclusion of more detailed, technical sections in the text extraction process of the tool. Our tool is designed to focus primarily on human-related aspects of the accident reports, such as actions, decisions, and roles. Consequently, it tends to omit sections that discuss the technical, intricate mechanical details, or complex system. These sections often contain specialized terms and information that are critical for a fully nuanced understanding of the accident causes, particularly the technological PSFs.

The current performance of our classification tool on summarized content demonstrates the effectiveness of the summarization in retaining critical information about human roles. However, the decrease in classification metrics suggest possible need for the development of another summarization tool, that can better accommodate the technical complexities inherent in accident reports, ensuring that all essential aspects are represented accurately in the summaries. This combined approach will enhance the utility for a broader range of applications, particularly in

detailed technical assessments and safety compliance monitoring. Integrating insights from historical incidents refines any analysis, ensuring that lessons learned from past accidents inform the risk assessment processes for new projects. This incorporation of past knowledge strengthens our overall approach to safety and reliability.

4.3.3.3 Manual Assessment

The final verification of the summarizer's performance is a manual assessment, achieved by manually reading and comparing the reports and respective summaries of ten accident reports. From this, it can be said that the tool generates a summary that accurately captures the main points and key information of the identified sections in a concise and readable format in an efficient manner. However, in an attempt to fit to the token limit, the tool can sometimes combine human actions in a way that misrepresents what is stated in the original accident report, missing key information of contextual significance. An instance of this is, "*...The operator shut down the pump when a leak was detected in the pipeline... The technician repaired the leak and the pump was restarted...*" was summarized to "*The operator shut down the pump, the technician restarted the pump.*" Here any information regarding the leak and repair was excluded. Such omissions highlight the challenges of maintaining the contextual integrity of accident reports.

To enhance the accuracy and relevance of the summaries, especially concerning domain-specific terminology and critical incident details, the development of a fine-tuned layer for the summarization tool could be suggested. This layer would utilize summaries written by human experts as a training base. Although this approach would require a significant time investment in the initial stages, the long-term benefits shown in similar applications that such investments in domain-tailored training significantly enhance tool performance, improving accuracy and the contextual relevance of automated summaries (Yadav et al. 2023).

The *Human-Centric Summarizer*, despite some observed limitations in detail retention as shown by the ROUGE metrics and classification tool performance, has demonstrated substantial potential in processing extensive accident reports both swiftly and effectively. The ability to extract

and condense crucial human-centered information from detailed and lengthy reports into a more manageable form presents significant benefits. It facilitates decision-making, enhances the accessibility of essential data, and provides evidence for the PSFs identified by the classification tool.

4.4 Case Studies

Having explored the development and performance of the tools, some practical case studies are presented that underscore their utility. These three case studies demonstrate the tool's efficacy in analyzing incidents, extracting critical insights, and enhancing incident investigation processes. Through the lens of two distinct accidents and one SOP, how the developed tools aid in understanding the underlying causes of incidents and provide important context is demonstrated. The case studies provide an example of how the tools outputs can be used to inform proactive risk mitigation strategies and highlight the most influential PSFs in a procedure to aid assessors of HRA.

4.4.1 First Case Study - FPSO CDSM 2015 Incident

The first case study revisits the accident report for the FPSO CDSM incident, discussed in Section 3.4.2 (Vinnem 2018). This report has been classified by an expert and the first-generation tool, providing an opportunity to compare the outcomes generated by different generations of the classification tool.

Table 4.4 shows the comparison between the human factors classifications obtained by the human expert and both versions of the classifier tools.

				Expert	First Gen	Second Gen
HUMAN	Action	Execution (Error Modes)	Wrong Time	0	0	0
			Wrong Type	0	0	0
			Wrong Object	0	0	0
			Wrong Place	1	0	1
	Specific Cognitive Functions	Observation	Observation Missed	1	0	0
			False Observation	0	0	0
			Wrong Identification	0	0	0
		Interpretation	Faulty diagnosis	1	0	1
			Wrong reasoning	1	0	1
			Decision error	0	0	0
			Delayed interpretation	0	0	0
			Incorrect prediction	0	0	0
		Planning	Inadequate plan	1	0	0
			Priority error	0	0	0
	Temporary Person Related Functions		Memory failure	0	0	1
			Fear	0	0	0
			Distraction	0	0	0
			Fatigue	0	0	0
			Performance Variability	0	0	0

Continued on next page

			Expert	First Gen	Second Gen
		Inattention	0	0	1
		Physiological stress	0	0	0
		Psychological stress	0	0	0
	Permanent Person Related Functions	Functional impairment	0	0	0
		Cognitive style	0	0	0
		Cognitive bias	1	0	1
TECHNOLOGY	Equipment	Equipment failure	0	0	0
		Software fault	0	0	0
	Procedures	Inadequate procedure	1	1	0
	Temporary Interface	Access limitations	0	0	0
		Ambiguous information	0	0	0
		Incomplete information	1	0	1
	Permanent Interface	Access problems	0	0	0
		Mislabeling	0	0	0
ORGANIZATION	Communication	Communication failure	1	0	1
		Missing information	1	0	1
	Organization	Maintenance failure	1	1	1
		Inadequate quality control	1	1	1
		Management problem	0	0	0
		Design failure	1	1	1

Continued on next page

			Expert	First Gen	Second Gen
		Inadequate task allocation	1	1	1
		Social pressure	1	0	0
	Training	Insufficient skills	1	0	1
		Insufficient knowledge	1	0	1
	Ambient Conditions	Temperature	0	0	0
		Sound	0	0	0
		Humidity	0	0	0
		Illumination	0	0	0
		Other	0	0	0
		Adverse ambient conditions	0	0	0
	Working Conditions	Excessive demand	1	0	1
		Inadequate work place layout	0	0	0
		Inadequate team support	0	0	0
		Irregular working hours	0	0	0
		Sum of true positives		5	14
		Sum of true negativies		35	33
		Sum of false positives		0	2
		Sum of false negatives		13	4
		Accuracy		75%	89%

Continued on next page

	Expert	First Gen	Second Gen
	Precision	100%	88%
	Recall	28%	78%
	F1-Score	43%	83%

Table 4.4: Case Study - FPSO CSDM, HF Classifier Comparison

Both the SVM and BERT methods demonstrate high precision and accuracy. However, the BERT method shows a significant improvement in recall and, consequently, in the F1-score. Recall can be considered the most crucial metric for a human factors classifier, given the severe implications of undetected human errors or PSFs, that would remain unaddressed, with no resource allocated to its risk reduction, can have more severe consequences than false alarms. On the other hand, precision is also important when allocating resources, as it ensures that the identified factors are mostly accurate, preventing wasteful investments. The F1-score, which balances precision and recall, thus emerges as the most valuable metric. The marked enhancement in the F1-score with the BERT method, as evidenced by the performance metrics Table 4.1 and illustrated in this case study, support the development of the BERT-based tool.

The improvement in the tools ability to correctly identify present factors (TPs), is highlighted in Figure 4.3, especially when compared with Figure 3.5.

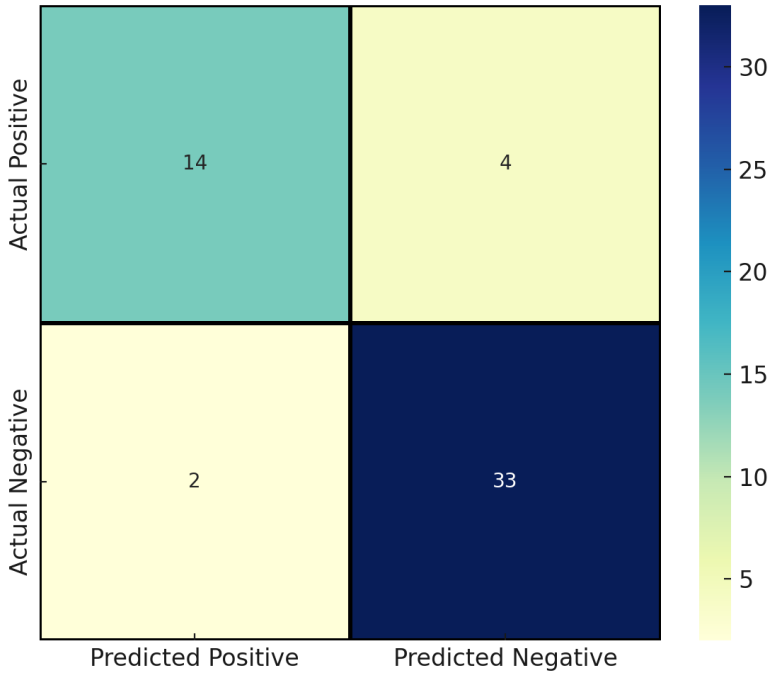

Figure 4.3: Confusion Matrix Heatmap - FPSO CSDM acident (HF Classifier 2.0)

In this incident within the oil & gas sector, the BERT tool identified several human failures and PSFs. The human failures included: human errors of execution such as wrong place, interpretation with faulty diagnosis and wrong reasoning, and permanent person related functions due to cognitive bias. Technological factors included issues like incomplete information related to temporary interfaces. Organizational factors encompassed communication failures, missing information, maintenance lapses, inadequate quality control, design flaws, improper task allocation, insufficient skills and knowledge, and excessive demands related to working conditions. Additionally, the tool identified two personal factors related to temporary functions, memory failures and inattention, not identified by the expert assessment. These findings do not necessarily indicate mistakes by the tool but rather highlight areas where further investigation might be necessary, challenging the assumption of a flawless manual assessment.

For the summarization of this accident report the metrics used in the evaluation of the tool can again be calculated, namely the ROUGE metrics and the classification tool comparison, these are given in Table 4.5 and Table 4.6.

ROUGE Metric	F1-Score
ROUGE-1	40%
ROUGE-2	30%
ROUGE-L	33%
ROUGE-S	25%

Table 4.5: Case Study - FPSO CDSM, Summarized Report ROUGE Metrics

Metric	Original Reports	Summarized Reports
Accuracy	89%	79%
Precision	88%	65%
Recall	78%	58%
F1-Score	83%	61%

Table 4.6: Case Study - FPSO CDSM, Original vs Summarized Classification Performance

The performance of the summarizer tool on this report was found to be lower compared to in Section 4.3.3. The report itself is extensive, spanning 380 pages and containing approximately 65,000 words. During the text extraction phase, sections totaling 8,000 words were identified for summarization, which were then condensed to just 500 words.

This may be a cause of the lower performance as this particular report is significantly longer than the majority of reports tested in the evaluation section, meaning more vocabulary and context has to be excluded to meet the word cap of the summary. Despite this limitation, a manual assessment revealed that the summarized report remained readable and provided a broad overview of the incident. The summarization tool provides evidence to explain the identification of many of the factors by the classification tool. For example, the factor of "Incomplete Information" is illustrated by the statement, "*They lacked access to information necessary for executing the jobs that should be included in the service ticket report for their relevance...*". "Communication Failure" is highlighted with the observation of "*...unsuitable communication between shifts....*" Furthermore, "Insufficient Skills" and "Insufficient Knowledge" are demonstrated in the summary by "*the operational practice did not include all phases, only the starting and stopping, and it was not in the guide.*"

However, the summarization process struggled to capture factors concerning control room design, equipment and maintenance, and how these might have contributed to errors. For instance, while the text mentions operational and technical issues, specifically the degradation of systems and

equipment and inadequate emergency response systems, these aspects were not explicitly detailed in the summary. This omission can be attributed in part to the initial text extraction phase, which prioritizes human actions over more technical sections. Given this example, it may be beneficial to adjust the word count cap for summarization, especially for comprehensive documents like this report, to ensure a more complete representation of the content, including both human and technical factors.

4.4.2 Second Case Study - Firefighting System Incident

The second accident case study, examines a report regarding an incident involving the spurious activation of the carbon dioxide firefighting system during maintenance work on the diesel generators, trapping two employees in the room, leading to one fatality. This document was originally in Portuguese and was translated into English using the free document translation option available on Google Translate before being input into the tool. This report contained a "*Human Factors analysis*" section that allows us to compare the tools' classification with the assessment of a different HFs expert. This section was removed before the report was uploaded to simulate the application of the tool instead of a HFs expert. The results are presented in Table 4.7.

				Expert Classification	**HF Classifier 2.0**
HUMAN	Action	Execution (Error Modes)	Wrong Time	0	0
			Wrong Type	0	0
			Wrong Object	0	0
			Wrong Place	0	1

Continued on next page

			Expert Classification	HF Classifier 2.0
Specific Cognitive Functions	Observation	Observation Missed	0	0
		False Observation	0	0
		Wrong Identification	0	0
	Interpretation	Faulty diagnosis	0	0
		Wrong reasoning	0	0
		Decision error	0	0
		Delayed interpretation	0	0
		Incorrect prediction	0	0
	Planning	Inadequate plan	0	0
		Priority error	0	0
Temporary Person Related Functions		Memory failure	0	0
		Fear	0	0
		Distraction	0	0
		Fatigue	1	1
		Performance Variability	0	0
		Inattention	0	0
		Physiological stress	0	0
		Psychological stress	0	0

Continued on next page

			Expert Classification	HF Classifier 2.0
TECHNOLOGY	Permanent Person Related Functions	Functional impairment	0	0
		Cognitive style	0	0
		Cognitive bias	1	0
	Equipment	Equipment failure	1	0
		Software fault	0	0
	Procedures	Inadequate procedure	1	1
	Temporary Interface	Access limitations	1	1
		Ambiguous information	0	0
		Incomplete information	0	0
	Permanent Interface	Access problems	0	0
		Mislabeling	0	0
ORGANIZATION	Communication	Communication failure	1	1
		Missing information	1	1
	Organization	Maintenance failure	1	0
		Inadequate quality control	1	1
		Management problem	0	0
		Design failure	1	1
		Inadequate task allocation	1	1
		Social pressure	0	0

Continued on next page

			Expert Classification	HF Classifier 2.0
	Training	Insufficient skills	1	0
		Insufficient knowledge	1	0
	Ambient Conditions	Temperature	0	0
		Sound	0	0
		Humidity	0	0
		Illumination	0	0
		Other	0	0
		Adverse ambient conditions	1	0
	Working Conditions	Excessive demand	1	1
		Inadequate work place layout	1	1
		Inadequate team support	1	0
		Irregular working hours	0	0
			Sum of true positives	10
			Sum of true negatives	36
			Sum of false positives	1
			Sum of false negatives	7

Continued on next page

	Expert Classification	HF Classifier 2.0
	Accuracy	87%
	Precision	91%
	Recall	63%
	F1-Score	70%

Table 4.7: Case Study - Firefighting System Incident, HF Classifier Output

The tool and expert agreed on the following PSFs, Fatigue, Inadequate procedure, Access limitations, Communication failure, Missing information, Inadequate quality control, Design failure, Inadequate task allocation, Excessive demand, Inadequate work place layout. However, the expert additionally identified the following factors, Cognitive bias, Equipment failure, Insufficient skills, Insufficient knowledge, Adverse ambient conditions, Inadequate team support. This is reflected in Figure 4.4 and in the lower recall performance.

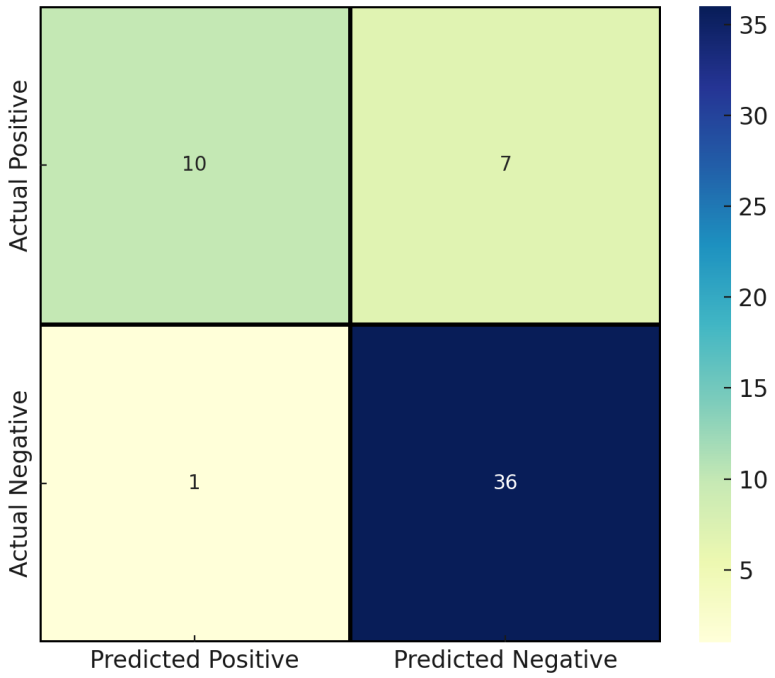

Figure 4.4: Confusion Matrix Heatmap - Firefighting System accident

One potential reason for this decline is that the report was initially written in Portuguese. Translating it likely introduced vocabulary and phrasing unfamiliar to the tool from its training phase. Furthermore, the tool has been trained and benchmarked against assessments from a single expert, potentially embedding a shared bias that might affect performance when comparing to another expert's judgement.

However, the tool identified the factor Wrong Place, which the expert initially did not identify. After further review it has been labelled as an oversight by the expert, highlighting the utility of the tool in complementing human expertise. Despite the challenges posed by translation, the majority of factors identified by the expert were also recognized by the tool, demonstrating its promising capabilities.

The summarization of this report takes the original report of 70 pages and 16,000 words, first to 1500 words after the extraction phase and then down to 485 words following the summary. As before shown before the metrics used in the evaluation of the tool can be calculated, and these are given in Table 4.8 and Table 4.9.

ROUGE Metric	F1-Score
ROUGE-1	50%
ROUGE-2	41%
ROUGE-L	45%
ROUGE-S	29%

Table 4.8: Case Study - Firefighting System Incident, Summarized Report ROUGE Metrics

Metric	Original Reports	Summarized Reports
Accuracy	87%	78%
Precision	91%	72%
Recall	63%	63%
F1-Score	70%	67%

Table 4.9: Case Study - Firefighting System Incident, Original vs Summarized Classification Performance

For this accident report there was a comparable level of performance to that displayed in Section 4.3.3. This report is similar in length to the test reports, and therefore requires less summarization, in turn losing less information.

The generated summary concisely, successfully, and accurately details the events that led to

the accident, providing context for many of the factors identified by the tool. For example, "*There was a lack of communication in the activities carried out in the engine generator room*" identifies a critical Communication Failure, and "*The unit lacked information about the safety procedures...*" highlights that the crew was Missing Information that impacted their reactions. The summary also included, "*...inspections by the safety technicians overlooked failures in the design of the fixed CO2 system, the absence of a lockout valve...*" This is not perfectly accurate to the original report, as these are two separate points that have been merged. However, this quote from the summary provides evidence for the classification identifying Inadequate Quality Control and Design Failure as PSFs in the accident. The error missed by the human expert but identified by the classification tool, Wrong Place, is supported by the summary including "*People remained in the room after the alarm...*"

The summarized report provides the necessary understanding of the human role in the incident to understand and engage with the human reliability/factors part of the accident review.

4.4.3 Third Case Study - Pigging Operation Procedure

Finally, the utility of *HF Classifier 2.0* in assessing procedure guides is explored, taking a transfer learning approach leveraging the shared vocabulary and context of industrial procedure and accidents, to preemptively identify vulnerabilities to specific influencing factors.

The exploratory case study focuses on a pigging operations guide from the oil and gas sector, an area where precise and reliable procedures are critical. Pigging ensures pipeline integrity by using devices to clean, inspect, and maintain pipelines without halting flow. The procedure provides details of the preparation, execution, and response to abnormalities, with stringent safety measures and specialized equipment. Effective pigging operations are vital for maintaining the efficiency and safety of oil and gas transportation systems.

The *HF Classifier 2.0*, initially untrained on procedure guides, has exhibited a remarkable capability to parse and analyze these documents due to its underlying understanding of language concerning processes and actions. By dissecting the procedure guide into individual steps and analyzing each for PSFs, multiple vulnerabilities, including faulty diagnosis, wrong reasoning, and inadequate procedures have been identified. Such insights are invaluable, providing the opportunity for potential risks to be further assessed and preemptively mitigated. This has the

potential to aid HRA assessors to identify and assess the possible vulnerability due to PSFs in safety critical procedures.

Feedback from industry experts highlights the potential of this tool as a supportive aid in incident investigation and operational safety analysis. While its ability to direct attention to potential areas of concern has been positively received, it is crucial to underscore that the tool's role is to assist, not to provide definitive solutions. By analyzing procedural documents and identifying potential latent risks or PSFs, it offers suggestions that require further investigation and expert evaluation.

This capability proves valuable in raising targeted questions, such as, "Are we confident this step is sufficiently robust?" or "Could this factor be contributing to a performance issue?" These insights generated are not conclusive but serve as a starting point for deeper exploration. In some cases, the tool might flag areas that, upon thorough evaluation by experts, could lead to actionable safety improvements that might have been otherwise overlooked.

While promising in its ability to support the identification of risks and PSFs, the tool should be considered a complementary resource within a comprehensive safety strategy. Its outputs require validation and integration with expert judgment, detailed investigations, and broader contextual analysis to effectively contribute to meaningful improvements in incident prevention and operational safety.

The case study of the tool's application in procedure guides demonstrates its utility in identifying and classifying human factor-related vulnerabilities, setting an example for its broader application in enhancing safety across high-stakes industries. However, it is essential, outside of data collection, to view the tools findings as insights providing a foundation for further exploration rather than definitive solutions.

Through these case studies, the practical applications of the developed tools in incident analysis, classification, and summarization are highlighted, showcasing their potential to improve incident investigation methodologies. By offering a systematic way to assess human factors in risk analysis, the tools can contribute to the enhancement of operational safety across various industries.

4.5 Limitations and Future Work

The tools' benefits and potential use in practical accident and safety assessments are underscored by robust performance metrics and successes in practical case studies. Despite these achievements, there remain avenues for improvement. Here, the challenges and constraints faced during development, implementation, and application are outlined, as well as areas that warrant further investigation and innovation. By acknowledging these limitations and charting a course for future work, a contribution to the continued advancement and refinement of automated accident analysis and the adoption of NLP and AI technologies in the field of human reliability and safety analysis is aimed for.

First, the size of the dataset in MATA-D poses a notable limitation, comprising only 238 accidents. The initial reason for this project's development is also a significant limiting factor in the training and performance of the developed tool. Additionally, the relatively small dataset size contributes to the risk of overfitting, particularly when fine-tuning advanced models like BERT, which require substantial and diverse data for optimal performance. Expanding the dataset will be a key focus for future iterations, potentially involving collaborations with industry stakeholders to collect and integrate additional reports.

Another concern is related to bias within the training data, the MATA-D was constructed and manually coded by a single expert. Thus, embedding the tool with their bias, as suggested when compared with another expert's assessment in the second case study. There have been efforts to mitigate bias through adherence to the CREAM framework and the provision of reasoning in MATA-D. However, to address this concern effectively, a manual assessment of a subset of reports, requiring expert involvement, would be beneficial albeit time-consuming.

The summarization tool can be characterized as a blend of extractive and abstractive techniques. While it targets specific sections and keywords, it employs abstractive methods to condense information into a readable summary. Currently, the tool is designed to prioritize the human factors in reports, which aligns with its intended purpose. However, this focus means that certain technical aspects influencing human performance may not receive the same level of attention. To enhance the accuracy and relevance of the summaries, especially concerning domain-specific terminology and critical incident details, the training of a fine-tuned layer for the summarization tool has been suggested. This layer would utilize summaries written by human experts as a training base. Although this approach would require a significant time investment in the initial

stages, the long-term benefits could be substantial.

When assessing the performance of the summarization tool, the limitations of the ROUGE metrics are highlighted. ROUGE primarily relies on vocabulary matching, disregarding nuances in semantics. The use of other assessment addresses these challenges in this work, however the incorporation of more advanced and recently developed metrics such as ROUGE-SEM, are worth consideration (Zhang et al. 2024). While the summarization effectively targets the human role, concerns regarding the potential omission of sections detailing the technical aspects that influence human performance, have been considered. A reworked extraction phase or the development of a separate technical summarizer could address this limitation effectively.

Continual advancements in NLP and LLM underline the importance of staying up to date with technological innovations. Future iterations of our tool should leverage these advancements, especially in the handling of multimedia elements like images and graphs.

The classifier can efficiently aid in identifying recurrent influencing factors, its role in informing specific safety recommendations should be viewed with caution. The classification output alone does not provide the detailed causal narratives required for meaningful safety interventions. Future work should explore ways to link classified factors with contextual causal descriptions, possibly through deeper integration of causal reasoning modules or human-in-the-loop methods to interpret classifier outputs more precisely.

Finally, the development of a user-friendly website to integrate our tools and facilitate data collection for MATA-D is imperative. This platform would enhance accessibility and usability for a wider audience, fostering further data accumulation and tool refinement.

4.6 Conclusion

This section presents key improvements in automated accident analysis. The improved performance of the *HF Classifier* provides greater confidence in its ability to support updates to the MATA-D repository, enhancing its value as a resource for structured data collection and trend identification within HRA, which can support deeper safety investigations and expert-led causal analysis. Expert oversight remains essential, albeit less intensive than with the previous version of the tool. This shift allows experts to focus on rarer and more nuanced factors in the dataset

(Moura et al. 2017a).

The tools demonstrate strong performance in classification and summarization, their primary value lies in supporting data organization and initial scoping of accident reports. The classification of influencing factors, such as identification of “Training” or “Inadequate Procedure”, is not intended to directly generate safety recommendations. Rather serve as entry points for more nuanced investigations. Meaningful safety improvements require a granular understanding of accident causality, procedural context, and human-machine interaction, aspects that go beyond what automated tools can currently extract.

The integration of the *Human-Centric Summarizer* further enriches the analytical toolkit, providing users with compact yet comprehensive insights into the human role and PSFs in accidents. Rather than accelerating the overall process, it enhances specific early stage steps, summarizing and contextualizing accident reports, allowing users to focus their efforts on areas of interest identified by the tool. By distilling complex accident reports into digestible summaries, the tool supports users in efficiently identifying potential areas of interest for further investigation. While this aids understanding and prioritization, informed decision-making and concrete risk mitigation strategies still rely on expert-led analysis.

The exploration of three case studies provided a thorough evaluation of the classifier and summarizer, showcasing their practical applications in incident analysis. The studies revealed the tools’ robust capabilities in identifying human errors and PSFs across diverse scenarios. For instance, in the FPSO CDSM incident, critical human errors and organizational issues such as faulty diagnosis and inadequate procedures were identified. The firefighting system incident highlighted similar insights, including communication failures and design flaws. Additionally, the analysis of a pigging operation procedure underscored the tool’s potential to assess SOPs and preemptively identify vulnerabilities. A significant outcome of these case studies was the tool’s ability to identify factors originally overlooked by human experts, a fundamental result that emphasizes the importance of integrating automated tools in accident analysis to enhance comprehensiveness. These results are promising but must be framed as supportive rather than definitive. The tools flag potential factors for further expert validation and investigation, and while they can point to issues overlooked in initial assessments, actionable insights require subsequent analysis and contextual understanding.

The case studies provided valuable insights into the performance and utility of the summarization tool. They confirmed the tool’s effectiveness in condensing lengthy and complex accident reports into concise, readable summaries. This capability is particularly beneficial in scenarios

where efficient assimilation of information is critical, allowing users to gain a coherent overview of the incidents without sifting through extensive documentation. Although challenges were noted in maintaining nuanced information from extensive reports with dense technical content or complex narrative structures. This limitation suggests that while the summarizer is adept at providing a high-level summary, it may sometimes omit finer details that could be significant in a full technical analysis.

Importantly, the case studies emphasized the *Human-Centric Summarizer's* role as an explainability layer for the classification tool. By providing clear and accessible interpretations of the classified factors, the summarizer enhances the transparency and usability of the classification results. This dual functionality not only supports the classification process by enhancing understanding of its outcomes but also helps communicate overall findings and ideas to stakeholders, particularly those who may lack the time or expertise to analyze full reports. However, it does not replace the need for experts to conduct in-depth reviews and thorough analysis when critical decisions are at stake.

Looking ahead, the trajectory of NLP models promises continued evolution and refinement. As new technologies emerge and datasets expand, NLP models will yield even more nuanced and accurate results, further enhancing their applicability in HRA. This work exemplifies the practical application of NLP in HRA, whether through data gathering initiatives or automating labor-intensive text-based tasks, thus paving the way for enhanced accident analysis methodologies and proactive safety measures.

Chapter 5

Enhancing Procedure Quality: Advanced Language Tools for Identifying Ambiguity and High-Potential Violation Triggers

This chapter introduces two novel tools that combine rule-based and ML techniques in NLP to improve the quality and clarity of SOPs in high-risk industries. The development was motivated by findings from the MATA-D dataset, where 45% of accidents were attributed to the Inadequate Procedure factor, highlighting the need for systematic approaches to improve procedural documentation. Designed to work in parallel, the first tool identifies ambiguities, while the second is trained on historical data to flag steps with high-risk potential if violated. Their development and application aim not only to ensure that SOPs are comprehensive, clear, and logical but also to support the deliberate refinement of procedures based on well-grounded safety insights. This approach fosters a proactive yet thoughtful strategy for enhancing safety and improving organizational performance in high-risk industries.

The first tool systematically analyzes SOPs to detect linguistic and structural ambiguities that could lead to misinterpretation or inconsistent execution. Using rule-based methodologies and NLP techniques, this tool identifies vague language, unclear instructions, and complex sentence structures that might introduce uncertainty. By flagging these ambiguities, the tool enables SOP authors and safety analysts to refine and clarify procedural directives, ensuring that instructions are easily comprehensible for operators. This proactive approach enhances the effectiveness of

SOPs by minimizing the likelihood of human error and improving compliance with industry standards.

The second tool is designed to identify SOP directives that, when violated, have significant consequences, thereby highlighting procedural steps that require enhanced clarity and enforcement. This tool is trained on datasets of past accidents, incidents, and near-misses, many of which involved procedural violations. By analyzing these historical events, it identifies directives that were previously violated and were associated with severe outcomes. The tool is then applied to SOPs to flag similar directives that share characteristics with those previously violated, thus identifying critical steps where compliance is most essential. By pinpointing high-risk directives, this tool provides a proactive approach to mitigating operational failures and ensuring procedural adherence in high-risk industries.

Each tool independently processes a submitted SOP. Upon submission, the text is extracted from the document and undergoes necessary preprocessing. The preprocessed SOP is then passed into either the *Ambiguity Identifier* or the *Violation Trigger Tool*, depending on the selected analysis. The *Ambiguity Identifier* outputs three files containing directives with issues related to 1.inconsistent units of measurement, 2.potentially ambiguous language, and 3.undefined abbreviations and acronyms. Conversely, the *Violation Trigger Tool* outputs a file containing directives that, based on past incident data, are associated with a high-potential risk if violated. This separation of tools ensures tailored evaluations depending on the specific quality aspect under review. A simplified workflow is shown in Figure 5.1.

While this work emphasizes the importance of timely communication, it equally recognizes that the development and modification of SOPs must be grounded in a thorough understanding of operational complexities. The tools presented here are designed to aid, not replace, human expertise and review, ensuring that procedural changes are deliberate and well-informed.

SOPs are foundational tools used by organizations and in industrial environments to ensure consistency and efficiency in the execution of tasks. SOPs dictate routine and emergency practices, encompassing purpose, terminology, roles, and step-by-step instructions detailing how to perform complex operations safely, effectively and consistently, thus forming the backbone of industrial safety management. They are critical in a wide range of industries, particularly in high-risk sectors like nuclear, oil & gas, and chemical processing, where precise, repeatable processes are essential to maintain quality, safety and compliance with regulatory standards (Office 2007). The clarity and accuracy of these procedures are paramount. Clear and precise SOPs are essential for consistent and correct execution, reducing the risk of errors, enhancing productivity,

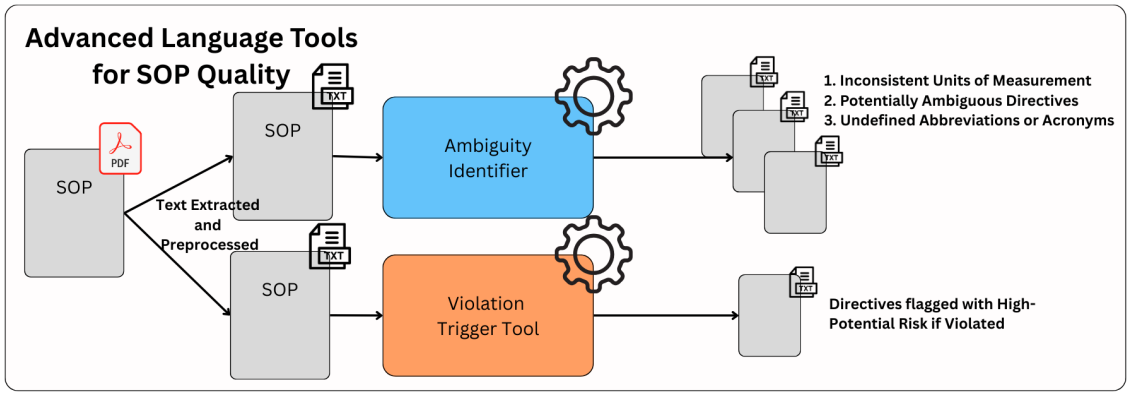

Figure 5.1: Workflow of Developed SOP tools

supporting effective training and communication, and ensuring compliance with industry standards and regulations. In environments where safety and precision are crucial, the importance of unambiguous instructions cannot be overstated.

Ambiguity in SOPs can lead to inconsistent outcomes, increased training costs due to the extra time needed to clarify unclear instructions, a higher rate of mistakes, and an overall increase in risk (Isin 2012). In general, ambiguity refers to the possibility of interpreting information in multiple ways (Curley et al. 1986). In the context of SOPs, ambiguity can arise from vague language, overly complex instructions, or insufficient details that leave too much room for individual interpretation. Such ambiguities can challenge even those deeply involved in the design and writing of guides, as their familiarity may mean they overlook practical operator perspectives (Manghani 2011). The presence of ambiguity can compromise the effectiveness of the SOP and lead to operational failures and contribute to cascading failures in high-risk environments (Sun et al. 2024).

Clear and easy-to-follow procedures are essential for minimizing factors that could increase the probability of human error because they ensure that all operators understand the tasks in the same way, reducing the chances of misinterpretation. By providing precise and straightforward instructions, SOPs help create a consistent workflow, thereby enhancing efficiency and safety across operations. This consistency is crucial for maintaining high standards of performance and reliability, ultimately leading to fewer mistakes and a more controlled and predictable opera-

tional environment.

Violations occur when work processes or activities are deliberately deviated from the standard procedures they should refer to, with the possibility of exposure to additional safety risks (Hudson et al. 1998) These violations often stem from perceived inefficiencies within the SOPs, resistance to standard practices, misunderstanding of the requirements, or extenuating circumstances. Violations are typically categorized into routine (habitual and often overlooked), situational (response to specific circumstances), and exceptional (rare, in extreme situations) types (Dougherty 1995). Such violations, whether due to oversight or inadequate design, undermine the efficacy of SOPs and introduce significant operational risks. Identifying procedural steps susceptible to intentional violations is crucial for enhancing compliance and rectifying potential operational vulnerabilities.

The term "circumvention" can serve as an alternative to "violation" when describing situations where operators intentionally omit or modify action sequences of SOPs to improve efficiency in task performance (Jang et al. 2021). This practice is based on their experience and belief that they can accomplish their tasks while saving time and minimizing the use of cognitive and physical resources. However, "circumvention" is more commonly associated with the nuclear power plant industry. Given that this work uses data from different industrial sectors, this intended action will continue to be described as "violation."

The concept of "work as done" versus "work as imagined" is a critical concept within organizational and safety management (Hollnagel 2017). "Work as imagined" refers to how tasks and procedures are designed and documented, often assuming ideal conditions and complete adherence to protocols. In contrast, "work as done" represents the reality of how tasks are actually performed in the field, where deviations from procedures often occur due to various practical challenges. This discrepancy poses a significant challenge because procedures based solely on "work as imagined" may not account for real-world complexities and constraints. It is essential that procedure guides reflect "work as done" to ensure that task analysis and safety protocols are both realistic and practical, thereby enhancing their effectiveness and adherence.

Further details of the motivations and any related studies are earlier discussed in Section 2.3.

The chapter begins by examining various types of ambiguity and the rules developed to identify them. Following this, the development of the tool for identifying high-potential violation triggers is detailed. The tools are subsequently applied to two case studies to demonstrate their practical application. Finally, the challenges, limitations, and opportunities for future work are discussed, concluding with a summary of the developments and their potential impact.

5.1 Ambiguity

In operational settings, ambiguity in SOPs can significantly impede clarity, consistency, and safety. Ambiguity occurs when instructions, guidelines, or procedural documentation allow for multiple interpretations, leading to operational inefficiencies and heightened potential of human errors. Ambiguities encountered in SOPs include general linguistic issues found in various texts, such as lexical and syntactic ambiguities, as well as domain-specific issues related to clarity about quantity, scope, abbreviations, and units of measurement. Each type of ambiguity introduces unique challenges that can lead to misunderstandings and discrepancies in the execution of tasks, thereby compromising the effectiveness of safety analysis.

5.1.1 Lexical Ambiguity

Lexical ambiguity involves the use of words or phrases that have multiple meanings (Li et al. 2024a). In general, the context of the word provides the necessary information to prevent misunderstanding or incorrect actions, however this is not always the case. An example that may be found in an industrial context is, "*Check the tank is clear.*" This phrase could lead to confusion due to the ambiguity of the word "*clear.*" One interpretation could be visually clear, suggesting one needs to ensure that the tank is visually free from obstructions and debris, meaning nothing is physically inside that should not be. Another possibility is that "*clear*" refers to the composition of the contents, implying that the tank's contents be clear, free from contaminants, sediment, or impurities.

This type of ambiguity can lead to erroneous actions, which are particularly critical in scenarios requiring precise operational standards.

To mitigate lexical ambiguity, SOPs should specify the context clearly within the document. Use of clear, unambiguous language and avoidance of jargon (unless defined) are essential steps towards clarifying SOPs at the lexical level.

5.1.2 Syntactic Ambiguity

Syntactic ambiguity emerges from unclear sentence structures, where the grammatical arrangement of words leads to multiple potential interpretations (Li et al. 2024a). The sentence structure could make it unclear how procedural steps are related to each other.

An SOP stating, "*Report all leaks observed after shutting down the system to the supervisor.*" Has multiple potential interpretations, this example could be interpreted as, leaks should only be reported if they are noticed after the system has already been shut down, or alternatively, first shut down the system and then report any leaks observed, regardless of when they were noticed.

This type of ambiguity can be particularly hazardous as it may lead to improper sequencing of tasks, potentially compromising safety. Clarification can be achieved by restructuring sentences to eliminate grammatical ambiguities and using bullet points or numbered steps to delineate procedures clearly.

5.1.3 Temporal Ambiguity

Ambiguity related to the lack of specific timing, sequence, or frequency of operations, which can cause inconsistencies in maintenance and routines, can be known as Temporal ambiguity (Campbell 2000). An instruction like "*Periodically shut down the machine*" could lead to variations in the maintenance schedule, affecting the machine's performance. Overuse or insufficient maintenance can both have dire consequences on the operational lifespan and safety of machinery.

SOPs should specify exact time intervals or conditions under which tasks are to be performed to avoid temporal ambiguity. Incorporating schedules, calendars, or explicit time-based triggers in SOPs can help standardize maintenance and operational procedures.

5.1.4 Quantity Ambiguity in SOPs

Another form of ambiguity occurs when the quantity or degree of action required is not specified, this arises particularly from vague quantifiers like "*some*", "*a few*", "*enough*", and others, which leave too much room for interpretation. In the directive to "*add some lubricant to the gear*" there is unclear guidance on the amount of lubricant, therefore operators might use an excessive or insufficient application, risking equipment failure or inefficiency. To address this, SOPs should provide precise measurements, potentially supported by visual aids or examples, to ensure that operators can achieve consistent results in tasks requiring specific quantities.

5.1.5 Conditional and Scope Ambiguity in SOPs

Conditional ambiguity arises when it is unclear under what circumstances certain actions should be taken (Geurts 2004). An SOP instruction like, "*If necessary, adjust the settings*," requires operators to make judgment calls that may not be consistent across the board. Whereas scope ambiguity arises when the extent or limits of a task are not well-defined (Li et al. 2024a). This could be in "*clean the workspace*" or "*perform maintenance on equipment*" where the specific areas/equipment and the level and degree of cleaning/maintenance required is not clearly specified. Conditional and scope ambiguities are two forms of instruction uncertainty that can significantly affect operational consistency and efficiency, creating room for interpretation that can lead to varied practices among operators. This variability can compromise the standardization of processes, which is critical in many environments. Clear criteria or examples are warranted and should be included to standardize responses to varying operational conditions.

5.1.6 Abbreviation and Acronym Ambiguity

Acronym ambiguity arises in SOPs when abbreviations or acronyms are used without being defined. This practice can lead to confusion, especially for new or less experienced operators who may not be familiar with the abbreviated terms. The confusion stems from the assumption that all readers have the same level of knowledge as designers and writers. Unfamiliar terms can hinder understanding and lead to procedural errors as employees may misinterpret what is required of them or fail to follow steps correctly.

For example, an SOP that instructs operators to "*consult the VFD*" could be misleading if the employees do not know that "*VFD*" stands for "*Variable Frequency Drive*", a device used for adjusting motor speed. Without knowing what "*VFD*" refers to, operators might ignore the instruction, seek help, or waste time trying to decode the abbreviation, thereby disrupting workflow and potentially compromising safety.

To mitigate this type of ambiguity, it is crucial to either define each acronym or abbreviation the first time it is used in the document or to include a glossary or reference table of terms at the beginning or end of the SOP. This approach ensures that all personnel, regardless of their familiarity with the terminology, have a clear and consistent understanding of the procedures, enhancing compliance and operational efficiency.

5.1.7 Units of Measurement Ambiguity

Ambiguity in units of measurement in SOPs pose significant risks, particularly in tasks requiring high precision. This type of ambiguity occurs when the units of measurement are not specified or are inconsistent throughout the SOP. For example, an instruction such as "*measure and compare the length*" without stating whether the measurement can lead to critical discrepancies. Such discrepancies can significantly affect the quality of the output, safety, and compliance with industry standards. Throughout the SOP, inconsistent application of measurement units, such as switching between metric and imperial, can also cause confusion and errors.

To address these issues effectively units of measurement should be specified in each step of where measurements are required. Ideally one measurement system/unit should be used throughout

the SOP to avoid confusion, and if industry standard varies or if certain parts require to use different units of measurement, include a conversion table within the SOP. This table should provide a quick reference to convert all units used in the SOP to the standard units used by the industry/company to ensure consistency and ease of understanding.

5.1.8 Mitigating Ambiguity in SOPs

Eliminating ambiguity in SOPs is imperative to enhancing operational clarity, safety, and efficiency. This requires a comprehensive approach to identifying and resolving ambiguous language and instructions across all aspects of SOP documentation. By ensuring that SOPs are precise, unambiguous, and systematically reviewed for clarity, organizations can significantly reduce operational risks and improve compliance with standards, thereby fostering a safer and more efficient working environment. A proactive approach to SOP management not only enhances operational outcomes but also supports a culture of safety and precision in industrial settings and reduces the probability of human error.

5.2 Ambiguity Identification Tool

To enhance the management, assessment, and review of SOPs, a dedicated tool (*Ambiguity Identifier*) has been developed specifically to identify ambiguities and uncertain procedural instructions. The creation of such a tool was motivated by the inherent difficulties associated with identifying ambiguous texts. Often, the text seems clear to those who write it, while the individuals executing the SOPs do not always have the opportunity to highlight their difficulties in understanding it. This tool aims to assist the writers of the SOPs or human reliability assessors in evaluating and enhancing the quality of the procedures before they are handed over to the executors. By doing so, it ensures that the procedures are clear and comprehensible, ultimately reducing the risk of user errors and improving operational efficiency.

A significant challenge is the absence of a pre-labeled or annotated corpus of ambiguous texts, es-

pecially those that are procedural in nature. This scarcity is compounded by the time-consuming nature of manually constructing a sufficiently large dataset.

Given these constraints, an AI-driven approach was not considered feasible at this stage. Instead, the tool adopts a rule-based methodology, which has been shown to be effective in similar studies. This approach involves the application of predefined, structured rules to systematically analyze text for ambiguities. These rules are meticulously crafted based on established linguistic principles, recognized patterns, and specific vocabularies known to lead to ambiguities.

Beyond the minimal data requirement there are multiple advantages of employing a rule-based system. The structured nature of the rules supports systematic analysis, ensuring a consistent and comprehensive examination of texts. Rule-based systems can also quickly analyze texts with clear and explainable outputs, aiding swift decision-making. The rules can be easily modified or extended as new patterns of ambiguities are discovered, for specific domains and industries, or as procedural requirements evolve.

For each type of ambiguity previously identified, a corresponding set of rules and algorithms have been developed. This methodological choice ensures that the tool is not only tailored to address specific ambiguities but is also capable of evolving to include new types of ambiguities.

5.2.1 Lexical Ambiguity Rule

Developing a way to identify lexical ambiguity within text requires identifying terms with multiple possible meanings, and then incorporating the surrounding contexts meaning to decide whether the terms meaning is clearly implied.

The PoS tagger from WMatrix, a corpus analysis and comparison tool, inspired an approach to achieve this (Rayson 2009). The PoS tagger profiles words by quantifying their usage in different grammatical contexts. It can discern how likely a word is to be a noun versus a verb. For example the term "*Document*" appears multiple times in one SOP, for one occurrence the PoS tagger assigned, `NN1` (singular common noun) `- 66` and `VV0` (base form of a verb) `- 34`, meaning it is 66% confident it's a noun, and then another time it assigned `NN1 - 100` and `VV0 - 0`, meaning it is 100% certain it's a noun in this context.

This inspired the approach to the rule and functionality developed in this work to identify lexical ambiguity. First using a different PoS tagger potentially ambiguous words are identified, specific-

ally homographs. Homographs are words that are spelled the same but have different meanings (and sometimes different pronunciations) when they represent different parts of speech (Grammarly 2023).

The sentences containing these words are then encoded and input into the LLM, BERT (Devlin et al. 2018). One of the outputs from the BERT model contains the contextual embedding for each word. This is a representation that takes into account the context in which they appear, making them powerful for understanding nuanced meanings.

Then for each ambiguous term, the possible definitions are retrieved from a lexical database. It then calculates the embeddings for each definition. A similarity calculation is then used to compare these definition embeddings with the context informed embeddings from the BERT model, to find the most probable matching definition in the lexical database.

If the most probable definition is not the first three in the lexical database, or if none of the definitions have a satisfactory similarity, the word and sentence is flagged as ambiguous.

5.2.2 Syntactic Ambiguity Rule

The syntactic rule-based algorithm designed to detect syntactic ambiguities, focuses primarily on complex sentences with subordinate clauses and potential ambiguities introduced by prepositional phrases.

5.2.2.1 Identification of Complex Sentences with Subordinate Clauses

The first rule of the algorithm determines the complexity of a sentence by identifying the presence and number of subordinate clauses. Subordinate clauses are dependent clauses that contain a subject and a verb but do not express a complete thought and cannot stand alone as a sentence (Traffis 2020). These clauses typically complicate sentence structure, which can lead to ambiguity.

Subordinate clauses can contain additional information that modifies or clarifies the main clause. However, if the placement or connection of the subordinate clause to the main clause is unclear, it can lead to multiple interpretations. Sentences containing subordinated clauses are usually longer and structurally more complex. This complexity might obscure the necessary actions, especially when multiple subjects and verbs rely on implicit relationships within the main clause that are not immediately obvious. When subordinate clauses are nested within each other, it can be particularly challenging to decipher the sentence. Each additional layer of subordination adds to the cognitive load required to parse the sentence, increasing the potential for misinterpretation. The algorithm uses dependency parsing to analyze the grammatical structure of sentences. This technique represents the sentence as a tree-like structure where each word is connected to another through a dependency relation. Subordinate clauses are identified by searching for tokens with a specific dependency label, which denotes markers of subordination (e.g., conjunctions, prepositions, or other types of subordinating words). A sentence is considered complex and potentially ambiguous if it contains more than two subordinate clauses.

5.2.2.2 Detection of Prepositional Phrase Attachment Ambiguities

The second part of the syntactic algorithm focuses on detecting ambiguities related to the attachment of prepositional phrases. These phrases begin with a preposition and end with an object of the preposition, often including modifiers that describe the object. Prepositional phrases can logically modify more than one element in a sentence, leading to different interpretations depending on their attachment (Merlo and Ferrer 2006). These phrases often imply relationships that rely heavily on contextual understanding, which can lead to confusion.

The algorithm checks for the preposition tag and examines the two preceding words or tokens. If both preceding words could logically be modified by the prepositional phrase, the sentence is flagged for potential attachment ambiguity.

The combination of these rule-based algorithms effectively highlights sentences with potential

syntactic ambiguities, arising from subordinate clauses and prepositional phrases. By systematically identifying these structures and the ambiguities due to their complexities, the algorithm serves as an effective tool for evaluating procedural steps. Allowing the reallocation of time spent manually identifying syntactic ambiguity to the improvement and refinement of SOPs.

5.2.3 Temporal Ambiguity Rule

Identifying temporal ambiguity in instructions involves detecting instances where the timing, sequence, or duration of events or actions is unclear or can be interpreted in multiple ways. A corpus of vague temporal words and expressions was constructed, that is terms which do not specify exact timings, including terms such as "*frequently*", "*periodically*", "*every so often*" etc. A simple algorithm then checks each procedural step for these terms and flags any occurrences as potentially ambiguous.

Other temporal conjunctions and adverbs such as "*before*", "*after*", "*during*", "*simultaneously*", etc. are not inherently ambiguous, however they can be if it is not clear which events they are connecting. Ambiguity arises when instructions do not explicitly state the sequence of actions, potentially leading to multiple valid interpretations of the order in which tasks should be completed.

To identify these potential ambiguities, the algorithm begins by parsing each word in a sentence searching for temporal conjunctions and adverbs. Again, using dependency parsing, the surrounding grammatical structures within the text are analyzed to determine the relationships between the identified temporal qualifier and the adjacent nouns or verbs. This analysis helps identify whether the temporal phrases could modify the preceding and/or following actions. Sentences where such potential ambiguity is detected are flagged, as they have the potential to lead to confusion about the timing of events.

Combining both algorithms gives a comprehensive tool for identifying potential temporal ambiguity. These can then be manually resolved by clarifying or revising the wording to specify the timing, sequence, or relationships more explicitly.

5.2.4 Quantity Ambiguity in SOPs Rule

The approach for identifying quantity related ambiguity is similar to the first part of the temporal ambiguity approach. In general, many terms and ways of expressing quantity can be very precise, however also not. A corpus of ambiguous quantity related words was constructed, containing terms such as "*some*", "*many*", "*a handful*", "*enough*", by combining existing lists of numeric hedge words and other terms identified from the reading and analysis of example SOPs (Ferson et al. 2015).

A simple algorithm then checks each step of the procedure, flagging any containing terms from the ambiguous quantity corpus. The flagged steps should then be reviewed and clarified to ensure precision and consistency in the communication of quantity within the procedure. This review process may involve consulting subject matter experts, vendors, the system designers, and the design specifications, to determine the intended meaning behind the flagged terms and potentially replacing them with more specific or quantifiable language where necessary. Additionally, providing context or defining the intended range or amount can help mitigate ambiguity and ensure that the procedure can be effectively followed by users without confusion or misinterpretation.

5.2.5 Conditional and Scope Ambiguity in SOPs Rules

Conditional ambiguity often arises when it's unclear whether the outcome described in the main clause of a sentence strictly depends on the condition stated in the subordinate clause. Conditional conjunctions such as "*if*", "*when*", "*unless*", etc. are identified, their presence indicates that the sentence likely has a conditional structure. If a conditional keyword is found, the sentence's grammatical structure, specifically the dependency parse tree is then analyzed. The condition is identified in the structure and is checked for disjunctive conjunctions (like "*or*" and "*either*") and additional conditional conjunction.

Disjunctive conjunctions can introduce confusion in conditional sentences by creating multiple possible scenarios within the same condition. When these conjunctions are used within a conditional clause, they often split the condition into separate parts, each leading to potentially

different outcomes or interpretations of what must be true for the main clause to take effect. This can make it unclear which specific condition or set of conditions need to be met for the consequence in the main clause to occur, including further conditional dependencies.

Having a conditional clause nested within another conditional clause can lead to confusion due to the layering of multiple, dependent conditions that must be satisfied for the outcome to occur. This creates a more complex logical structure that can be difficult to follow and interpret accurately. In such sentences, each conditional clause adds a new layer of criteria that influences the ultimate outcome stated in the main clause. This layering requires the reader to keep track of multiple contingencies and understand how they interact, which can easily lead to misunderstandings and mistakes.

This approach to identifying conditional ambiguity, by analyzing the presence of conditional conjunctions and the structural complexity of the dependency parse tree, is effective because it systematically dissects sentence structure to pinpoint potential sources of confusion.

Identifying scope ambiguity presents multiple challenges, and creating a comprehensive set of rules to cover all possible cases of scope ambiguity is extremely challenging, as there can be subtle nuances and contextual factors that affect the scope interpretation. The interpretation of scope often depends on the broader context. Attempting to check for this contextual information is a significant challenge, it requires a deeper understanding of the domain and complete document than can be coded within a ruleset.

In task directives the presence of a direct object to specify the target of the verb's action typically enhances this clarity. The absence of direct objects can contribute to scope ambiguity, affecting the extent and responsibility of the task. The rule identifies and tags each verb, before analyzing the surrounding syntactic structure of the sentence using dependency parsing. For each verb, the code looks at its children, tokens that depend syntactically on the verb. The code filters these children to find the words where the dependency label is direct object. If it is unable to locate such a token this sentence is flagged as potentially ambiguity.

As well as this, several other contributors to scope ambiguity include factors identified by other rules developed in this project, in particular the quantity ambiguity and the conditional ambiguity rules.

Accurately identifying and resolving conditional and scope ambiguities presents significant challenges and necessitates a detailed analysis of sentence structure using advanced linguistic meth-

ods. Refining these rules to include specific operational contexts will help develop a more robust rule set, ensuring the highest standards of clarity and effectiveness in SOPs. Future integration of advancements in NLP will be crucial for a smarter approach to detecting these ambiguities in SOPs.

5.2.6 Abbreviation and Acronym Ambiguity Rules

Without proper definition and context, shorthand forms can lead to confusion and misinterpretation. Our methodology addresses this ambiguity by systematically identifying and verifying abbreviations and acronyms using regular expression (regex) techniques (Li et al. 2008).

To detect potential abbreviations and acronyms, a regex pattern designed to capture sequences that are typically all uppercase and range between 2 to 5 characters in length has been emplyed. The regex pattern used is "`\b[A-Z]{2,5}\b`", which is used to identify these sequences within the document and temporarily store them as potentially ambiguous.

Once identified, the next step is to verify whether each abbreviation or acronym has been defined within the text. Using several regex patterns found in Table 5.1 common definition formats are searched for, this table can be easily expanded to include other formats found in specific company's SOPs. These patterns help in capturing most definition styles employed any acronyms/abbreviations found in this format are removed from the potentially ambiguous list.

Free Form Text	**Regex Pattern**
Acronym/Abbreviation (Full Form)	`\b([A-Z]{2,5})\s*\(([^)]+)\)`
Full Form (Acronym/Abbreviation)	`\b([^)]+)\s*\(([A-Z]{2,5})\)`
Acronym/Abbreviation - Full Form	`\b([A-Z]{2,5})\s*-\s*([^.\n]+)`
Full Form – Acronym/Abbreviation	`\b([^.\n]+)\s*-\s*([A-Z]{2,5})`

Table 5.1: Acronym/Abbreviation Definition Regex Patterns

It is also necessary to differentiate between abbreviations/acronyms and units of measurement. This differentiation is crucial as units are often formatted similarly but do not necessarily require in-text definitions. This is achieved by checking if numerical data proceeds the potential acronyms/abbreviations, filtering these out to prevent them being erroneously identified as po-

tential causes of ambiguities.

Additionally, acronyms/abbreviations are often defined within tables. Therefore, tables within the document also need to be checked. The table structure is extracted and checked for the earlier identified acronyms/abbreviations. If found in the table structure, adjacent cells are inspected for text strings, it is assumed that these strings likely serve as definitions or explanations for the abbreviations or acronyms found within the table, removing these from the potentially ambiguous list.

The remaining acronyms/abbreviations not defined through the above methods are output as potentially ambiguous. These can then be manually reviewed, and then defined within the SOP if determined to be necessary. This approach captures a broad spectrum of abbreviation and acronym definitions, the practical implementation could potentially be improved by incorporating domain-specific dictionaries that frequently appear within certain contexts or domains. However, despite the potential familiarity of some terms within specialized domains, it is a best practice to define every abbreviation or acronym at least once in a document to prevent any misunderstanding, thereby enhancing the clarity of the communication, efficiency and reducing the risk of human error.

5.2.7 Units of Measurement Ambiguity Rule

Inconsistent or unclear usage of measurement units can lead to significant discrepancies in understanding procedural requirements. This classification aids in the systematic analysis of the documents. To detect ambiguities in the presentation of numerical data, specifically related to measurement units within SOPs, a comprehensive set of measurement units was compiled. These units are categorized by what they measure: length, weight, volume, etc., and were further subdivided into groups based on measurement systems, such as metric (*mm, cm, m, km*) versus imperial (*inch, feet, yard, mile*).

From the analysis of various SOPs, it was determined that the majority of numerical and measurable values are presented within tables, with units included in the table headings. This standardized presentation significantly reduces ambiguity in data interpretation.

For numerical data outside of tables, including content but excluding page numbers and section headings, the document is searched for numbers. The presence of a defined unit immediately

following a numeric value is then verified. If no unit is present, the sentence is flagged.

The consistency of unit usage within a single document is also examined. Inconsistencies, such as the mixed use of metric and imperial units or different units of the same type (e.g., mixing *mm, cm, and yards*), all units causing this inconsistency are output. This step is crucial as changing unit types within a document can confuse the reader, leading to potential oversight and erroneous assumptions. This output might be used to support SOPs' walkthroughs in real installations, so the user can check if the different units in procedures match the units shown in the human machine interfaces.

The methodology presented here provides an effective strategy for identifying ambiguities in the use of measurement units in SOPs. This structured approach to managing measurement units within SOPs sets a foundation for clearer communication and enhanced operational precision across various sectors, mitigating risks associated with misinterpretation of critical measurements and data.

5.2.8 Implementation

By combining these developed rules and algorithms together, this specialized tool effectively addresses the challenge of identifying ambiguities in SOPs through a robust, rule-based methodology. By meticulously crafting rules based on established linguistic principles and recognized patterns, the tool systematically analyzes procedural texts to pinpoint ambiguities that could potentially compromise clarity and operational efficacy. The tool's design allows for the consistent, comprehensive examination of texts, providing clear, explainable outputs that facilitate swift decision-making and continuous improvement of SOPs.

The implementation of these rules leverages various Python libraries and toolkits, enhancing the tool's capability to handle complex textual analysis efficiently. Users can then select specific types of ambiguities they wish to identify, tailoring the analysis to their needs. The tools processes are outlined in Figure 5.2. First, the documents are converted from PDF into a raw text, stripping non-essential elements, such as formatting, to focus purely on the content.

The tool analyzes the text by applying the selected rules one by one, thoroughly examining each directive for potential ambiguities. The results are systematically compiled into a text file that lists potentially ambiguous sentences. Additionally, two separate files are generated, one

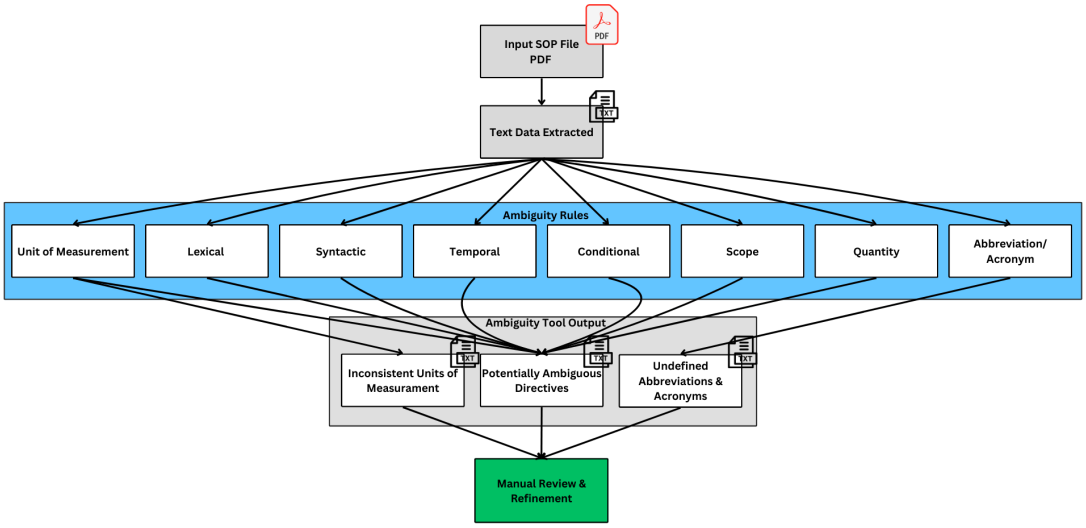

Figure 5.2: Ambiguity Identifier Workflow

for Units of Measurement, which provides the units identified during the consistency check, and one for undefined Abbreviations and Acronyms. These outputs not only highlight areas of concern but also organize the findings in a manner conducive to further review and refinement of the SOPs, thereby enhancing the precision and reliability of procedural documentation across various domains.

Further discussion on the technical development of the *Ambiguity Identifier* is presented in Chapter 6, Section 6.5.1.

5.3 High-Potential Violation Trigger Identification Tool

The *High-Potential Violation Trigger Identification tool* (*Violation Trigger tool*) leverages past incidents to enhance learning opportunities. Initially, a database of incident reports was compiled and manually analyzed to identify violations as influencing factors. This data is used to train an NLP classification model, based on the hypothesis that shared language between incident reports and SOPs will enable the model to apply its insights to procedural directives. Despite

structural differences, this linguistic overlap supports model generalization. Rather than explicitly identifying the potential for violation in the SOPs, the model highlights procedural steps that exhibit linguistic characteristics of high-risk directives if violated based on past incidents, improving predictive capabilities.

5.3.1 Dataset

To gather the necessary data for training the NLP classification model, a database comprising two key sources has been assembled. The first source includes incident reports from the International Oil and Gas Producers (IOGP), which focus on oil and gas industry incidents (IOGP 2024). The second source, MATA-D, offering its cross-industry perspective (Moura et al. 2016). Together, these sources provide a comprehensive dataset of 300 reports used to train the model.

5.3.2 IOGP Database

The IOGP manages a comprehensive database known as the Safety Zone, established to collect and analyze data on incidents within the oil and gas industry (IOGP 2024). This repository includes detailed records of fatal incidents dating back to 1991, capturing events reported by both companies and contractors, and more recently, incidents involving third-party fatalities.
Since 2000, the IOGP has also been collecting data on high potential events, which are incidents or near misses that could realistically have resulted in one or more fatalities under different circumstances (IOGP 2024). A near miss is an incident where an adverse event is narrowly avoided. Although no actual harm or significant damage results, near misses are still considered critical safety concerns as they highlight potential vulnerabilities in the system or process (Jones et al. 1999).
It is important to note that the IOGP database is not exhaustive and should not be considered a complete record of all high potential events in the upstream oil industry or among its membership. Despite this, the information contained in the database offers valuable insights into violations and their impacts, which are crucial for the development and refinement of predictive

models aimed at enhancing safety measures.

Within both IOGP collections, 119 incidents where a violation was stated or named as a cause or contributing factor were identified and form a critical part of the training dataset for the NLP classification model. The database entries typically include three sections, "*Incident Description*", "*What Went Wrong*" and "*Corrective Action*". These sections are combined into a single "*Incident Report*" in the compiled dataset, providing the training text data necessary for the model.

5.3.3 MATA-D

Part of the data used in this tools development is from the MATA-D (Section 2.1.1). However, violations are not directly accounted for in the framework used in its construction, so they had to be manually assessed. This assessment involved reviewing the comments added to each accident report by the original authors, keyword searches and reviewing report summaries using the *Human-Centric Summarizer* (Section 4.3).

The MATA-D contributes 184 accidents to the compiled dataset, with 16 identified as having a violation committed discussed in the accident report.

5.3.4 Compiled Dataset

The compiled dataset contains 303 total reports, with 135 identified as having a violation as a contributing factor. Although there are several thousand more incidents available, particularly in the IOGP database, they are not included to maintain a balanced training set. The non-violation related data is instead included from the MATA-D, where the reports are more comprehensive. This compiled dataset contains an "*Incident Report*" field and a binary classification indicating whether a violation was an influencing factor in the incident or not.

An analysis of the vocabulary in SOP documents and the compiled incident reports was completed to showcase the similarities and support the underlying hypothesis that the model will be able to effectively apply its findings to procedural directives.

First text was extracted from several SOPs and the incident reports. Then, punctuation, numbers, common stop words (like "*the*", "*and*", "*of*"), and place/company names were removed, leaving just the keywords. These keywords contained a mix of ordinary nouns and verbs, as well as industry-specific language and terms.

Each word was then reduced to its stem form, meaning that words were simplified to their root form (e.g., "*running*" becomes "*run*"), and the frequency of occurrence in the texts was counted. Of the keywords identified in the procedure guides, 99.5% of the terms were also present in the compiled incident reports from IOGP and the MATA-D. Therefore, the model has been fine-tuned on a text dataset very similar to its target domain and has learnt most of the vocabulary it will encounter in the SOPs from within a similar context. The main limitation of this approach is that the specific SOPs referenced in the training dataset do not perfectly align with the SOPs to be analyzed by the tool. However, procedural violations often exhibit common patterns across industries, which allows for generalization (Phipps et al. 2015). But expanding the training dataset to include a broader range of incidents, and eventually labeled SOPs will improve model accuracy in industry-specific applications.

To further highlight the shared language between both document types, word clouds for each document type's keywords are shown in Figure 5.3 and Figure 5.4. The larger the terms, the more frequently they occur. From these word clouds, it can be seen that terms such as "*piping*", "*fire*", "*process*", and "*pressure*" are common in both document types. Additionally, document-specific language is evident: words like "*incident*", "*explosion*", and "*investigate*" occur more frequently in incident reports, while "*test*", "*startup*", and "*process*" are more common in SOPs.

5.3.5 Classification Model

The compiled database is used as the training date for the proposed Violation Trigger tool model. This tool utilizes the same methodology as the *HF Classifier 2.0* (Section 4.2). It leverages a modified BERT model, with an additional output classification layer, fine-tuned on a pre-labeled dataset.

The model's nuanced understanding of context makes it highly adaptable to various types of text. This capability can then be effectively transferred to related domain texts, which often

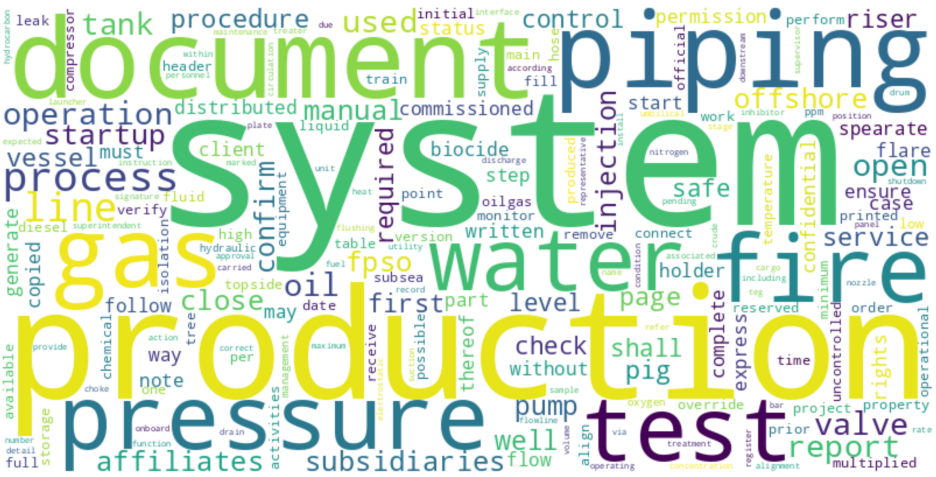

Figure 5.3: SOP Keyword Word Cloud

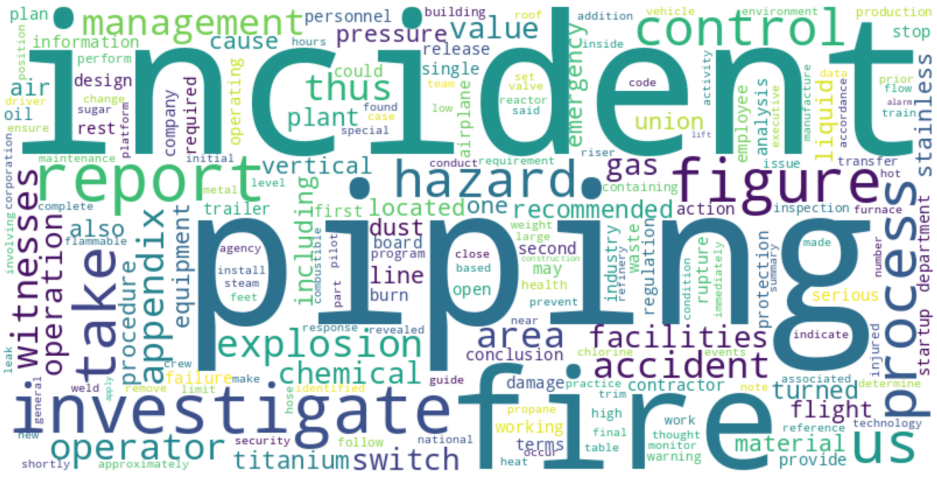

Figure 5.4: Incident Report Keyword Word Cloud

share similar formal language and thematic elements. As demonstrated in Section 4.4.3 (*Third Case Study - Pigging Operation Procedure*), the fine-tuned classification model was successfully applied to different types of documentation without the need for additional specialized training in such materials,

This process exemplifies transfer learning, where a model trained in one domain is adapted for use in another, leveraging its learned features for different but related tasks. This is a key strength of BERT, making it ideal for the proposed approach, fine-tuning the classification task on incident reports and applying it to SOP directives.

5.3.6 Out-Of-Distribution Considerations

While the model has been designed for cross-industry generalization, it is important to consider how this shift from descriptive narratives (incident reports) to structured directives (SOPs) may introduce out-of-distribution (OOD) characteristics.

Despite strong vocabulary overlap across industries, differences in structure and linguistic form between incident reports and SOPs could introduce a distributional shift (Liu et al. 2024). Incident reports are post-event narratives, often written in past tense and with cause-effect relationships, whereas SOPs are prescriptive and imperative, detailing required actions. While the core concepts remain aligned, these differences in linguistic framing and information presentation can impact how the model interprets high-risk procedural steps.

This scenario does not fit into far-OOD (completely different domains) or near-OOD (same domain, different categories) (Hendrycks et al. 2020). Instead, it falls somewhere in between, where the content remains procedural but the format and intent shift from reactive to proactive. BERT-based models, like the one used in our tool, handle domain generalization well but can be sensitive to structural variations (Liu et al. 2024). This is due to representation anisotropy, where embeddings of different sentence types (narrative vs. directive) may cluster differently in vector space, potentially impacting how well the model transfers its learned representations to new formats

Several design choices in the approach help mitigate these structural differences and improve the model's ability to generalize across SOPs and incident reports. The model is trained on incident reports from multiple industries, ensuring exposure to a diverse set of contexts. This reduces

the risk of overfitting to a single industry's linguistic style and helps it capture commonalities in procedural violations, even when structural variations exist between reports and SOPs. Additionally, the model utilizes contextualized word embeddings, rather than relying solely on surface-level text matching, allowing it to associate narrative-style descriptions of past violations (e.g., "The operator bypassed the safety interlock") with their corresponding prescriptive SOP directives (e.g., "Ensure the safety interlock is engaged before proceeding"). By focusing on semantic similarity rather than exact phrasing, the model can better generalize across formats.

5.3.7 Implementation

Before the fine-tuned classification model can be applied to new text, the input must be preprocessed. To facilitate this, the necessary preprocessing functionality has been integrated into the tool. First, from the uploaded SOPs irrelevant sections, such as title pages, contents, and appendices, and formatting such as page numbers, footers, etc. are removed. Then the remaining text is split into procedural steps and sections. This is accomplished using NLP techniques where the text is segmented into individual sentences and sections based on punctuation and formatting cues. Each segmented section of text is then tokenized and prepared to be processed through the fine-tuned BERT classification model. The model evaluates each section for language patterns and contexts indicative of high-potential violation. Sections identified as such are flagged and subsequently stored. Finally, these are aggregated and output as a text file for further review and analysis. This systematic approach ensures that each part of the report is thoroughly analyzed, allowing for precise identification of critical points where violations with high-risk potential may occur. This information is intended to supplement expert analysis and should be reviewed within the context of full operational understanding and system constraints before any procedural changes are implemented. The data gathering, labeling, modeling training and testing are outlined in Figure 5.5.

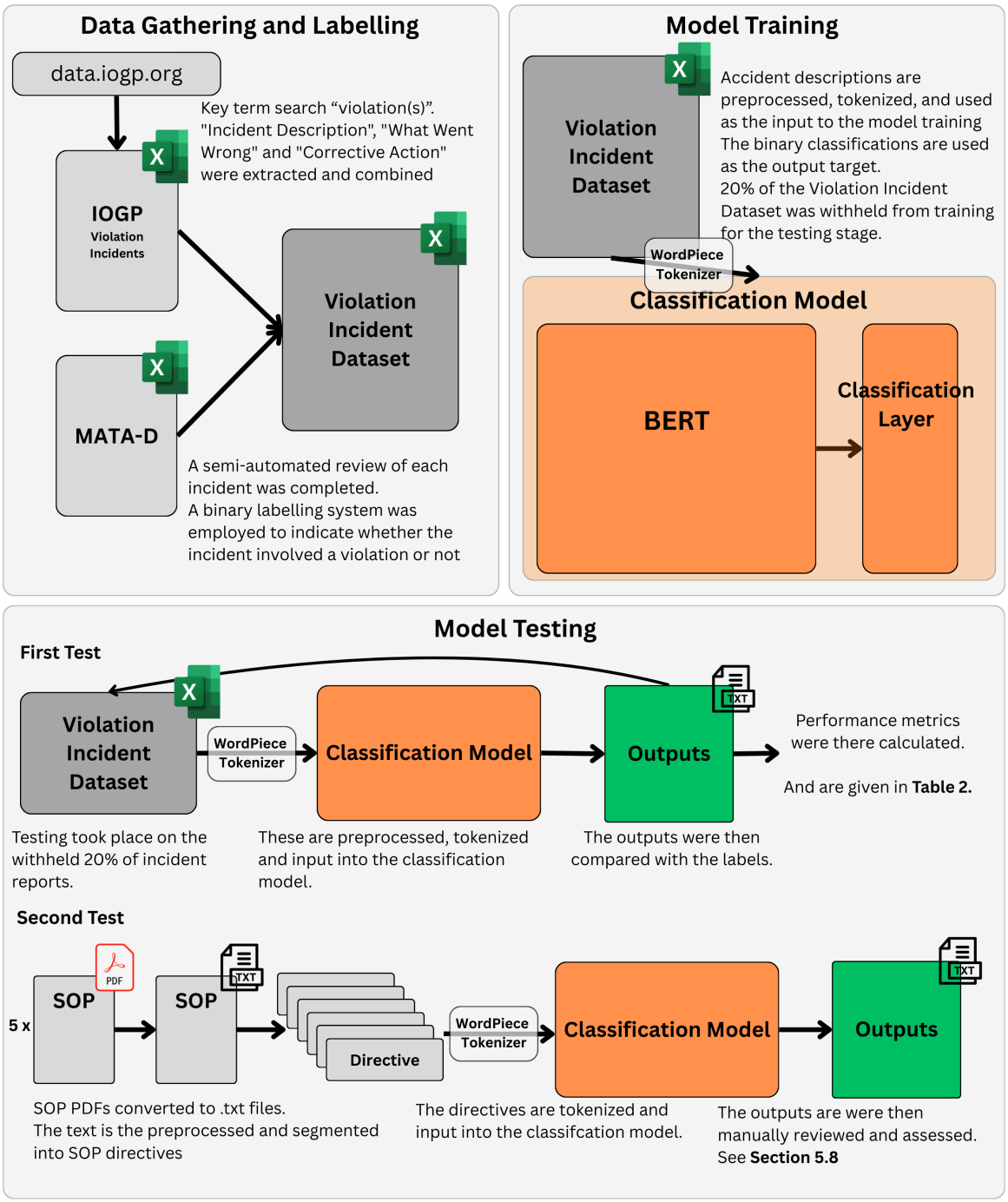

Figure 5.5: High-Potential Violation Trigger Identification Tool Development Process

5.3.8 Performance and Assessment

To assess the performance of the *Violation Trigger tool*, its performance classifying incident reports withheld from the training process can be analyzed. The performance of classification tools is again quantified using the metrics from Section 3.3 (Evaluating Performance): accuracy, precision, recall and F1-score. In this context, a TP is a correct classification where a violation was a factor, and a TN is a correct classification where a violation was not a factor. Conversely, a FP is an incorrect prediction where a violation was classified as a factor when it was not, and a FN is an incorrect prediction where a violation was not classified as a factor when it was.
The performance metrics for the *Violation Trigger tool*, obtained from a test set comprising approximately 40 cases (20% of the total dataset), are given in Table 5.2.

Metric	**Score**
Accuracy	94.6%
Precision	96.9%
Recall	90.5%
F1-Score	93.6%

Table 5.2: Violation Classifier Performance on Test Set

These metrics indicate significantly higher performance compared to the metrics obtained by the *HF Classifier 2.0* using the same methodology (Table 4.1). The improvement can be attributed to this study's focus on a single factor and the use of a well-balanced dataset. The previous study demonstrated effective application when applied to different documents, despite lower performance metrics. Therefore, given its higher performance, it is expected that this tool will achieve at least equal success and provide valuable insights when applied to this projects target domain of SOPs.
Assessing the tool's possible performance on SOPs is challenging, as even manually identifying procedure directives that may be violated with high-risk potential is a significantly difficult task. To evaluate the tool, it has been applied to five SOPs. Directives or sections classified with a high-potential violation factor were then manually reviewed to assess the risk potential if they are not followed correctly.
From the five SOPs, which ranged from seven pages to 240 pages, 26 sections were identified

as directives vulnerable to high-potential violations. Among these, two sections were incorrectly identified as directives when they were more administrative in nature. In four cases, the identified directives were very unlikely to lead to significant risk if violated. In eight cases, the potential violations were higher risk but still unlikely to lead to a major incident. In the remaining 12 cases, it was determined violating the directive could lead directly to a significant increase in risk and the likelihood of a major incident.

The manual assessment revealed that the tool tends to identify steps involving terms such as "*ensuring*" or "*confirming*", or those mentioning initial or final checks. This pattern suggests that the tool is particularly sensitive to actions that require verification or validation, which are critical control points in many processes. These steps are often pivotal in preventing errors and ensuring compliance, hence their frequent flagging by the tool. While this sensitivity helps in identifying high-risk areas, it may also result in FPs, particularly in sections where such language is used for administrative purposes rather than operational directives.

Overall, the *Violation Trigger tool* shows promising results. Its application could be a key component of an effective strategy to identify procedural steps susceptible to high potential violations. By incorporating this tool, organizations can improve the quality of their SOPs before issuing them to the operators. This way they can then liaise with operators to identify actual operations and behaviors, discuss the identified high-risk steps, emphasize these more in the SOPs, and provide additional training if necessary. This proactive approach can enhance compliance, reduce the likelihood of major incidents, and improve overall operational safety.

5.4 SOP Tools - Case Studies

These case studies examine SOPs from two companies in the offshore oil and gas industry, referred to as Company A and Company B. Both SOPs are instrumental in outlining the operational protocols necessary to ensure safety in high-risk environments. These SOPs provide an opportunity to test the specialized tools developed in this work for pinpointing ambiguities and identifying high-risk potential steps in procedural documentation. The subsequent sections

provide insights into their effectiveness and areas for improvement. When presenting examples from the SOPs, certain details have been modified to ensure confidentiality. The example actions within a procedure are described in a generalized form, this approach retains the integrity of the process explanation without revealing sensitive or specific operational data.

5.4.1 Case Study One - PIG Operations

Company A's SOP encompasses a concise document of 7 total pages, however the complete procedure steps are contained within just 3 pages. It is designed to guide the operational process involving Pipeline Inspection Gauges (PIGs), which are tools used for cleaning and inspecting pipelines. The SOP outlines necessary equipment, special precautions, and step-by-step instructions for conducting PIG operations, emphasizing the importance of maintaining pipeline integrity and preventing blockages.

Company A's SOP is a simple guide, mainly comprised of short, simple and clear directives. Despite this the ambiguity tool identified several potential ambiguities.

The lexical ambiguity rule identified multiple sections containing industry-specific terminology used, such as "*pig*", "*pigging*", "*Christmas (tree)*" and "*scale-deposits*." Out of the industrial context how these terms have been used could be potentially misleading or confusing. However, this demonstrates a limitation of the current version of the tool when handling industry and domain specific vocabulary.

The SOP showed no examples of syntactic ambiguity, as expected given its short and simple directives. The longest directive, at 42 words, is split into three sentences, reducing the likelihood of encountering syntactic ambiguity.

However, the tool did flag sections containing temporal and quantity ambiguities. For example, the terms "*periodically*" and "*enough space*" are used, these require judgment calls based on experience, which can lead to inconsistent operation, maintenance and assessments. To improve clarity, these can be easily addressed, "*periodically*" can be replaced with a precise frequency dependent on the requirements, and "*enough*" should be quantified or further explanation of how to determine sufficiency should be provided.

Additionally, the conditional ambiguity rule identified three cases of potential ambiguity. For example, "*If abnormalities not described above occur, contact...*", as the main condition "*if*"

contains a nested condition, "*not described...*" however in this case it adds clarity and precision. Whereas, another example identified, "*If there..., wet ... water, avoiding ... and ...*" is ambiguous. It is not clear whether the main condition is linked to either the verb "*wet*" or the entire sentence. It could be interpreted as modifying either the verb or serving as a separate instruction. This can be resolved by explicitly linking the conditional phrase to the verb or separating it as a distinct instruction.

The scope ambiguity and units of measurement rules did not identify any potential issues. However, there were several abbreviations/acronyms identified that were not defined within the SOP. Besides the company name and the term "*PIG*," there are six other abbreviations/acronyms undefined. While some may be well-known within the industry, it remains best practice to provide definitions of abbreviations/acronyms in SOPs to enhance understanding and ease of use. Additionally, included these can better facilitate translations if the SOP needs to be used in countries where languages other than English are spoken.

Addressing the highlighted ambiguities would further improve the quality and clarity of the SOP, reducing potential misunderstandings and inefficiencies, and minimizing potential risks.

The radar chart, Figure 5.6, provides a proportional visualization of the ambiguity types identified in Company A's SOP, based on the total of observed instances of potential ambiguity. Each axis represents a category of ambiguity, and the values are scaled to show their relative frequency. Lexical ambiguity stands out as the most common, accounting for nearly 30% of the total, reflects the expected use of domain specific language. The presence of temporal and quantity ambiguities suggests areas where operational timing and material thresholds are left open to interpretation, relying on user experience, potentially affecting consistency. In contrast, categories like syntactic and scope ambiguity appear minimal, which aligns with an SOP's straightforward, directive-based structure. The chart emphasizes that while the document is structurally sound, domain language and vague descriptors remain key sources of interpretive risk.

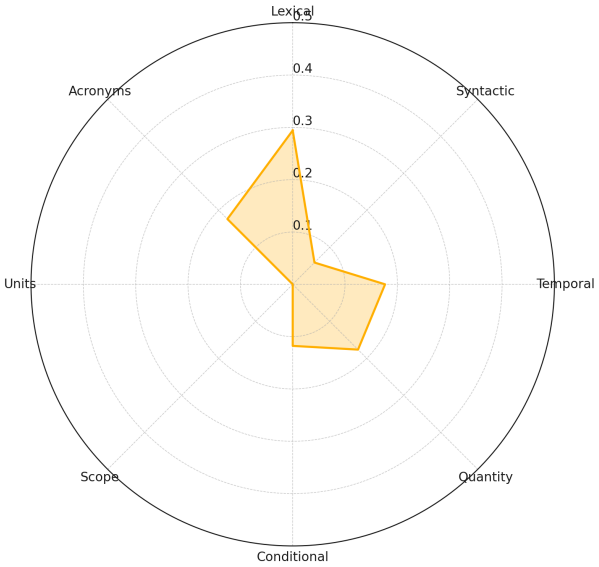

Figure 5.6: Proportional distribution of ambiguity types in Company A's SOP.

The *Violation Trigger tool* identified one critical section of the SOP, "*Open ... carefully, if possible ... professional, to check ... radioactive material and remove ...*" This step must be followed carefully and correctly due to the significant risks associated with exposure to radiation and the potential contamination risks. The tool's identification of this section is likely due to the use of terms such as "*check*", "*radioactive*" and "*carefully.*" The importance of adhering to this step is well underscored and does not necessitate any revisions. However, it would be beneficial to analyze the performance influencing factors of this specific task (e.g. human-machine interface), and emphasize this step in any training related to the process to ensure it is executed safely and meticulously.

5.4.2 Case Study Two - Biocide Storage

Company B's SOP, which spans 20 pages, details the procedures for applying biocide to storage tanks to prevent the growth of sulfate-reducing bacteria. These bacteria can produce hydrogen-sulfide, posing significant health and safety risks. This document is considerably more complex than the first example, incorporating detailed guidelines for the use of biocides, continuous monitoring procedures, and the necessary steps emergency response. It reflects a comprehensive

approach to managing environmental and safety risks associated with biofouling and chemical treatments in storage tanks used in offshore oil and gas operations.

Company B's SOP was input to the ambiguity tool, and a few examples of lexical ambiguity were identified, terms such as "*interface*", "*blanket*" and "*shock*" have domain specific meanings that do not line up with the most common definitions. These examples could be potentially confusing to those without as much industrial experience, and providing some additional definition or explanation may be helpful. However, these also again highlight the challenge of handling industrial vocabulary using the current approach.

Despite being overall well-written, precise, and easy to follow, there are a couple of examples of ambiguous and vague terms that were flagged by the tool related to quantity and timing, such as "*extended time*" and "*ample*." The phrase "*extended time*" is subject to individual interpretation and could be replaced with a more definitive duration. Similarly, the term "*ample*" necessitates a judgment call and should generally be avoided in procedural descriptions. It would be better replaced with a precise measurement or a method to determine the correct amount.

The phrase "*Where ... present, and thus ...* is required, no further action is required" triggered the scope ambiguity rule. The syntax does not clarify alone what the scope of "*no further action*" refers to as it lacks a direct object specifying the exact action or process to which this inaction applies. The standalone directive is correctly identified as potentially ambiguous, however within the context of the entire guide clarity is provided.

The use of conditional conjunctions in the document is frequent, yet they are straightforward and lack nested conditions or disjunctions. Overall, the SOP is consistent with its use of measurement units. The only inconsistency noted was the use of both "*ppm*" and "*mg/L*" for concentration measurements. This is not necessarily problematic, as specific units may be required when measuring and quantifying certain substances, particularly if these are the units shown at the specific human-machine interfaces described in the task. However, including a conversion table within the SOP may be useful. Common industry specific rules related to units of measurement could also be added to the algorithm to improve the tools efficiency.

The SOP helpfully includes a table of abbreviations and acronyms at the beginning, which defines most of these terms. However, the tool identified four terms that are not defined anywhere in the SOP. These could be easily added to the initial table to enhance clarity.

Figure 5.7 illustrates the relative distribution of ambiguity types found in Company B's SOP, highlighting areas where clarity could be improved. The most prominent issue is the presence of undefined acronyms, despite an existing glossary. Lexical ambiguities are also notable, largely

due to specialized terms that may confuse non-expert readers. While temporal, quantity, scope, and unit ambiguities occur less frequently, they still suggest minor areas of vagueness that could impact operational consistency. The absence of syntactic and conditional ambiguities reflects the SOP's otherwise clear and structured language. Figure 5.7

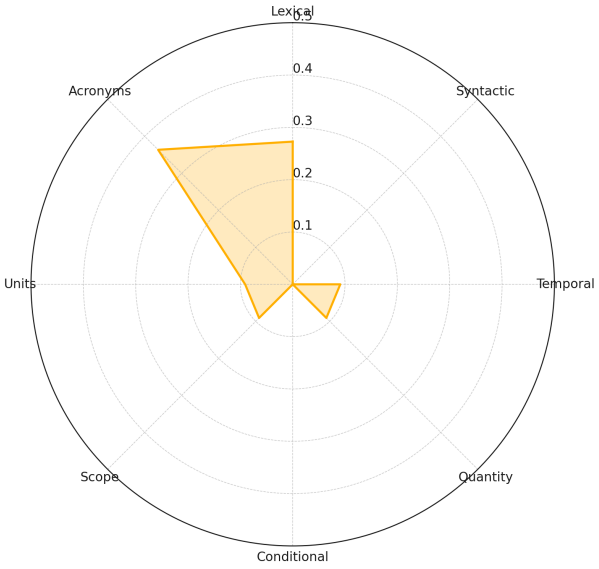

Figure 5.7: Proportional distribution of ambiguity types in Company B's SOP.

The radar charts for Company A (Figure 5.6)and Company B's (Figure 5.6) SOPs highlight both similarities and key differences. Both documents show notable lexical ambiguity, stemming from domain-specific terms. However, Company A's SOP exhibits a broader spread of ambiguity types, including a greater proportion of temporal, quantity, and conditional ambiguities, reflecting more general and imprecise phrasing. In contrast, Company B's ambiguities are more concentrated, with undefined acronyms being the most prominent issue, and minimal syntactic or conditional ambiguity. This may suggest that Company A's SOP is simpler but less precise, while Company B's is more technical and structurally consistent.

As expected, due to the increased length and detail in this SOP, the violation tool identified a higher number of cases, eleven in total. Of these, only one was an administrative section that can be disregarded. The remaining ten cases pertain to critical operational tasks, which include one or more of the following: simultaneous operations, overrides, fire and gas detection, and safety instrumented function testing.

Simultaneous operations refer to different activities occurring at the same time that may interfere with each other, potentially leading to complications. Without careful management,

simultaneous activities can create unsafe working conditions, increasing the risk of accidents due to conflict or hazardous task interactions. The complexity of simultaneous operations can exacerbate the consequences of even minor errors, potentially leading to catastrophic outcomes. Overrides, such as those involving start-up/shutdown sequence overrides, timer settings, and critical instrumentation, need control. Uncontrolled overrides may cause excessive stress on equipment, leading to premature failure or malfunctions. If critical system functions like pressure or temperature control are overridden without strict controls, it could lead to failures that escalate into high-risk potential scenarios. Ultimately, an override might be bypassing a safety barrier function defined by the designers (CCPS 2020).

The primary function of fire and gas detection systems is to identify hazardous conditions early before they develop into more severe incidents. Non-operational systems cannot provide these early warnings. Without functioning detection systems, there may be a delay in emergency response, allowing a fire or gas release to grow uncontrolled (Cowdrey 2023).

Similarly, safety instruments are crucial for detecting deviations and returning processes to a safe state. Failing to these systems can lead to inadequate safety responses when they are most needed. Without reliable safety instruments, equipment and systems become more susceptible to conditions that can lead to catastrophic failures, potentially resulting in, environmental damage, significant operational downtime, and high-risk situations for operators.

Common vocabulary present in the identified procedural steps falls into three main categories, operational terms, safety and compliance terms and technical components. Operational terms such as "*start-up*", "*check*", "*commissioning*", "*shutdown*", "*safety*", "*detection*", and "*instrumentation*" are foundational to understanding the functional aspects of the processes being analyzed. These terms help the tool in pinpointing procedural stages where precise actions are necessary to ensure the smooth running of operations. For instance, during the "*start-up*" or "*shutdown*" phases, the precise execution of steps is imperative to avoid operational failures or safety breaches.

Similarly, the presence of terms such as "*compliance*", "*safety*", "*risk*", and "*management*", are usually included in steps where there is an emphasis on safety and compliance, therefore strict adherence is essential.

The category of technical components, featuring terms such as "*integrated*", "*overrides*", "*fire and gas*", and "*commissioning*", focuses on the technical equipment and safety systems integral to industrial operations. The tool outputs a positive response, presence of a high-potential violation trigger, as it has learnt that these terms underscore steps that interact with or impact

high risk systems.

The *Violation Trigger tool* can play a pivotal role in enhancing safety and compliance within industrial operations. By effectively highlighting procedural steps that necessitate focused review, the tool aids in identifying where safety breaches are most likely and where high-risk operational failures could occur.

5.5 Limitations and Future Work

The development and implementation of the tools, designed to identify potentially ambiguous directives and potential high-risk violation triggers within SOPs, have demonstrated significant promise in enhancing the safety and compliance of industrial operations. However, as with any emerging technology, there are limitations and opportunities for further refinement. This section outlines the prospective avenues for future enhancements and addresses the current constraints of our tools. Through this discussion, a roadmap for evolving these tools into more robust, automated, and universally applicable systems that significantly contribute to industrial safety protocols is provided.

The first significant development is that of an integrated web version of these tools which will significantly increase their accessibility and ease of use. This platform would allow users to upload documents directly, interact with the tool's outputs, and contribute to the ongoing refinement of the tools through feedback.

The effectiveness of both tools is significantly enhanced by the manual preprocessing of the input SOPs. Current automated preprocessing attempts to extract the sections of interest, and remove formatting, page numbers, and other non-relevant text. Improving the algorithms that handle these tasks could further streamline the analysis process and output quality. Additionally, encouraging a uniform layout would simplify the task and effectiveness of such algorithms. Adopting a standardized document format for SOPs is strongly recommended, as it would significantly reduce common sources of ambiguity, particularly those related to inconsistent syntax and undefined abbreviations. A structured format that includes mandatory sections such as a definitions table, units of measurement, step-by-step actions, and responsible roles can help ensure that critical information is consistently captured and clearly presented. For instance,

requiring a dedicated section for abbreviations and acronyms would help prevent misinterpretation by ensuring that all technical terms are explicitly defined. Moreover, enforcing consistent linguistic patterns and formatting across SOPs would facilitate automated analysis, enhance readability, and support training and compliance efforts across different teams and departments. As such, standardization is not only a means of reducing ambiguity but also a practical step toward improving procedural clarity, safety performance, and communication in high-risk environments.

5.5.1 Ambiguity Tool Enhancements and Limitations

The current ambiguity tool does not significantly utilize ML due to data limitations. Future work should explore the development of an ML-based approach to ambiguity detection. Such an approach could leverage context from the entire document to enhance rule-based analyses, particularly for addressing scope and conditional ambiguities. Training of an ML model is dependent on the creation of a sufficiently large dataset of procedural texts with labeled ambiguities.

To facilitate the development of an ML model, this work proposes an effort to gather procedural steps with ambiguities, via the website hosting the integrated tools. This system would allow users to submit examples of ambiguous procedural language, which would be invaluable for training and refining ML algorithms. This collected data could also be used to refine and test the current rules-based approach.

Additionally, enhancing the ambiguity tool, particularly in the areas of lexical ambiguities, abbreviations/acronyms, and units of measurement, by incorporating industry and domain-specific vocabularies, definitions, and rules could significantly improve the tool's performance. Special attention to these elements is crucial, as it directly impacts the tool's effectiveness. Creating a dynamic database of industry-specific terms and commonly used phrases would further refine the tool's capability to identify ambiguities effectively.

Another potential avenue of development draws on the research in (Ferson et al. 2015). This study explores how common natural language terms, referred to as numerical hedges or approximators, such as "*about*", "*around*", and "*at least*", impact the interpretation of numerical data.

It empirically measured the quantitative implications these words have on the perceived uncertainty of numbers and offers specific numerical bounds for these interpretations. The ambiguity tool could be enhanced to suggest specific bounds and ranges in place of vague quantitative and temporal terms. This improvement would help address a common source of ambiguity in SOPs.

5.5.2 Developments for the High-Potential Violation Trigger Identification Tool

The *Violation Trigger tool* is trained solely on past incident reports, rather than its target domain of SOPs. This poses some limitations, primarily due to the nature and scope of the documents. While focusing on negative and known outcomes helps in risk management, it restricts the tool's exposure to a variety of procedural contexts that have not resulted in accidents but may still contain critical safety insights. This narrow focus might hinder the tool's ability to fully identify potential safety issues. Incident reports may also not include all the technical or industry-specific terminology that is present in SOPs. If the tool lacks exposure to this broader vocabulary, it might fail to recognize or properly interpret important steps or instructions when analyzing other procedural documents, leading to inaccurate assessments of risk.

The tool has been mainly trained on reports from the oil and gas industry, so it would likely perform best on SOPs from this sector, where it has greater exposure to the specific industry vocabulary. Broader training is necessary to improve its usability in other sectors.

Further manual analysis of SOPs could help improve the automated tool, providing a comprehensive dataset of procedure steps or vocabulary that are typically associated with high-risk potential could be incorporated into the model's training process. This would provide deeper insights into the nuances of procedural language that the systems trained only on incident reports. Additionally, expanding the dataset would not only improve training but also enable better evaluation of the current model's performance, ensuring that its assessments align with real-world safety risks across various industries.

Much of the training data currently comes from the oil and gas sector, this may lead to biases based on the predominant data sources which could affect their performance in other sectors. Collecting and analyzing documents from other industries would improve the tool's performance. This diversity in data sources would allow the tool to learn a broader range of operational

contexts, terminologies, and safety standards, enhancing the tool's versatility and accuracy.

A potential avenue for future research is developing a structured dataset that links accident reports to the specific SOPs involved in incidents. This would enable the training of targeted models to predict SOP violations and their consequences. However, industry participation is essential, as access to SOPs and expert insights are crucial for accurately mapping procedural steps to historical incidents.

Leveraging existing datasets like MATA-D and the IOGP database could be a starting point, but acquiring SOPs remains a challenge since they are often proprietary. Collaboration with industry stakeholders would be necessary to access these documents and ensure accurate alignment between procedural directives and reported violations. Industry experts would play a key role in verifying these mappings, given their contextual understanding of SOP applications in accident scenarios.

With a robust dataset, ML models could assess the risk of noncompliance in procedural directives, enabling proactive identification of high-risk SOP steps and enhancing safety measures in complex operational environments.

Beyond additional data collection, further enhancements to the tool's performance and adaptability could be achieved by incorporating uncertainty-based filtering, where low-confidence classifications trigger human review, ensuring that OOD cases are appropriately flagged. This approach aligns with research on OOD detection, which suggests that confidence-based techniques improve reliability when models encounter distributional shifts (Liu et al. 2024). Additionally, contrastive learning methods could be employed to refine sentence embeddings for both incident reports and SOPs, enhancing the model's ability to differentiate between structurally distinct yet semantically related procedural texts (Cheng et al. 2023). By integrating these refinements, the tool can continue to improve its accuracy and reliability, while maintaining strong cross-industry generalization.

5.5.3 Limitations in Addressing Context-Dependent Risks

The developed tools effectively identify ambiguities in SOPs and flag procedural steps with high potential for violations, enhancing procedural clarity and compliance. However, they remain limited to static analysis, evaluating textual and structural elements without considering external operational goals, conflicting objectives, or situational constraints that influence real-world decision-making. While they can highlight steps where violations are hazardous, they do not assess why or when these violations occur, such as due to time pressure, competing priorities, or systemic workarounds developed by operators.

Procedural violations often arise as adaptations to operational constraints rather than deliberate non-compliance. In high-risk industries, workers frequently develop workarounds to balance efficiency and safety when strict procedural adherence is impractical (Steen et al. 2024). These adaptations are shaped by misleading procedural guidance, conflicting operational goals, and cognitive workload, underscoring the need for safety assessments that consider these complexities rather than relying solely on procedural compliance (Podofillini et al. 2021). Since the current tools do not evaluate these factors, they may overlook key contributors to risk escalation.

Normalization of deviance further embeds violations into operational practices. When procedural shortcuts do not lead to immediate negative outcomes, they may become standard practice, increasing systemic failure risks (Sedlar et al. 2023). While the violation detection tool identifies critical steps requiring strict compliance, it does not analyze the underlying drivers of these violations. Incorporating scenario-specific analysis would provide deeper insight into these risks.

A systems-thinking approach is essential, as procedural compliance alone is insufficient for managing risk. Effective safety management must account for interactions between procedural guidance, organizational culture, and external constraints (Leveson 2011). Without these considerations, safety interventions may address only symptoms rather than root causes. While the current tools enhance procedural correctness by improving clarity and identifying critical steps, they do not dynamically assess real-world complexities.

Scenario-specific analysis can be approached through systematic methodologies such as HRA Bell and Holroyd (2009) or safety-critical task analysis Institute (2020). These methodologies assess critical tasks within procedures and consider key contextual factors (PSFs). This chapter addresses specifically the PSF often known as procedure quality. However, a comprehensive HRA would involve evaluating additional factors, including human-machine interface, physical

layout, and training.

Future work could integrate these tools with simulation-based risk assessments or decision-support systems that account for situational constraints. Scenario modeling would enable a broader understanding of how SOPs function in practice. Refining the violation identification model with domain-specific incident data would improve its ability to recognize contextual drivers of violations. Additionally, developing a framework for assessing procedural vulnerabilities under different operational conditions could enhance risk mitigation strategies.

The proposed enhancements, including the creation of an integrated web platform and improvements to preprocessing algorithms, along with a move towards standardizing document formats, are likely to significantly boost the usability and effectiveness of these tools. Furthermore, the shift towards a ML-based approach for ambiguity detection and the expansion of training datasets for both tools underscore a commitment to technological evolution and precision.

To further enhance their impact, future developments should integrate a systems-thinking approach to procedural risk assessment. This includes exploring methods to incorporate scenario-specific analysis and identifying contextual factors.

By addressing existing limitations and integrating user feedback into continuous development cycles, these tools are set to become more robust, adaptable, and essential components of SOP evaluation and HRA assessments, with an improved ability to anticipate and mitigate procedural risks in high-risk environments.

As such, the tools are not intended to promote hasty procedural changes but to serve as analytical aids that enhance the ability of experts to make informed, carefully considered decisions. By augmenting human judgment, they help ensure that any modifications to procedures are based on a deep understanding of operational realities.

5.6 Conclusion

This work presents the development and application of advanced language tools designed to enhance the quality of SOPs by identifying potential ambiguities and high-risk potential steps if violated. The tools, employing a combination of rule-based, NLP and ML techniques, represent a significant advancement in the field of safety within high-risk industries.

The value of these tools is multifaceted. Firstly, they serve as critical aids for those involved in the creation and review of SOPs, offering a means to ensure clarity and prevent the oversight of potentially ambiguous or high-risk content. This is particularly crucial in industries where the precise execution of procedures can be the difference between safe operations and catastrophic failures. The ability of the tool to efficiently scan and flag areas of concern in both existing and newly written SOPs not only saves valuable time but also significantly enhances procedural understanding and safety. Automating these processes allows the time spent reviewing the SOPs to be allocated to improving and refining them.

Secondly, the historical and pre-existing SOPs, which often remain unchecked due to their voluminous nature, can now be revisited with these tools. This allows organizations to update and refine their operational documents without the daunting task of manually reviewing each one, thereby maintaining a current and highly relevant body of procedural documentation.

Additionally, the tool supports human reliability analysts as it helps to assess the quality of SOPs, an important PSF for many HRA techniques.

In summary, the integration of these NLP tools into the review processes of SOPs is not merely an enhancement of compliance measures but a transformative move towards a more proactive and informed approach to safety management. Their ability to identify high-potential violations and ambiguities ensures that all personnel are well-equipped to perform their duties safely and effectively, ultimately safeguarding human lives and environmental health.

The ongoing proposed refinement of these tools, coupled with broader adoption and integration into existing safety frameworks, will make a substantial contribution to industrial safety management. The implications for future advancements in this technology are extensive and represent a significant step forward in the evolution of procedural safety and efficiency.

Chapter 6

Technical Development and Implementation

This chapter presents the technical development and implementation of a collection of computational, data-driven, and ML tools specifically designed to address a range of challenges within the HRA domain, as introduced in the preceding chapters (plus an additional tool not yet discussed).

The technical development of each tool has been kept separate and mostly confined to this chapter for several reasons. Firstly, the preceding chapters have been designed to focus on the idea, motivation, and practical use and testing of the tools, targeting users and those interested in the application of ML, NLP, and data in HRA. This ensures that the practical implications and benefits of the tools can be understood without being overwhelmed by technical details. Secondly, a dedicated chapter for technical aspects allows for a deeper exploration of the workings of ML and NLP, and how they have been integrated, catering to readers interested in the technical side. Including these details earlier could be confusing and distract from the focus on the use and benefits of the tools.

The tools have a wide range of goals, from supporting data gathering, enhancing explainability, learning from past accidents, constructing data-driven models and evaluating and improving procedural safety.

The core tools discussed in this chapter are:

- *HF Classifier(s)*: These tools are designed to automatically analyze accident reports and classify the contributing factors, as presented in Chapter 3 (Machine Learning Approach to Automated Human Reliability Data Collection) and Section 4.2 (Classification Tool) (Morais et al. 2022a), Johnson et al. 2023b)
- *Human-Centric Summarizer*: Focused on delivering concise, human role-centric summaries of incidents, this tool helps safety professionals quickly grasp the critical aspects of events without wading through extensive documentation, introduced in Section 4.3 (Summarization Tool) (Johnson et al. 2023b
- *Human Factors Causal Relationships tool*: This, not yet discussed, tool identifies the dependency relationships between contributing factors and integrates expert knowledge to construct Human Error (HE) models (Johnson et al. 2022, Johnson et al. 2023a).
- *Violation Trigger tool*: This tool identifies directives within procedural guides that when violated pose a high-risk potential, presented in Section 5.3 (High-Potential Violation Trigger Identification Tool) (Johnson et al. 2024).
- *Ambiguity Identifier*: Designed to detect various types of ambiguities and misleading steps within procedure guides, this tool ensures that safety protocols are clear and unambiguous, introduced in Section 5.2 (Ambiguity Identification Tool) (Johnson et al. 2024).

This chapter will detail the technical aspects of these tools, discussing their development, underlying algorithms, and software implementation. This chapter is essential as it further explains how the tools work to build trust and encourage their use, provides a foundation for future developments, and highlights how the technology is and can be utilized.

6.1 Virtual Human Factors Classifier(s)

In the Chapter 3 (Machine Learning Approach to Automated Human Reliability Data Collection), the need for and development of the *Human Factors Classifiers* was discussed. The first generation utilized SVM and BoW objects, and its implementation is detailed within Chapter 3.

6.1.1 Second Generation - BERT

Following this the second-generation tool (*HF Classifier 2.0*) that leveraged BERT was introduced and presented in Chapter 4, Section 4.2 (Classification Tool).
This section will now provide a more detailed explanation of how BERT works, how it has been leveraged, and how each step and process within the *HF Classifier 2.0* works.

6.1.1.1 Data Preparation

The first step in developing the second-generation *HF Classifier 2.0* involved preparing the textual data from accident reports. The available accident reports are mostly "text PDFs", these are PDFs that were created from digital sources, where the text is easily extractable. However, some are "image PDFs", these are essentially digital images of physical documents, typically created by scanning printed or handwritten materials. The text in these documents is not as simply extracted as the content is represented as an image. For simplicity, all reports were first converted into plain text files (.txt).
To convert text PDFs to plain text, the "PyPDF2" library in Python was utilized, which is capable of extracting text directly from PDF files created from digital source (Fenniak et al. 2022). Each report was iteratively converted to plain text. First, the PDF file was opened in binary read mode ("`rb`"), which means the file is read as a binary file. This is necessary for handling the PDF file format, which is a binary format containing various data types beyond just plain text. A "PdfReader" object is created to handle the PDF file. The script iterates through each page of the PDF, extracting text using the "extract_text" method of the "PdfReader" object. The extracted text from each page is accumulated in a string variable, and stored in a plain text file. The text file is opened in write mode with "UTF-8" encoding to ensure compatibility with various characters, ensuring that the text content is properly saved.
After this initial conversion, the script checks if the resulting text file is empty or not. If the file is empty, this likely meant that the report was an "image PDF", for such files OCR is required (Sharma 2023).
First, utilizing the "pdf2image" library each page of the PDF is converted into an image (Belval

INCIDENT	INDUSTRY	LOCATION	YEAR	Wrong Time	*Other Factors...*
1	Upstream	Piper Alpha	1998	1	0|...

Table 6.1: MATA-D Example Extract

2023). Then the "pytesseract" library, a Python wrapper for Google's Tesseract-OCR Engine which is one of the most accurate and widely used OCR engines, is used to process the image files and extract the text content embedded within the images (Hoffstaetter 2023, Smith 2007). The OCR engine processes images to detect lines of text and distinguish individual characters. It converts the visual data into readable text using models trained to recognize and interpret character shapes.

The recognized text string is then written to a plain text file, again with "`UTF-8`" encoding. The plain text version of the reports is then saved and stored with the same name as the original PDF.

In addition to this, some modification to the MATA-D was required. The MATA-D in its original state is laid out as shown in Table 6.1. However, to remove the additional complexity of trying to automate the pairing of the accident report with the corresponding row, the "INCIDENT", "INDUSTRY", "LOCATION" and "YEAR" columns were removed and manually replaced with a column stating the report file name in the MATA-D file used for training the model.

6.1.1.2 Data Processing and Tokenization

The next step involved isolating specific sections of interest, such as "*recommendations*" and "*lessons learned*" within the report texts. This system relies on a dictionary of key terms that are most likely to indicate the start and end of these targeted sections, defined in the development of the first generation tool (Morais et al. 2022a).

The confidence scoring system operates by scanning the text for these predefined start and end target words. When these words are detected, the system marks the likely beginning and conclusion of a section. The segments of text enclosed between these markers are then extracted as the content of interest. This approach allows for targeted extraction based on the contextual appearance of specific words and phrases identified during the development of our dictionary.

For instance, a report might contain a sentence like "The following recommendations are pro-

posed based on the investigation findings." The keyword "recommendations" would trigger the system to mark this as the potential start of the targeted section. Similarly, phrases such as "In conclusion" or "End of recommendations" may indicate the endpoint, allowing for accurate extraction of the intervening text.

This method ensures that the extracted sections are those most relevant to the objectives of the analysis, improving the precision of the data fed into the model. If the system identifies these key sections within a report, the extracted text replaces the complete report, if not, the original report remains unaltered.

Having extracted the contents of interest from the original reports, the text now needs to be prepared for input into the model. Tokenization is a foundational preprocessing step in NLP, it is essential for transforming text into a format suitable for model training and inference. This process involves breaking down text into smaller, manageable units called tokens, which can be words, subwords, or characters (Jurafsky and Martin 2008).

Consider the simple sentence "Pilot miscommunication led to the incident." The tokenizer might split this into tokens: `['Pilot', 'mis', 'communication', 'led', 'to', 'the','incident', '.']`, where "communication" is a subword token generated using the "WordPiece" method.

The BERT model was pretrained using specific tokenization rules, the input data must be tokenized using the same methods and rules to ensure that the model processes the input correctly. The BERT tokenizer handles special tokens like "[CLS]", "[SEP]", and "[PAD]" which are integral to the model's functioning (Devlin et al. 2018). These tokens are used to indicate the beginning of a sequence, separation of segments, and padding of sequences to uniform length, respectively. These tokens have specific positions in the BERT vocabulary and are crucial for tasks like classification.

A typical input might look like: ['[CLS]', 'The', 'aircraft', 'experienced', 'engine', 'failure', '[SEP]', 'Crew', 'responded', 'with', 'emergency', 'procedures', '[SEP]', '[PAD]', ..., '[PAD]'], where "[PAD]" is repeated to fill up to the maximum token length.

The BERT tokenizer, which employs the "WordPiece" method, deconstructs text into meaningful units (Song et al. 2020). This approach helps in handling unfamiliar and OOV terms while maintaining a manageable vocabulary size, which is critical for model efficiency and performance.

Consider the rare term "autostartsequence," which may not exist in BERT's vocabulary. The "WordPiece" tokenizer would break this down into subword units such as `['auto', 'start', 'sequence']`. Even though the full word is unfamiliar, the tokenizer enables the model to infer

its meaning based on the known subcomponents. This mechanism allows the model to generalize better to technical or compound terms that were not seen during pretraining.

Each report text file is processed by the BERT tokenizer using the "`Transformers`" library from Hugging Face, and the tokenized texts are stored as objects within Python Face 2023c.

Due to BERT's architectural constraint, which limits inputs to 512 tokens, reports exceeding this token limit are segmented into smaller sections, each containing up to 512 tokens. To mitigate information and context loss, an overlap of 50 tokens is included at the beginning and end of each segment. For example, a report with 900 tokens would be split into two segments: Segment 1 (tokens 0–511) and Segment 2 (tokens 462–900), creating a 50-token overlap. This segmentation strategy allows each part of the text to be independently processed by the BERT model, accommodating the entire text without minimal data loss. Each segment is assigned the PSFs for the corresponding accident from the MATA-D.

This segmentation presents significant challenges, particularly for reports where target sections were not identified, leading to the creation of numerous segments. Different parts of the reports discuss a wide range of incident aspects, and each segment contains only a portion of the overall narrative. This results in a loss of broader context and can lead the model to associate certain phrases or words incorrectly with classifications. Manually assigning classifications could address this limitation, but it would be very time-consuming, but worth considering in further developments. Despite these challenges, the numerous other advantages of using a BERT-based approach make it a valuable tool.

6.1.1.3 Word Embedding

After tokenization, the text data undergoes a transformation process to convert the tokenized text into embeddings using BERT's pre-trained layers. These embeddings are high-dimensional vectors that encapsulate the semantic meanings of the tokens (Devlin et al. 2018).

An embedding is a dense vector representation of a token, mapping tokens to continuous vector spaces of lower dimensionality where semantically similar tokens have similar representations (Patil et al. 2023). Unlike one-hot encoding, which creates sparse vectors with high dimensionality, embeddings generated by BERT are high-dimensional vectors (typically 768 dimensions for BERT-base) that capture the contextual information of each token within the sentence

(Devlin et al. 2018). This process involves converting each tokenized input sequence into three types of embeddings: token embeddings, segment embeddings, which indicates which sentence a token belongs to, and position embeddings, representing the position of the token in the sequence. These embeddings are summed to produce the final embedding for each token, which is then passed through BERT's transformer layers to capture contextual information, as shown in Figure 6.1(Devlin et al. 2018). Consider the sentence "The drilling crew initiated a controlled shutdown after detecting abnormal pressure." In traditional one-hot encoding, the word "pressure" would be represented as an isolated binary vector, treating it as unrelated to other domain specific terms like "shutdown" or "blowout." However, BERT embeddings capture semantic relationships and contextual usage. In this sentence, BERT will represent "pressure" as a dense vector that reflects its relationship with operational terms like "drilling" and "shutdown." In another context, such as "pressure from regulatory bodies," the same word would receive a different vector because BERT adjusts the representation based on surrounding words.

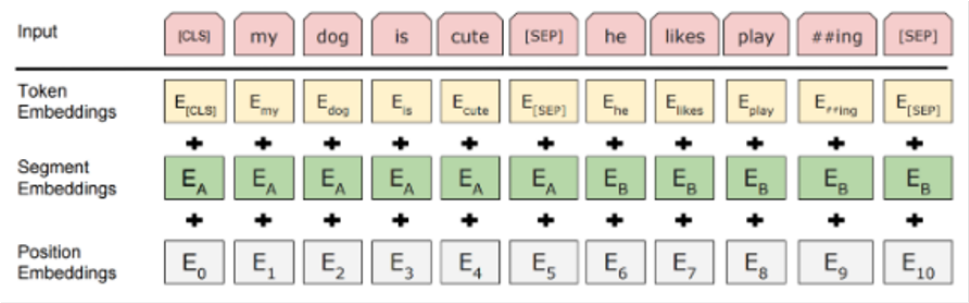

Figure 6.1: Input Word embeddings for BERT Model (Devlin et al. 2018)

6.1.1.4 Transformer Layers

Transformer layers are the core components of BERT and other transformer-based models, consisting of multiple layers of multi-head self-attention mechanisms and feed-forward neural networks (FNNs) (Vaswani et al. 2017). These layers process input sequences in parallel, enabling the model to capture long-range dependencies and contextual relationships more efficiently than sequential models like RNNs (Recurrent Neural Networks).

Self-attention mechanisms allow each token in the input sequence to focus on, or attend to, every other token in the sequence. This is achieved by computing a set of attention weights that quantify the importance of other tokens to the current token. The self-attention mechanism

computes three vectors for each token: the Query (Q), Key (K), and Value (V) vectors,

$$Q = KW^Q, K = XW^Q, V = XW^V \tag{6.1}$$

where, X is the input sequence, and W^Q, W^K, W^V are learned weight matrices (Vaswani et al. 2017).

These vectors are used to compute the attention scores and the weighted sum of the value vectors,

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}}) \tag{6.2}$$

where, d_k is the dimensionality of the Key vectors, and the softmax function ensures that the attention scores sum to 1 (Vaswani et al. 2017).

Multi-head attention involves applying the self-attention mechanism multiple times in parallel, each with different learned weight matrices (Vaswani et al. 2017). The results are concatenated and linearly transformed to produce the final output. The process can be expressed as,

$$MultiHead(Q,K,V) = Concat(head_1, ..., head_h)W^O \tag{6.3}$$

where,

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{6.4}$$

and W^O is a learned weight matric (Vaswani et al. 2017).

Each transformer layer also includes a position-wise FFNs applied to each position separately and identically, consisting of two linear transformations with a ReLU (Rectified Linear Unit) activation in between,

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{6.5}$$

where, W_1, W_2, b_1, b_2 are learned parameters (Shen et al. 2023).

Consider a report sentence from an offshore incident log: “After a sudden drop in wellhead pressure, the drilling supervisor ordered an immediate evacuation due to suspected gas influx.”

Using the self-attention mechanism, BERT can learn that “gas influx” is semantically and causally linked to “sudden drop in wellhead pressure” and “evacuation.” Even though these phrases

are several tokens apart, self-attention allows the token "evacuation" to attend more strongly to "gas influx" and "pressure" than to unrelated parts of the sentence. This captures critical cause-effect relationships, which traditional RNNs would struggle to retain over longer sequences.

In the multi-head attention mechanism, different heads may focus on distinct aspects of the sentence, one head might focus on technical terminology ("wellhead," "pressure," "gas influx"), another on action-oriented language ("ordered," "evacuation"), and yet another on timing or sequence ("after," "immediate"). When these diverse perspectives are concatenated, the model builds a multidimensional understanding of the situation.

This capacity to understand complex, interdependent relationships between technical terms, actions, and causes is especially valuable in this project, where precise interpretation of events is essential for effective classification.

6.1.1.5 Model Configuration

To construct the *HF Classifier 2.0* model the pre-trained BERT model is fine-tuned for each factor's classification task, this involves adding a classification layer on top of BERT to handle the binary classification of individual factors. This additional layer typically consists of a dense (fully connected) layer followed by a sigmoid activation function to output probabilities for each class (Qasim et al. 2022). The classification function is represented as,

$$p = \sigma(W_h + b) \tag{6.6}$$

where, p represents the vector of predicted probabilities for each factor, W and b are the weights and biases of the classification layer, h is the hidden state from BERT (the output of the last transformer layer) and σ is the sigmoid functions, which maps logits to probabilities (Devlin et al. 2018).

The implementation leverages "`TFBertForSequenceClassification`" from the "`Transformers`" library (Face 2023c).

6.1.1.6 Loss Function and Optimization

In the fine-tuning process, the binary cross-entropy loss function measures the discrepancy between the predicted probabilities and the actual labels, making it suitable for binary classification tasks. The loss function is defined as,

$$L = -[y \log(p) + (1-y) \log(1-p)] \tag{6.7}$$

where, y is the actual label (0 or 1), p is the predicted probability for that instance (Goodfellow and Courville 2016).

The Adam optimizer is used to update the model parameters. Adam is an adaptive learning rate optimization algorithm that computes individual learning rates for different parameters based on estimates of the first and second moment s of the gradients (Kingma and Ba 2014). The parameter update rule is given by,

$$\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{v_t} + \varepsilon} \tag{6.8}$$

where, θ is the model parameters at step t, η is the learning rate, m_t and v_t are the first and second moment estimates of the gradients (mean and uncentred variance, respectively), and ε is a small constant to prevent division by zero (Kingma and Ba 2014).

If a report is labeled as involving "Wrong Time" ($y = 1$) but the model predicts a probability of 0.2, the binary cross-entropy loss penalizes this low confidence in the true class. The Adam optimizer then adjusts weights to reduce similar errors in future predictions.

The loss function and the optimizer are implemented using the "Tensorflow" library with the functions "`tf.keras.losses.SparseCategoricalCrossentropy`" and "`tf.keras.optimizers.Adam`" (TensorFlow 2015).

6.1.1.7 Training Procedure

The training procedure involves multiple epochs for each binary classification model, where each epoch consists of a forward pass, loss computation, backward pass, and parameter update. To accommodate long input sequences, each report is split into overlapping or non-overlapping chunks of up to 500 tokens, allowing the model to process the full content without truncation. During the forward pass, all chunks associated with a single report are independently passed through the BERT model. The resulting logits from each chunk are then averaged to produce a single, aggregated prediction for the entire report.

These aggregated logits are passed through a softmax activation function (for two-class classification with `num_labels=2`) to compute class probabilities (Qasim et al. 2022). The binary cross-entropy loss is then calculated between the predicted probabilities and the true label of the original report. This per-report loss is used for backpropagation, where the gradients of the loss with respect to the model parameters are computed using backpropagation (Goodfellow and Courville 2016).

The model parameters are updated using the Adam optimizer, which applies the computed gradients to adjust the model's weights and biases in the direction that minimizes the loss. This process is repeated for each batch of chunked inputs in the training set. An epoch is completed once all batches have been processed. Typically, multiple epochs are required for the model to converge, at which point performance stabilizes and loss is minimized.

By repeating this training process for each binary factor, an ensemble of fine-tuned BERT-based classification models is produced—one for each human factor. Collectively, these models comprise the *HF Classifier 2.0*, which can be used during inference to generate predictions for new, unseen incident reports. The evaluation of both the first- and second-generation classifiers is provided in Section 3.3 (Evaluating Performance) and Section 4.2.3 (Classification Tool Performance), respectively.

Further methodological details, supplementary code and explanation can be found in the Appendix, first-generation in Section A.1 and second-generation in Section A.2.

6.2 Human-Centric Summarizer

The *Human-Centric Summarizer* is a tool designed to summarize accident reports, focusing on the human role. It supports explainability of identified factors and provides stakeholders with clear, concise understanding of events, particularly for those not involved in detailed analysis. By highlighting human involvement, the summarizer aids timely, informed decision-making in safety management. It condenses lengthy and detailed accident reports into concise summaries, emphasizing critical human-related details. This automation saves time and resources, enhancing productivity in safety analysis and risk assessment.

The outline of its process, application, and evaluation is provided earlier in Section 4.3 (Summarization Tool). The technical process of the tool involves extracting relevant sections and sentences, followed by using an abstractive summarization approach to condense and structure the information. This section will detail the tool's processes and explain how BART is leveraged to achieve high-quality summarization.

6.2.1 Information Extraction

Similar to the*HF Classifiers*, the *Human-Centric Summarizer* requires the accident report to be in a PDF format. The tool therefore also uses the same process to convert the PDF to raw text (.txt), whether the file is a "text PDF" or an "image PDF".

From here an extractive summarization process is employed to identify and extract specific sections and sentences of interest from the raw text. Part of this uses the same algorithms as the *HF Classifiers* to identify such as the "*recommendations*" and "*lessons learned*".

Alongside these sections the algorithm defines a set of keywords to identify specific sentences that may be of interest. These include pronouns such as "he", "she", "they", and a comprehensive list of personnel and job roles such as "operator", "user", "engineer". Using regular expression libraries "`re.split`" function, the text is split into sentences, and sentences containing any of the keywords are extracted. These sentences are then combined with the extracted sections of interest and saved (Foundation 2023).

In a nuclear incident report, the algorithm might extract the section titled "Lessons Learned,"

which contains statements such as “Operators failed to follow reactor cooldown procedures.” It also captures individual sentences like “The technician did not verify the valve status,” where keywords like “technician” and pronouns such as “he” signal relevance to human performance issues.

6.2.2 Abstractive Summarization

Traditional purely extractive summarization techniques, which select and concatenate sentences directly from the source text, may fail to produce coherent and logically structured summaries (Giarelis and Karacapilidis 2023). Therefore, the addition of an abstractive summarization approach, which involves generating novel sentences that convey the essence of the original text, is preferred. The approach here utilizes another popular transformer based LLM, known as BART designed for sequence-to-sequence tasks (Lewis et al. 2019).

Instead of directly quoting “The operator missed a step in the reactor startup checklist, which led to pressure buildup,” the BART model might generate an abstractive summary such as: “Failure to follow procedures in startup resulted in operational risk.” This captures the core meaning in a more concise, generalized form for review.

6.2.3 BART

BART, developed by Facebook AI, is a sophisticated model used for various NLP tasks, including abstractive summarization (Lewis et al. 2019). Its architecture combines the strengths of two types of transformers: bidirectional encoders and autoregressive decoders. This unique dual-encoder-decoder architecture makes BART particularly effective for summarization tasks Zhang et al. 2022.

The bidirectional encoder in BART reads the entire input sequence bidirectionally, that means it processes the sequence from both beginning to the end (left-to-right) and the end to the beginning (right-to-left) simultaneously (Lewis et al. 2019). This bidirectional approach allows the encoder to capture the context of each word considering both its preceding and succeeding

words. This helps in building a comprehensive understanding of the input text.

The decoder generates the output sequences in an autoregressive manner, which means it predicts one token at a time from left-to-right. At each step, the decoder considers all previously generated tokens to predict the next token. The sequential dependency ensures that the generated text is coherent and logically flows from one word to the text.

The BART model is trained using a denoising autoencoder approach, where the original input sequence $X = (x_1, x_2, ..., x_n)$ is corrupted using a noising function, $q(X|X')$. This means that certain parts of the sequence are randomly masked, deleted, or shuffled to create a noisy version, X', of the original sequence (Lewis et al. 2019). The model is then trained to reconstruct the original sequence,X, by minimizing the reconstruction loss function,

$$L(X, X') = -\sum_{t=1}^{n} \log(P(x_t|X', \theta)) \tag{6.9}$$

where, $P(x_t|X', \theta)$ represents the probability of the model correctly predicting the t-th token, x_t, given the noisy input $X'$and the model parameters θ (Lewis et al. 2019). This loss function computes the negative log-likelihood of each token's prediction, and the summation aggregates this across the entire sequence. The goal is to adjust the model parameters θ to maximize the probability of accurately reconstructing the original sequence from the noisy input.

Given a sentence like "A pressure buildup in the secondary containment vessel was not noticed due to sensor calibration failure", the bidirectional encoder understands context like "pressure buildup" and "sensor failure," while the autoregressive decoder rewrites it coherently.

6.2.3.1 BART for Summarization

BART performs text summarization through its encoder-decoder architecture, effectively reducing the length of the input text by identifying and retaining key information. The input text that needs to be summarized is passed through BART's bidirectional encoder, the encoder processes the entire input sequence in both directions, capturing rich contextual information for each token by considering both its preceding and succeeding words. This helps the encoder build a comprehensive understanding of the text, including identifying important concepts, entities, and relationships.

The encoder outputs a set of contextual embeddings that represent the comprehensive understanding of the input text (Peters et al. 2018). These embeddings encapsulate the meaning and context of each token within the entire sequence, highlighting the critical information that needs to be conveyed in the summary.

The decoder then takes these contextual embeddings and begins generating the summary in an autoregressive manner (Lewis et al. 2019). It starts with a special start token and predicts the next token in the sequence one at a time. The decoder uses an attention mechanism to focus on different parts of the encoder's outputs, prioritizing the most relevant and important information from the input text (Vaswani et al. 2017). This selective attention helps the decoder emphasize tokens that contribute to the main idea or key points, while omitting less important details and redundant information.

Each token generated by the decoder is influenced by all previously generated tokens as well as the contextual embeddings from the encoder. This ongoing interaction ensures that the summary is coherent and logically flows from one word to the next. Previously trained to use more concise language to convey the same meaning, the decoder aims to replace long descriptive phrases with single words or shorter phrases that capture the essence of the original text (Lewis et al. 2019). This allows the model to construct a shorter sequence that still accurately represents the original text, maintaining coherence and focus throughout the summary.

6.2.3.2 Tokenization

As with BERT, before the extracted text can be input into BART, it must first be tokenized, which involves converting the text into a format that the model can process. BART uses a subword tokenizer, which is particularly efficient for handling rare words by breaking them into more frequent subwords (Sennrich et al. 2015). This approach ensures that even complex or infrequent words can be effectively processed by the model.

In this model "BartTokenizerFast" from the pre-trained BART model is used to tokenize the input text (Face 2023a). The tokenization process involves converting the text into a sequence of "token IDs" that the model can then process. BART employs Byte-Pair Encoding (BPE), a subword tokenization technique that iteratively merges the most frequent pairs of bytes in a given text corpus, creating a fixed vocabulary of subwords (Khanna 2021). The "BartTokenizerFast"

is an optimized version of the BPE tokenizer that allows it to handle a wide range of text inputs, ensuring that the tokenization and detokenization processes are efficient and aligned with the model's expectations (Face 2023a).

In this process, the input text is split into subword units using the BPE algorithm, and then these subwords are then converted into corresponding "token IDs" based on the pre-trained vocabulary of the BART model, forming numerical representations that the model can process. Special tokens such as "`[CLS]`" (classification token), "`[SEP]`" (separator token), and "`[PAD]`" (padding token) are added as needed to help the model understand the structure of the input and manage different text segments (Face 2023a). The tokenized output consists of input IDs, which are sequences of numerical IDs representing the text, and an attention mask, which indicates which tokens are actual input tokens (1) and which are padding (0), allowing the model to focus on the relevant parts of the input (Lewis et al. 2019).

6.2.3.3 Iterative Summarization

Now that the extracted sections from the accident report are tokenized, they can now be input into the BART model for summarization.

However, BART has an architectural constraint that limits the maximum input length to 1024 tokens. To manage this constraint the tokenized text is split into equal chunks, the number of chunks is determined by rounding the number of tokens up to the next thousand, and then dividing by 1000. Each chunk is then independently fed into the model to generate "summary IDs" which represent the tokens of the summarized text. The generated "summary IDs" are then decoded back into human-readable text using the tokenizer. Each "numerical ID" corresponds to a subword token in the tokenizer's vocabulary. The tokenizer uses the predefined mappings, that pairs each ID with a specific subword token, to convert the summary IDs into subwords (Khanna 2021).

These subwords then need to be combined to form complete words. During the training phase the tokenization learnt a set of rules to split the words, during detokenization BART follows the reverse of these rules to recombine the subwords (Lewis et al. 2019). The final step is to assemble these words into coherent sentences by adding spaces, proper punctuation, and capitalization based on convention.

The generated summaries for each chunk, are then combined and the length checked if the summary remains greater than 500 words, the process is repeated. Again splitting into chunks and summarizing until the desired length is achieved. This strategy ensures that long documents are adequately summarized, without completely discarding sections of the extracted text.
The final generated summary is then output as a raw text file, this can easily be combined with the *HF Classifier's* output and formatted into an official report to share with invested parties. By leveraging BARTs sophisticated transformer architecture, the approach provides coherent and concise summaries that are well-suited for practical safety investigation and management. Further methodological discussion and discussion on the code can be found in the Appendix, Section A.3.

6.2.3.4 ROUGE Metrics

To assess the *Human-Centric Summarizer*, the ROUGE metrics were selected over BLEU and METEOR due to their demonstrated effectiveness, specifically by measuring content overlap between original texts and their summaries more accurately (Ng 2015). ROUGE is primarily used to evaluate the quality of summaries by comparing them to one or more reference texts. The key variants of ROUGE include ROUGE-N, ROUGE-L and ROUGE-S.
ROUGE-N measures the overlap of n-grams—continuous sequences of n words—between the generated and reference texts (Lin 2004). Overlap refers to the number of n-grams that appear in both the generated summary and the reference summary. ROUGE-1 and ROUGE-2 are the most common variants, focusing on unigrams (single words) and bigrams (two-word sequences), respectively. ROUGE-N scores are usually reported as F1-scores, which combine precision (the proportion of generated n-grams that are also in the reference) and recall (the proportion of reference n-grams that are captured in the generated summary).

$$\text{Recall} = \frac{\text{Overlapping Number of n-grams}}{\text{Number of n-grams in reference text}} \tag{6.10}$$

$$\text{Precision} = \frac{\text{Overlapping Number of n-grams}}{\text{Number of n-grams in generated summary}} \tag{6.11}$$

where, "Overlapping Number of n-grams" is the count of the n-grams in both the reference and summary (Kızılırmak 2023). The F1-score, the harmonic mean of recall and precision is calculated as previously defined in Section 3.3.

ROUGE-L focuses on the LCS, LCS is the longest sequence of words that appear in both the summary and reference text, while keeping the order of the words intact, it is important to note that LCSs are not necessarily consecutive but still in order (Kızılırmak 2023). This metric is useful for assessing the fluency of the summary. To calculate the ROUGE-L first the LCS between the two texts is identified. Once this is identified recall and precision are calculated as,

$$\text{Recall} = \frac{LCS(S,R)}{|R|} \tag{6.12}$$

$$\text{Precision} = \frac{LCS(S,R)}{|S|} \tag{6.13}$$

where, $LCS(S,R)$ is the length of (number of words in) the LCS, $|R|$ is the length of (number of words in) the reference text, R, and $|S|$ is the length of (number of words in) the summary, S (Kızılırmak 2023). The ROUGE-L is again given by the F1-score.

Finally, ROUGE-S which measures skip-bigram co-occurrence statistics (Kızılırmak 2023). A skip-bigram is any pair of words in their sentence order, allowing for arbitrary gaps. For example, in the sentence "*Operators must report any equipment malfunction immediately*", skip-bigrams include adjacent pairs like ("*must*", "*report*") and ("*equipment*", "*malfunction*"), as well as non-adjacent pairs such as ("*operators*", "*equipment*") or ("*report*", "*immediately*"), as long as the original word order is maintained. This allows for the measurement of the semantic relationship between words that are not necessarily adjacent but are in close proximity. Recall and precision are calculated similarly to the previous version given as,

$$\text{Recall} = \frac{\text{Overlapping Number of skip-bigrams}}{\text{Number of skip-bigrams in reference text}} \tag{6.14}$$

$$\text{Precision} = \frac{\text{Overlapping Number of skip-bigrams}}{\text{Number of skip-bigrams in generated summary}} \tag{6.15}$$

where, "Overkapping Number of skip-bigrams" is the count of the skip-bigrams in both the reference and summary (Kızılırmak 2023). Allowing the F1-score to be calculated for the ROUGE-S metric.

The ROUGE metrics, as well as other assessments of the *Human-Centric Summarizer* are provided in Section 4.3.3 (Summarization Tool Performance).

6.3 Human Error Modeling – Bayesian Networks

With the gathered data and a comprehensive understanding of the events, effective modeling becomes crucial for accurate analysis and decision-making. BNs have been extensively utilized in HRA and HEP modeling (Mosleh and Groth 2012). A BN is a probabilistic graphical model that represents a set of variables and their conditional dependencies through a directed acyclic graph (DAG). In a BN, nodes represent variables or factors, and directed arcs between nodes indicate causal influences. The strength of these relationships is quantified through CPTs, which specify the likelihood of a variable given its parent variables (Kosko and Noble 2009).

BNs explicitly capture and visualize these interdependencies, enhancing the clarity and comprehensibility of the relationships between PSFs. This makes it easier for analysts and stakeholders to understand the causal links and dependencies within the model.

By incorporating conditional probabilities, BNs can be used to compute the HEP under different contextual conditions. They help explain why certain human errors occur by illustrating the interconnections and influences among various PSFs. Understanding these causal relationships allows for the identification of interventions that can mitigate the risk of human errors.

Despite the advantages of BNs, there is a notable reliance on expert opinion rather than empirical data for determining the structure of these models (Mosleh and Groth 2012). While empirical data is often used to estimate the conditional probabilities within the network, the identification of causal links and the overall structure are predominantly based on expert judgments. This reliance on expert opinion introduces potential biases, as experts may overlook or misinterpret causal relationships due to cognitive biases or limited perspectives (Nunes et al. 2018).

By not fully leveraging available data, analysts may miss critical insights into the true nature of dependencies among PSFs, which could affect the reliability and effectiveness of HRA and HEP assessments. Therefore, incorporating more data-driven approaches to structure learning could enhance the accuracy and credibility of BNs in human reliability and error modeling.

6.4 Human Factors Causal Relationships Tool

Rather than relying purely on expert opinion to construct the model, there are many different approaches to structure learning in use in other fields. There are three classes of algorithms typically used to learn the structure of BNs from data. These are constraint-based algorithms, which use conditional independence tests to learn the dependence structure of the data, score-based algorithms, which use goodness-of-fit scores as objective functions to maximize, and then hybrid algorithms that combine aspects of both approaches (Scutari et al. 2019). The *Human Factors Causal Relationships tool* (*HF Relationships tool*) has been developed to incorporate the strengths of both constraint and score based algorithms to identify the conditional dependencies between PSFs, and in turn build the structure of a HEP model (Johnson et al. 2022).

6.4.1 Constraint-Based Algorithms

Constraint-based algorithms rely on conditional independence tests to determine the structure of the network. These approaches use statistical tests to discover dependencies and independencies among the variables, which can then be used to construct the network structure (Steck 2001). The general approach followed by constraint-based algorithms is:

1. Start with a complete undirected graph, where all variables are connected to each other.
2. Perform a series of conditional independence tests to remove edges between variables that are conditionally independent given a subset of the other variables.

3. Orient the remaining undirected edges to form a DAG representing the BN structure, based on certain rules and patterns in the remaining edges (Steck 2001).

6.4.2 Score-Based Algorithms

Score-based algorithms search through the space of possible network structures and evaluate each candidate structure using a scoring function, which measures how well the structure fits the data (Scutari et al. 2019).

The general approach for score-based algorithms can be summarized as follows:

1. The first step is to choose a scoring function, a mathematical criterion used to evaluate how well a proposed network structure explains the observed data. This function typically balances goodness-of-fit with model complexity, penalizing overly complex networks to avoid overfitting (Scutari et al. 2019).
2. The next step is to search through the space of possible network structures to find the structure(s) that maximize the chosen scoring function. This search space can be incredibly large, even for a moderate number of variables, so efficient search strategies are required.
3. Various search strategies can be employed, including greedy search, hill-climbing, simulated annealing, and genetic algorithms. These strategies explore the search space by making local changes to the current network structure, such as adding, removing, or reversing edges.
4. For each candidate network structure encountered during the search, the scoring function is computed on the data to evaluate how well the structure fits the data.
5. The search continues until a termination criterion is met (e.g., a maximum number of iterations, a score threshold, or convergence). The algorithm then returns the network structure(s) with the highest score(s) as the learned BN structure (Bouchaala et al. 2010).

6.4.3 Chosen Algorithms

The *HF Relationships tool* leverages the advantages of both approaches, by integrating one constraint-based and one score-based algorithms into the tool's functionality. The chosen constraint-based algorithm is the NPC algorithm.

1. Initialization:
    - This algorithm requires an input dataset of variables and the values, and starts with a fully connect undirected graph, G, where each node represents a variable. Every variable is initially considered to be potentially connected to every other variable.
2. Iterative Conditional Independence Testing:
    - For each pair of variables X and Y, test for conditional independence given various subsets S of other variables.
    - Remove edges, if X and Y are found to be conditionally independent given S, ($X \perp Y|S$), remove the edge $X-Y$ from the graph, G.
    - Record S as the seperating set for X and Y
3. Edge Orientation Using PC-Stable Rules:
    - Apply the following rules to orient the remaining edges in the undirected graph:
        - If $X-Z-Y$ is identified, where X and Y are not directly connected, and Z is in the separating set of X and Y, then orient $X-Z-Y$ as $X \rightarrow Z \leftarrow Y$ to create v-structures.
        - For each directed edge, $X \rightarrow Y$, if there exists and undirected edge $Y-Z$ such that X and Z are not connected, orient $Y-Z$ as $Y \rightarrow Z$
        - For each pair of nodes $X \rightarrow Y$ and $Y \rightarrow Z$, if the edge $X-Z$ exists orient it as $X \rightarrow Z$ to prevent cycles.
    - terate these rules until all edges are oriented.
4. Once all the edges are oriented, the final DAG is the learned structure of the BN (GUI 2020).

For the score-based approach the K2 algorithm, a heuristic search approach, was chosen.

1. Initialization:

- This algorithm requires an input dataset, D, with m, cases, the set of n nodes, and an upper bound, u, on the number of parents a node can have.
- The algorithm also requires a predefined ordering of the nodes which can be either be randomly generated or determined using some other method. This initial ordering can have significant effect on the final network structure.
- Let π_i represent the parent set of the node X_i. The parent set is the set of parent nodes. A parent node of a given node, X_i, is any node, X_j that has direct influence on X_i. That is the directed edge $X_j \rightarrow X_i$ exists.

2. For each node, $i = 1,..,n$, the algorithm uses the scoring function $f(i,\pi_i)$, to evaluate how well the parent set explains the data for a given node,

$$f(i,\pi_i) = \prod_{j=1}^{q_i} \frac{(r_i-1)!}{(N_{ij}+r_i-1)!} \prod_{k=1}^{r_i} \alpha_{ijk}! \tag{6.16}$$

where, π_i is the set parents of node X_i

$q_i = |\phi_i|$

ϕ_i is the list of all possible instantations of the parents of X_i in the database, D. That is, if $P_1,...,P_s$ are the parents of X_i then ϕ_i is the Cartesian product, $\{v_1^{P_1},...,v_{r_{P_1}}^{P_1}\} \times ... \times \{v_1^{P_s},...,v_{r_{P_s}}^{P_s}\}$, of all possible values of the attributes P_1 through P_s.

$r_i = |V_i|$

$V_i|$ is the list of all possible values of the attribute X_i.

α_{ijk} is the number of cases in D in which the attribute X_i is instantiated with its k^{th} value, and the parents of X_i in π_i are instantiated with the j^{th} instantiation in ϕ_i.

$N_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$. The number of instances in D in which the parents of X_i in π_i are instantiated with the j^{th} instantiation of ϕ_i.

3. For each node iterate through the other nodes in the given order,
   - Let the current node be X_i.
   - Initialize the best parent set for X_i as empty, $\pi_i = \varnothing$.
   - Calculate the score for X_i with no parents.
   - Incrementally add parents to X_i from the set of nodes preceding X_i in the given order.
   - For each potential parent X_j (where $j < i$ according to the node order).
     - Temporarily add X_j to the parent set, π_i, of X_i.
     - Calculate the new score for X_i with this updated parent set.

- If the new score is better than the current best score,
  (a) Update the best parent set to include X_j,
  (b) Update the best score to the new score.
- If no improvement is found, move to the next potential parent.

- The best parent set for X_i is finalized once the addition of more parents does not improve the score or the upper bound, u, on the number of parents a node can have is reached.
- This process is repeated for each node in the predefined order until all nodes have their parent sets determined.

4. The determined parent sets are then used to construct the final BN structure (Cooper and Herskovits 1992).

To determine the initial node ordering for the K2 algorithm, a methodology based on information theory has been developed, based on ideas first presented by Benmohamed et al (Benmohamed et al. 2022).

1. The mutual information (MI), a measure of the amount of information one random variable contains about another, for each conditional relationship is calculated,

$$I(X;Y) = H(X) - H(X|Y) \tag{6.17}$$

where,
$H(X)$ is the entropy of X
$H(X|Y)$ is the conditional entropy of X given Y.
Entropy is a measure of the uncertainty or randomness in a random variable, given by,

$$H(X) = -\sum_{x \in X} P(x) \log P(x) \tag{6.18}$$

and,

$$H(X|Y) = -\sum_{y \in Y} P(y) \sum_{x \in X} P(x|y) \log P(x|y) \tag{6.19}$$

where,

$P(x)$ is the probability of of x, $P(y)$ of the probability of y, and $P(x|y)$ is the probability of x given y (Brownlee 2020).

2. Pairwise triangular structures are then tested. Nodes X and Y are tested with another node Z to check if they form a cycle. By the directivity property, the amount of MI between the input messages and the output messages is likely to become smaller once the exchanged message has gone through multi-level treating.

$$I(X;Y) \leq I(X;Z) + I(Z;Y) \tag{6.20}$$

After some manipulation it the following condition is obtained,

$$(I(X;Y) > I(X;Z)) \vee (I(X;Y) > I(Z;Y)) \tag{6.21}$$

If this is satisfied, it suggests that a dependency relationship between X and Y exists (Benmohamed et al. 2022).

3. A dependency matrix, D, is constructed based on the identified dependency relationships, with entries of 1if variables i and j are dependent and 0 otherwise.
4. As MI is symmetric, it cannot be determined whether X or Y is the parent in the relationship. Therefore, the conditional relative average entropy (CRAE) is calculated for each dependency relationship to determine which is the parent node. The CRAE measures how much uncertainty about X remains when Y is known, normalized by the entropy of X and the number of states of X,

$$CR(Y;X) = \frac{H(X|Y)}{H(X) \times |X|} \tag{6.22}$$

$$CR(X;Y) = \frac{H(Y|X)}{H(Y) \times |Y|} \tag{6.23}$$

where,

$|X|$ and $|Y|$ represent the number of possible states/values X and Y can take.

$CR(Y;X)$ measures the relative uncertainty of X when Y is known, and $CR(X;Y)$ measures the reverse. Therefore, if

$$CR(Y;X) < CR(X;Y) \tag{6.24}$$

it can be said that knowing Y reduces the uncertainty in X more than the reverse, therefore Y is likely the parent, and X is the child (Braverman 2011).

5. The order is compiled from the dependency matrix by evaluating the CRAE for each dependency relationship. This ensures that the order reflects the hierarchical structure of dependencies, starting from nodes with fewer dependencies (higher in the hierarchy) to nodes with more dependencies (lower in the hierarchy) (Benmohamed et al. 2022).

6.4.3.1 Integrating Expert Knowledge

To address any potential knowledge gaps in the learnt models due to data availability, the tool includes the option to include previously determined conditional relationships based on expert opinion in the model structure. The tool has been encoded with a conditional relationship table for the 53 included PSFs. This is based on a conditional relationship table originally proposed with the development of the CREAM method (Hollnagel 1998).

An antecedent is a factor that influences another factor, while a consequent is the factor being influenced. The relationships between these factors were established so that each consequent corresponds to one or more antecedents from a different set of factors. A scheme of these relationships and the start of a table summarizing these relationships was proposed (Hollnagel 1998). Entries within the table show (forward) links between the factors, with the columns representing the antecedents, listed in the top row, and the rows representing consequents, listed in the left column. This table was then completed by Morais et al. following the provided scheme

(Morais et al. 2022b).

This table contains links in both directions and cycles between some factors. Such relationships would cause issue in the generation and use of a BN, therefore based on additional expert opinion these instances were resolved.

6.4.3.2 Grouped Performance Shaping Factors

The MATA-D provides the dataset for training the NPC and K2 algorithms. However, due to the potential challenges posed by the sparseness of the data, an average of 46 negatives out of the 53 factors were identified per incident, an option to use a grouped version of the dataset and PSFs has also been developed(Morais et al. 2022b).

The data is originally classified into 53 factors, which are breakdowns of 15 categories; “Error Action”, “Observation”, “Interpretation”, “Planning”, “Temporary Person Related Functions”, “Permanent Person Related Functions”, “Equipment”, “Procedures”, “Temporary Interface”, “Permanent Interface”, “Communication”, “Organization”, “Training”, “Ambient Conditions”, “Working Conditions” (Hollnagel 1998, Moura et al. 2016). In this grouped version of the dataset the average incident has 5 out of the 15 factors attributed to its cause. The K2 and NPC algorithms are likely to exhibit improved performance with the grouped dataset due to the reduction in data sparsity and a more balanced representation of the factors. Nevertheless, it is important to acknowledge that valuable information is lost in the process of grouping factors. Consequently, the tool has been designed to offer both the original and grouped dataset options to ensure comprehensive analysis capabilities.

The grouped option also required the modification of the expert opinion table, as grouping the factors lead to more two-way links and cycles, these were again addressed by further expert evaluation.

6.4.4 Implementation

The *HF Relationships tool* was developed and implemented in MATLAB (Inc. 2023). First, the user selects the desired method (K2, NPC, Expert Knowledge, or Aggregated) through a dialog box created using. The "Aggregated" method combines results from the K2 and NPC algorithms along with expert knowledge. The aggregated DAG is determined by simply only including links that are presence at least two of the three methods DAGs. A simplified workflow of this process is outlined in the following Figure 6.2.

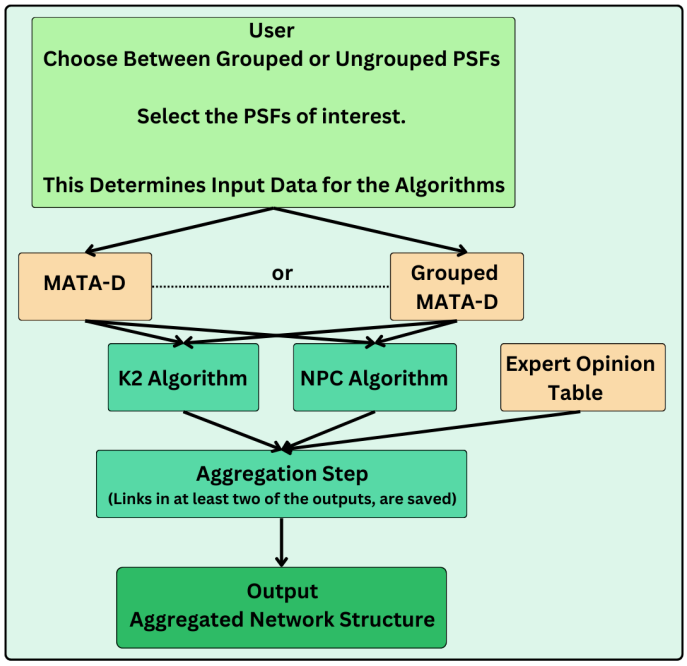

Figure 6.2: Simplified workflow of the Human Factors Causal Relationship tool (Aggregated Option)

The user then can choose whether they want to proceed with either the original PSFs or the grouped version. This allows the user to select features of interest, perhaps the PSFs identified by *HF Classifier tools* if analyzing and reviewing an accident.

Depending on the method chosen, additional parameters such as the significance level for the NPC algorithm and the maximum number of parents per node for the K2 algorithm are requested.

The K2 algorithm, NPC algorithm, node ordering algorithm and other supporting functions are custom implemented within the MATLAB code. These algorithms and functions rely on

MATLAB's built-in functions for mathematical calculations, "`sum`", "`prod`", "`randperm`", and "`gammaln`", statistical tests, "`chi2inv`" and array manipulation, indexing, "`find`", and, "`unique`". These built-in functions facilitate efficient computation and enable the custom algorithms to operate effectively within MATLAB's environment, leveraging MATLAB's robust mathematical and statistical capabilities. The NPC algorithm script used in the tool is based upon the work developed by Guangdi Li (Li 2009).

The *HF Relationships tool* outputs an adjacency matrix, A,for the chosen factors. If n factors were chosen, the adjacency matrix is an $n \times n$ matrix, where the entries determine where there is a directed edge (link) between the nodes. For example, the entry at row i and column j indicates whether there is a directed edge from i to j, 1 if there is and 0 if not. The tool then creates a visualization of the DAG based off this adjacency matrix using the "`digraph`" function.

6.4.5 Performance

The primary objective of developing this tool was to reduce reliance on expert opinion and challenge past biases. Therefore, instead of comparing the algorithm outputs to an expert-derived table, the efficacy of the algorithms were assessed based on their ability to reconstruct a known DAG from simulated data. A test network with 15 nodes was constructed to serve as the goal DAG. Following the method described by Oehm, synthetic datasets were generated by sampling from the conditional probability distributions defined by this DAG (Oehm 2019). This process produced three datasets with 50, 200, and 2000 entries, respectively. These varying sample sizes enabled assessment of the algorithms' performance under different data availability conditions. Both the NPC and K2 algorithms were tested on each of these three datasets. The resulting adjacency matrices were then compared with that of the known DAG to evaluate their performance. To assess the performance of the algorithms, the same logic and metrics, precision, recall and F1-score, as defined in Section 3.3 were used. The performance metrics achieved by both algorithms on each dataset is given in Table 6.2.

Metric	NPC Algorithm			K2 Algorithm		
	50 Samples	200 Samples	2000 Samples	50 Samples	200 Samples	2000 Samples
Accuracy	73%	86%	91%	77%	83%	87%
Precision	7%	52%	55%	14%	52%	55%
Recall	14%	72%	77%	15%	72%	74%
F1-Score	9%	60%	64%	15%	60%	63%

Table 6.2: Performance of NPC and K2 Algorithms on Different Sample Sizes

For this test network, the algorithms performed well for all metrics with the 2000 samples. However, the algorithms also performed well with 200 samples, which is closer to how many samples are in the MATA-D. The algorithms performed better in the recall metric than precision, this means that the algorithm is more likely to produce a false positive than a false negative.
In the implemented tool the aggregated network in part addresses these limitations by incorporating the expert opinion table, and in practice additional expert judgement could be used to prune the network, checking and removing arcs that are considered misleading or incorrect.

6.4.6 Example

An example network has been generated using all grouped features, employing the aggregated approach, as shown in Figure 6.3. This visual representation illustrates the conditional dependencies inferred between various human and system-related factors. Some of the links within the network align well with expected relationships. For example, the progression from Organisation to Planning and Action, as well as the influence of an Observation error on an Action, reflect well-established causal pathways observed in system design and human reliability literature (Hollnagel 1998). These links suggest that higher-level organizational factors influence how operators plan and act, which in turn impacts task performance.
However, several unexpected or questionable links also appear, notably Action to Working Conditions and Action to Interpretation. Intuitively, one would expect Working Conditions to affect Action, not the reverse. Similarly, Interpretation, often a cognitive precursor to action, would more logically influence Action than be affected by it. These counterintuitive dependencies likely

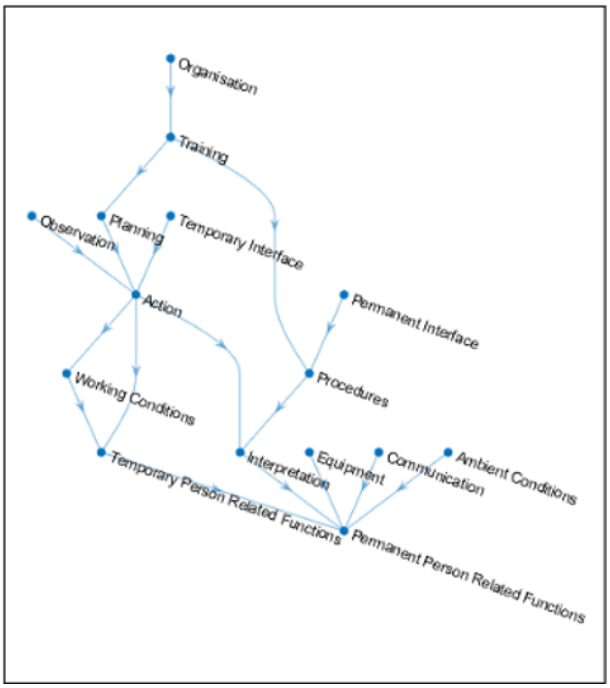

Figure 6.3: Example Output of Human Factors Causal Relationships Tool (Johnson et al. 2022)

result from limitations in the data, such as small sample sizes, unobserved confounding variables, or insufficient granularity in the sequencing of events. In this framework, where associations are inferred from co-occurrence patterns, such limitations can lead to erroneous directional assumptions.

It is also possible that some of these relationships reflect indirect effects that are not well captured in a simplified network. For example, repeated actions may indeed feed back into interpretations over time through learning or bias development, suggesting a bidirectional relationship that is not well represented in a static model.

These issues underscore a key limitation of the current approach. While the aggregated data-driven method allows for rapid structure generation, it does not always respect known causal constraints or expert domain logic. This highlights the need for integrating real-time or dynamic expert judgement into the model-building process.

6.4.7 Conclusion

While the current model reveals promising insights and some expected pathways, it also highlights the risks of drawing strong inferences from limited data.

Nevertheless, the core concept of the *HF Relationships tool*, integrating data-driven structure learning with expert oversight, remains a compelling approach for developing robust, interpretable models of human-system interaction.

The tool enables identification of causal links between PSFs, construction of BNs, and modeling of HEP through real-world data. This has potential to reduce complete reliance on expert judgment while still leveraging their domain expertise.

By aligning structure learning with established methods for determining CPTs, it is possible to develop a human reliability model grounded entirely in empirical data. This approach helps mitigate confirmation and financial biases, leading to more objective HEP estimates and clearer identification of key error-reduction targets.

To improve model confidence and broaden insights, additional data is needed—highlighting the importance of expanding the MATA-D dataset.

Additional methodology details and code discussion for the *HF Relationships tool* can be found in the Appendix, Section A.6.

6.5 Tools for Procedure Guides

Chapter 5 (Enhancing Procedure Quality: Advanced Language Tools for Identifying Ambiguity and High-Potential Violation Triggers) introduces the development, application and testing of tools developed to improve procedural clarity and quality. These tools aim to minimize the chances of mistakes and associated risks. The two innovative tools developed are *Violation Trigger tool* and the *Ambiguity Identifier*. These tools leverage a mix of advanced ML and traditional rule-based NLP techniques to enhance the quality and effectiveness of SOPs. This following section will focus on the technical implementation of the *Ambiguity Identifier*, as

the *Violation Trigger tool* employs the same methodology (Section 6.1.1 (Second Generation – BERT)) to train the background model as the *HF Classifier 2.0*, utilizing the data detailed in Section 5.3.1 (Dataset). Further details, including the specific differences, can be found in the Appendix, Section A.5.

6.5.1 Ambiguity Identifier for Procedure Guides

The *Ambiguity Identifier* was developed to detect various types of ambiguities and misleading steps within procedure guides. The target types of ambiguity, the logic and algorithms applied to identify these and some example applications of this tool are detailed in Chapter 5. The tool employs a range of techniques and methodologies to support the algorithms developed to identify these ambiguities.

Corpus-based approaches are applied for detecting temporal and quantity ambiguities by leveraging predefined collections (corpora) of vague terms and quantifiers. These corpora serve as reference lists containing words and phrases commonly associated with ambiguity. When such terms are detected in procedural instructions, they are flagged as potentially unclear, prompting further review or revision to improve precision and consistency.

PoS tagging is used in the identification of syntactic, conditional, and scope ambiguities by tagging words to determine their grammatical roles and relationships (Sharma 2024). Dependency parsing analyzes the grammatical structure of sentences to identify complex and potentially confusing constructions that lead to syntactic ambiguities (Yamamoto et al. 2022).

Regular expressions (regex) are employed to detect instances where abbreviations and acronyms are used without proper definition, using specific patterns to capture and verify these terms (Friedl 2006).

To handle lexical ambiguity, terms identified as homographs by the PoS, are checked against a dictionary, specifically "WordNet", to identify the multiple definitions (Miller 1995, Oram 2001). This process is enhanced with contextual word embeddings from the BERT model, which helps determine the meaning of each term in the given context (Devlin et al. 2018). Potential ambiguity is identified by comparing the contextual definition to the dictionary definitions.

To implement the rules and techniques, the text is first tokenized using the "`word\_tokenize`" function from the "NLTK" library (Bird et al. 2009). This tokenization process is essential for

subsequent analysis steps, as it allows the text to be processed at the level of individual tokens. By incorporating these techniques, the tool ensures a comprehensive and systematic approach to identifying and resolving ambiguities within procedure guides, ultimately enhancing the clarity, consistency, and safety of operational protocols.

6.5.1.1 Corpus-Based Approach

The corpus-based approach began with compiling a dictionary of terms derived from domain knowledge and an analysis of typical SOP language. This dictionary includes terms known to cause ambiguity, such as imprecise temporal conjunctions and quantifiers. Examples include terms like "*frequently*", "*periodically*", "*some*", and "*enough*", which can lead to varied interpretations.

The algorithm iterates through the tokenized text and matches the tokens against the corpus of ambiguous terms. Each token is checked to determine if it corresponds to any term in the dictionary. When a match is found, the algorithm flags the entire sentence containing the ambiguous term for further review. This step is essential as it helps in pinpointing specific parts of the SOP that may need clarification due to potential ambiguity. The flagged sentences are stored and output as a list for further analysis, reviewers can then assess and address these potential ambiguities.

This approach effectively combines domain-specific knowledge with NLP techniques to identify and flag potentially ambiguous instructions within SOPs.

6.5.1.2 Part-of-Speech Tagging

PoS tagging is a fundamental NLP technique where words in the text are tagged with their corresponding parts of speech (nouns, verbs, adjectives), helping to analyzing their grammatical roles and relationships (Sharma 2024). For instance, the word "*lead*" can be tagged as a noun (`NN`) or a verb (`VB`) depending on its context within the sentence.

The PoS tagger used in this tool is development is supplied by the "`spaCy`" library (Honnibal

and Montani 2017). This tagger has been trained to handle complex sentences and accurately evaluate grammatical roles. The tagged text is further evaluated using a dependency parser to identify different types of ambiguities.

To detect syntactic ambiguity arising from prepositional phrases, the tokenized text is first tagged. For each preposition tagged, "`(token.dep_=='prep')`", the method examines the two preceding tokens to determine if both can be logically modified by the prepositional phrase. The suitability of these tokens is evaluated based on their PoS tags, specifically considering nouns, proper nouns, and verbs as potential candidates for modification. These relationships are then evaluated using the dependency parser.

For example, in the sentence "*Secure the panel with the clamps near the turbine*", the phrase "*near the turbine*" could attach either to "*clamps*" or "*panel*". Both would be tagged as nouns (`NNS` and `NN` respectively), making both grammatically viable targets of the prepositional phrase. The parser assesses the dependency tree to determine which noun the prepositional phrase most likely modifies. If both options are structurally valid, the sentence is flagged for prepositional phrase ambiguity.

Conditional ambiguity arises when the conditions in a sentence, often introduced by conjunctions such as "*if*", "*when*", and "*unless*", are not clearly defined. The PoS tagger identifies the conditional conjunctions, before dependency parsing is utilized to examine the dependency structure of each sentence to check if these conjunctions are followed by well-defined condition clauses. For instance, in the instruction "*Activate the cooling system if pressure rises and temperature falls below threshold*", the use of the conjunction "*and*" creates uncertainty, is the condition triggered when both sub-conditions are met, or is either sufficient.

Scope ambiguity arises when the extent or boundaries of actions within a sentence are unclear, often due to poorly defined associations between verbs, subjects, and direct objects. By iterating over tokens and analyzing PoS tags the presence of a nominal subject and a main verb for each direct object is verified. Dependency parsing is then applied to check the relationship between these objects, by checking the token's dependencies and their hierarchical structure to confirm clear associations. If any of these components are absent or incorrectly associated, the sentence is flagged for scope ambiguity. For example, consider the instruction, "*The supervisor will approve the inspection and the checklist for the maintenance task.*" PoS tagging identifies "*approved*" as the main verb and "*inspection*", "*checklist*", and "*task*" as noun phrases. However, the dependency structure reveals that "*inspection*" is a clear object of "*approved*", while it is not clear whether "*the checklist*" is also subject to "*approves*", or if it is associated with "*the maintenance*

task" as part of a different clause. Furthermore, "*for the maintenance task*" could modify either "*checklist*" or the entire phrase. As not all objects are unambiguously tied to the main verb, the parser is unable to resolve the verb direct object mapping. As a result, the sentence would be flagged for scope ambiguity.

6.5.1.3 Dependency Parsing

Dependency parsing analyzes the grammatical structure of sentences to identify complex constructions that might lead to ambiguities in SOPs, using the "`spaCy`" library again (Honnibal and Montani 2017). Dependency parsing represents each sentence as a tree where words are nodes, and dependencies between words are edges (Yamamoto et al. 2022). This tree structure helps in identifying how words in a sentence relate to each other, making it easier to identify complex syntactic constructions that could be ambiguous. The dependency parser in "`spaCy`" assigns a head (a root word) and dependencies to each token in a sentence. By examining these dependencies, the tool can identify various relationships and structures within sentences.

This capability, combined with the PoS, has been utilized in several ways, including identifying the presence or absence of required clauses, detecting sentences with multiple subordinate clauses, analyzing nested structures, and pinpointing ambiguous prepositional phrase attachments.

Prepositional phrase attachment ambiguity is detected by examining the dependencies of the identified prepositional phrases. A prepositional phrase consists of a preposition and its object ("*with a wrench*", "*in the chamber*") and typically functions to modify a noun or a verb. Attachment refers to the grammatical process of linking a prepositional phrase to another element in the sentence to indicate which word or phrase it modifies. These attachments are crucial for interpreting sentence meaning. However, ambiguity arises when a prepositional phrase could logically modify more than one part of the sentence, leading to multiple plausible interpretations. The parser evaluates the potential attachment points for each prepositional phrase based on syntactic structure and dependency relations. If a prepositional phrase can attach to more than one constituent without a clear disambiguating cue, the sentence is flagged for ambiguity.

For conditional ambiguity, after the conditional conjunctions are tagged and identified, the algorithm then examines the dependency structure of each sentence to check if these conjunctions

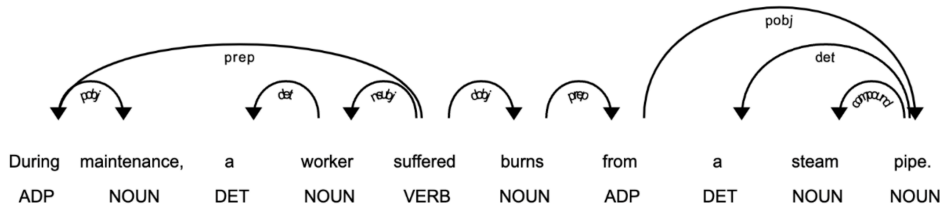

Figure 6.4: Dependency Parsing and PoS Tagging visualization

are followed by well-defined condition clauses. Specifically, the function looks for adverbial clause (a group of words that functions like an adverb) dependencies associated with the conjunctions, which indicate a clear conditional relationship. If such dependencies are absent, the condition is marked as ambiguous. Scope ambiguity also relied upon dependency parsing to check the tokens, of the nominal subject and main verbs, dependencies and their hierarchical structure to confirm clear associations. An example of dependency parsing and PoS tagging is visualized in Figure 6.4, generated using "*SkillNER*" (Fareri et al. 2021). Sentences containing multiple subordinate clauses can be syntactically ambiguous as the relationships between the clauses may become unclear, leading to multiple potential interpretations of the sentence's meaning. To address this, the tool iterates through each sentence and identifies subordinate clauses by looking for tokens with the dependency label, "`mark`". If a sentence contains more than two subordinate clauses, it is flagged as potentially ambiguous. As an example, the sentence "*The system will reboot when the voltage drops, if the coolant fails, and unless the override is engaged*", the presence of three conditional subordinate clauses ("*when*", "*if*", and "*unless*") complicates the logical flow and makes the condition for reboot ambiguous.

Nested structures complicate sentence interpretation when a subordinate clause contains another subordinate clause, creating layers of dependencies that can obscure the sentence's main point (Lakretz et al. 2020). The dependency parser detects these nested structures by analyzing the hierarchical relationships between clauses, and checking if any subordinate clause has children with the "`mark`" dependency label. If subordinate clauses are nested within each other, the sentence is flagged due to the increased complexity and potential for misunderstanding. For example, in the sentence "*If the operator acknowledges that the pressure drops when the pump is activated, then the process may continue*", the clause "*that the pressure drops when the pump is activated*" contains a nested subordinate clause inside a conditional statement. This layering increases processing complexity and potential misinterpretation, prompting the system to flag it as a potentially ambiguous.

6.5.1.4 Regular Expression (Regex)

Regular expressions are a powerful tool for detecting patterns in text and have been applied to identify acronyms and abbreviations that are undefined (Friedl 2006). The implementation of regex in Python is facilitated by the "re" module, which provides a set of functions for working with regular expressions (Foundation 2023).

Regex patterns are defined to match specific text sequences. Acronyms and abbreviations usually consist of uppercase sequences of 2 to 5 characters. The pattern used to identify the acronyms and abbreviations is "`r'\b[A-Z]{2,5}\b'`". In this pattern, "`\b`" denotes a word boundary, "`[A-Z]`" specifies uppercase letters, and "`{2,5}`" indicates the sequence length must be between 2 and 5 characters. This pattern effectively matches any word boundary followed by 2 to 5 uppercase letters and ending with another word boundary. For example, it would match terms such as `HSE`, `PPE`, and `SOP`, but would not match lowercase words or mixed-case terms like `Co2` or `risk`.

To extract acronyms and abbreviations a text, the "`re.findall`" function is used. This function returns all non-overlapping matches of the pattern in the string as a list. This approach provides a straightforward way to collect these patterns for further analysis. These are stored as potentially ambiguous acronyms/abbreviations.

After extraction, it now needs to be verified whether the acronyms or abbreviations are defined within the text or not. This verification can be done using additional regex patterns to search for common definition formats. The patterns used are given in Section 5.2.6 (Abbreviation and Acronym Ambiguity Rules). These are patterns that state whether the acronym/abbreviation is followed or proceeded by its full form/definition in parentheses or with a hyphen.

By defining these patterns and the identified acronyms/abbreviations, the text can be searched, using the "`re.search`" function, to see if corresponding definitions are provided. If a definition is identified the term is removed from the list of potentially ambiguous acronyms/abbreviations list. For example, if zz*Application Programming Interface (API)*" is found in the document, then "*API*" would be excluded from the final ambiguity list.

Acronyms and abbreviations are also often defined in tables within SOPs. To ensure these are not overlooked, it is important to extract and inspect the table structures for definitions.

Using the "`Camelot`" library, tables are extracted from SOP PDF documents and then converted into a structured tabular format using a `DataFrame` object from the "`pandas`" library

(Developers 2021, Team 2024). "`Camelot`" is specifically designed for parsing tables from PDFs, while "`pandas`" provides powerful data structures and operations for data manipulation and analysis, making it well-suited for structuring and processing the extracted table data. After extracting the tables, iterate through each table to search for previously identified acronyms or abbreviations. For each identified acronym or abbreviation in a table, the adjacent cells are checked (to the left/right in the same row) for potential definitions or explanations. If an acronym or abbreviation is found in a table and a definition is present in an adjacent cell, remove this term from the list of potentially ambiguous acronyms/abbreviations.

The final list of potentially ambiguous acronyms and abbreviations is then generated for review. It is best practice to define each term at least once in an SOP. This ensures that all terms are clearly explained, thereby reducing ambiguity. By doing so, the clarity and precision of technical documents are significantly enhanced, making them more user-friendly and easier to understand. This approach not only aids in compliance and accuracy but also improves overall communication and operational efficiency.

6.5.1.5 Lexical Ambiguity Methodology

To manage lexical ambiguity in procedural documents, lexical databases are integrated with advanced NLP techniques. First, potential homographs are identified in the text using a PoS tagger, which labels each word based on its grammatical role. Homographs are words that are spelled the same but have different meanings and sometimes different pronunciations (Grammarly 2023). The PoS tagger processes the text and assigns a PoS tag to each word, indicating its grammatical role (e.g., noun, verb, adjective). By examining the PoS tags, words that are used in different grammatical contexts are identified. If a word appears with multiple PoS tags (e.g., both as a noun and a verb), it is flagged as a potential homograph.

For example, consider the sentence, "*The workers will lead the briefing before inspecting the lead shielding.*" Here, the word "*lead*" appears twice, first as a verb (to guide) and then as a noun (a heavy metal). The PoS tagger identifies the first instance as a verb (`VB`) and the second as a noun (`NN`), flagging it as a homograph due to its different grammatical roles and potential meanings.

After identifying these homographs, "WordNet" is used to retrieve all possible meanings of the

terms. “WordNet” organizes English words into sets of synonyms (synsets), providing definitions and example sentences for each synset (Oram 2001, Miller 1995). For each identified homograph, the associated synsets are retrieved, offering a comprehensive list of potential meanings.
For the word “lead”, WordNet may return multiple synsets,

- lead.n.01: a soft heavy toxic metallic element
- lead.v.01: to take charge or guide
- lead.v.02: to cause someone to go with one

To disambiguate the meanings of homographs within specific contexts, BERT is employed. This leverages BERT’s pre-training on a large corpus of text and its ability to understand the context of words bidirectionally (Devlin et al. 2018).
The process starts by encoding the sentences containing the homographs using BERT, generating contextual embeddings for each word. The embeddings encapsulate the meaning of the word within the specific sentence context. For each synset retrieved from “WordNet”, the embeddings are also generated.
The similarity between the contextual embedding of the homograph and each synset embedding is calculated using cosine similarity (Porter 2023. Cosine similarity between two vectors A and B is defined as,

$$\cos(\theta) = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{6.25}$$

where, A and B are the embedding vectors of the homograph in context and the synset definition, respectively. Cosine similarity is a measure used to determine the similarity between two vectors by calculating the cosine of the angle between them, with scores ranging from -1 and 1. A score of 1 indicates that the vectors are identical in orientation, meaning they point in the same direction and are highly similar. A score of 0 signifies orthogonality, where the vectors are at a right angle to each other, indicating no similarity. A score of -1 denotes that the vectors are diametrically opposed, pointing in opposite directions and signifying complete dissimilarity (Porter 2023). In this tool’s context, analyzing homographs using BERT embeddings and “WordNet” synsets, a cosine similarity close to 1 suggests that the synset meaning aligns closely with the homograph’s contextual use, while a score near 0 indicates poor alignment, and -1 would indicate that the meanings are completely opposed.

The similarity scores help identify the synset that best matches the homograph in its given context. If the highest similarity score is below 0.5 or if multiple definitions have similar scores, the term and therefore sentences containing it are flagged as potentially ambiguous.
Continuing the earlier example, the BERT embeddings of the first instance of "*lead*" in "*lead the briefing*" are compared against verb synsets, while the second instance in "*lead shielding*" is compared against noun synsets. The tool computes cosine similarities for each case,

- Contextual embedding (lead as verb) vs. lead.v.01: similarity = 0.81
- Contextual embedding (lead as noun) vs. lead.n.01: similarity = 0.68

These scores confirm the appropriate contextual meanings, and no ambiguity would be flagged. By combining these techniques and approaches, the *Ambiguity Identifier* can significantly enhance the use of established and emerging NLP technology to tackle critical safety and compliance challenges in high-risk industries. By systematically addressing ambiguities, these tools not only enhance the clarity and effectiveness of SOPs but also contribute to overall operational safety and efficiency. Additional methodology and coding choices for the *Ambiguity Identifier* are detailed in the Appendix, Section A.4.
Integrating the *Violation Trigger tool* and the *Ambiguity Identifier* into standard safety practices ensures a proactive approach to managing procedural risks and supports the continuous improvement of safety protocols in complex industrial settings.

6.6 Future Work

There is room for future development across each tool, as discussed in each of the previous chapters. Future enhancements will initially focus on improving the usability, ensuring that the tools are accessible to a broader audience, including those with limited technical expertise.
A significant and valuable piece of future work will be to develop a user-friendly interface that incorporates each tool, integrates their outputs, and supports data collection. This could take the form of either a web-based platform or standalone software, depending on user requirements. A web-based solution would facilitate easier data sharing, centralized access, and collaborative

use across organizations. In contrast, downloadable software may be preferable in settings where data privacy, local control, or offline access is a priority. In either case, the core objective is to provide seamless access to all tools through a unified interface.

Further development will also explore expanding the datasets used for training the models, incorporating more diverse sources of data to improve the robustness and generalizability of the tools. This expansion would help in addressing the epistemic uncertainties and adapting to changes in human behavior, societal norms, and technological advancements.

Continued research and development efforts should aim to integrate the latest technologies to further enhance the capabilities and reach of these tools. One promising avenue is incorporating NLP models for various languages, which would allow for a more comprehensive and accurate analysis of linguistic and grammatical structures, better than the pre-translation of documents into English (Intrator et al. 2024). Additionally, there are advanced NLP techniques that can handle multilingual data, which would support cross-cultural and international studies and learning (Li et al. 2024b). This multilingual support would significantly broaden the impact of these efforts, enabling more inclusive and globally relevant research.

Accident reports and safety documents often contain critical information embedded in images, diagrams, and charts. Integrating NLP models with image and multimedia analysis capabilities would significantly increase the available information and enhance the functionality of these tools (Jiao et al. 2024). By developing the ability to analyze and interpret visual data alongside textual data, these tools can provide a more holistic understanding of incidents.

By fostering collaboration between academia, industry, and regulatory bodies, future work can drive the continuous improvement and adoption of these advanced tools, ultimately contributing to safer and more reliable safety-critical systems.

6.7 Conclusion

This work has successfully demonstrated the innovative application of ML, data-driven techniques, and NLP in HRA and safety assessment.

By leveraging cutting-edge techniques and developing novel adaptations and methodologies, these tools provide valuable insights into the complexities of human factors in safety-critical

systems.

The integration of ML algorithms, alongside innovative NLP approaches, has resulted in tools that not only automate but also enhance the analysis and understanding of human factors in high-risk industries. This underscores the importance of interdisciplinary approaches in addressing complex safety challenges, combining expertise from ML, NLP, and HRA domains to develop practical and impactful solutions.

Chapter 7

General Conclusions and Future Work

The developments presented in this thesis represent a significant advancement for the field of HRA, offering a comprehensive suite of tools that enhance data gathering, explainability, learning from past accidents, constructing data-driven models, and evaluating and improving procedural safety. By leveraging cutting-edge techniques and developing novel adaptations and methodologies, these tools provide valuable insights into the complexities of human factors in safety-critical systems. The first-generation *HF Classifier* (Chapter 3) automates the identification of human errors and influencing factors in accident reports, enhancing the efficiency of HRA data collection. Building on this, the *HF Classifier 2.0* (Chapter 4, Section 4.2), leveraging BERT, improves classification accuracy and contextual understanding. Additionally, the *Human-Centric Summarizer* (Chapter 4, Section 4.3), utilizing BART, provides concise and informative summaries of accident reports, supporting informed and deliberate safety decision-making by reducing cognitive load and information review time. The *High-Potential Violation Trigger Identification tool* (Chapter 5, Section 5.3) and the *Ambiguity Identification tool* (Chapter 5, Section 5.2) for improving SOPs are a move towards a more proactive and informed approach to safety management. Their ability to identify high-potential violations and ambiguities ensures that all personnel are well-equipped to perform their duties safely and effectively, ultimately safeguarding human lives and environmental health. Furthermore, the HF*HF Relationships tool* (Chapter 6, Section 6.4) offers a data-driven approach to identifying causal links between PSFs, supporting the development of data-driven human error models.

This work underscores the importance of interdisciplinary approaches in addressing complex safety challenges, combining expertise from the ML, NLP, and HRA domains to develop practical and impactful solutions. The practical applications of these tools are vast, extending across various high-risk industries such as aviation, nuclear power, and healthcare, where understanding human factors and clear and accurate procedures are crucial for preventing accidents and maintaining operational safety.

7.1 Implications to HRA and Safety Practices

The usefulness of the tools is further demonstrated by their ability to integrate seamlessly into existing safety practices, providing stakeholders with advanced analytical capabilities without requiring extensive technical expertise. This accessibility ensures that the benefits of these tools can be widely realized, leading to safer and more reliable operations across diverse sectors.

HRA plays a crucial role in understanding, assessing, and mitigating human errors within safety-critical systems. Presently, HRA employs various methods to evaluate human performance and identify potential sources of error and increased risk. These methods typically encompass qualitative assessments, such as expert judgments, checklists, and cognitive task analyses, alongside quantitative approaches, including probabilistic risk assessments and statistical analyses of incident data.

However, traditional HRA methods encounter several significant challenges. One major issue is the subjectivity and bias inherent in expert judgment. While invaluable, expert input can introduce these elements, potentially compromising the consistency and accuracy of HRA outcomes. Additionally, the processes involved in comprehensive HRA are often time-consuming and resource-intensive, posing difficulties for frequent and thorough analyzes.

The tools developed in this thesis address these challenges by offering automated, data-driven solutions that support and enhance the efficiency of HRA. These tools contribute to enhanced objectivity and consistency by automating the analysis of accident reports and other safety-related documents. This automation reduces reliance on subjective judgment, providing more consistent and objective insights into human errors and contributing factors.

The tools significantly improve the efficiency of the HRA process by automatically processing

textual data. By automating labor-intensive tasks to support more efficient yet thorough analysis, the tools optimize resources, freeing up valuable assets that can be redirected to other critical activities. For example, allowing organizations to dedicate more time to refining their safety protocols, ensuring that these remain up-to-date and responsive to new information and emerging risks.

The tools are able to identify latent conditions and systemic issues that may not be immediately apparent through traditional methods. By analyzing and learning the patterns and correlations in the data, the tools uncover underlying factors contributing to human errors, allowing organizations to address root causes rather than merely treating symptoms.

Beyond specific HRA applications, these tools contribute to general safety improvements in several ways. The development of the SOP focused tools facilitates proactive risk management. By detecting potentially high-risk violation triggers and identifying ambiguities, the tools enable the early identification of risks so that they can be addressed through thorough evaluation and informed intervention before incidents occur. Additionally, the developed tools support training improvements by providing key information of recent incidents and identified key procedure directives, ensuring that personnel receive relevant and up-to-date training.

The integrated use of the presented tools provides a holistic approach to gathering human factors data, enhancing the explainability of human reliability, and improving safety protocols. Validation through case studies showcases their potential to transform safety practices by providing insights and supporting informed decision-making.

Importantly, the tools are not intended to expedite changes without appropriate review, but to assist safety professionals in identifying where detailed scrutiny is warranted, ensuring any updates are carried out with the necessary diligence and validation.

7.2 Recommendations for Future Work and Improvements

Throughout the chapters that introduce, detail the development, implement, and apply the tools, various opportunities for future work and improvements have been identified. These opportunities mainly relate to a few areas, the development of a user-friendly interface, the continued integration and leveraging of advancing technologies, further efforts to collect and analyze data, along with some more specific suggestions.

Developing a user-friendly interface is considered essential for enhancing the accessibility and usability of the tools developed in this project. While a web-based platform offers significant advantages, such as centralized access, streamlined document uploads, and integrated outputs, it also supports data sharing and collaboration across organizations. Alternatively, a standalone application may be more appropriate in contexts where data privacy, offline access, or strict security protocols are required. Regardless of the delivery mode, the interface should enable seamless interaction with the tools, allowing users to upload documents, view and interpret results, and work across modules with minimal friction. To support ongoing improvement, a feedback mechanism will be integrated, enabling users to report issues, suggest enhancements, and share real-world insights. This feedback loop will drive a continuous development cycle, ensuring that the tools evolve to meet user needs and maintain their relevance in practice.

To improve the tools' performance and generalizability continuous efforts to collect and analyze diverse data sources will be required. This involves collecting documents from a wide range of industries and sources to address epistemic uncertainties and adapt to changes in human behavior, societal norms, and technological advancements overtime. The website will facilitate this data collection, which will not only enable users to submit documents for analysis, but also submit manually reviewed texts whether this be procedural steps with ambiguities or classified accident reports. This user-contributed data will be invaluable for training and refining ML models, enhancing the overall effectiveness of the tools.

Staying at the forefront of technological advancements in NLP and ML is crucial for the continued development of these tools. Future iterations should focus on leveraging the latest advancements in NLP and LLMs, which are rapidly evolving and offer new capabilities that can significantly enhance the functionality of the tools. One key area of development is the integ-

ration of NLP models with image and multimedia analysis capabilities. Safety documents and incident reports often contain critical information embedded in images, diagrams, and charts. Developing the ability to analyze and interpret visual data alongside textual data will provide a more comprehensive understanding of incidents.

Additionally, to support the automated analysis of documents, several suggestions are made to the industries. The most significant recommendations include ensuring consistency in document format, such as using similar chapter titles and sequences throughout. Furthermore, if possible, accident reports should be made publicly available on the internet to facilitate easier access and analysis. Additionally, from the review and evaluation of many SOPs, the inclusion of a table of definitions for all acronyms, abbreviations, and units at the beginning of the documents is also recommended.

Less significant recommendations also include ensuring that sections dedicated to normal characteristics are clearly separated and preferably placed at the beginning or end of the file. As well as critical information regarding accident causes not being mixed within such sections of the document.

Reducing the amount of unimportant text, such as company names in headers and footers, will also aid in improving the clarity of the extracted text. Some of these suggestions are considered unlikely to be fully implemented. Therefore, the development of improved pre-processing and targeted text extraction algorithms will be necessary to effectively address these challenges.

This work has shown how NLP and ML can be effectively applied to a range of HRA-related tasks, offering scalable and consistent support to traditionally qualitative processes. Beyond these there are numerous related challenges, areas and fields that could benefit from the integration of such technologies. Some suggested areas of future work include,

- Automatic generation of tailored safety training materials based on recent incidents, identified risks, or specific job roles.
- Analysis of internal communications, surveys, and reports to gauge the overall safety culture within an organization and identify areas for improvement.
- Development of a tool trained on safety regulations and internal policies, which can then be used to scan and review documentation and communications to ensure adherence.

In conclusion, this work represents a significant step forward in the field of HRA and safety assessment. The developed tools contribute to the development of safer operational practices, ultimately safeguarding human lives and environmental health. Their outputs are intended to inform structured reviews, ensuring that procedural and safety-related decisions maintain the depth of understanding required in complex systems. They exemplify how innovative applications of ML and NLP can lead to substantial improvements in understanding and mitigating human factors in safety-critical environments.

Bibliography

Ahmed, L. et al. (2020). 'Development of a procedure writers' guide framework: Integrating the procedure life cycle and reflecting on current industry practices'. In: *International Journal of Industrial Ergonomics* 76. DOI: 10.1016/j.ergon.2020.102930.

Alper, S. J. and B. Karsh (2009). 'A systematic review of safety violations in industry'. In: *Accident Analysis & Prevention* 41.4, pp. 739–754. DOI: 10.1016/j.aap.2009.03.013.

Amare, G. (2012). 'Reviewing the Values of a Standard Operating Procedure'. In: *Ethiopian Journal of Health Science* 22.3, pp. 205–208. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3511899/.

ANP (2020). *Gas Natural E Biocombustiveis*. URL: https://braziliannr.com/category/anp/.

Arrieta, A. B. et al. (2020). 'Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI'. In: *Information Fusion* 58, pp. 82–115. DOI: 10.48550/arXiv.1910.10045.

Baybutt, P. (2015). 'Designing risk matrices to avoid risk ranking reversal errors'. In: *Process Safety Progress* 35.1. DOI: 10.1002/prs.11768.

BBC (2019). *Boeing: Which airlines use the 737 Max 8?* URL: https://www.bbc.co.uk/news/business-47523468.

Bell, J. and J. Holroyd (2009). *Review of human reliability assessment methods*. URL: https://www.hse.gov.uk/research/rrhtm/rr679.htm.

Belval, E. (2023). *PDF2image*. URL: https://pypi.org/project/pdf2image/.

Benmohamed, E., H. Ltifi and M. Ayed (2022). 'ITNO-K2PC: An improved K2 algorithm with information-theory-centered node ordering for structure learning'. In: *Journal of King Saud University - Computer and Information Sciences* 34.4, pp. 1410–1422. DOI: 10.1016/j.jksuci.2020.06.004.

Berry, D. M., E. Kamsties and M. M. Krieger (2003). *From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity*. URL: https://cs.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf.

Bhargav, S. et al. (2022). 'A comparison study of abstractive and extractive methods for text summarization'. In: *Proceedings of the International Conference on Paradigms of Communication, Computing and Data Sciences*, pp. 601–610. DOI: 10.1007/978-981-16-5747-4_51.

Bird, S., E. Klein and E. Loper (2009). *Natural Language Processing with Python*. O'Reilly Media Inc. URL: https://www.nltk.org/book/.

Boskeljon-Horst, L., R. de Boer and S. Dekker (2024). 'Reducing gaps between paper and practice requires more than a technical alignment'. In: *Safety Science* 170. DOI: 10.1016/j.ssci.2023.106291.

Bouchaala, L. et al. (2010). 'Improving algorithms for structure learning in Bayesian Networks using a new implicit score'. In: *Expert Systems with Applications* 37.7, pp. 5470–5475. DOI: 10.1016/j.eswa.2010.02.065.

Braverman, M. (2011). *Information Theory in Computer Science - Lecture 3*. URL: https://www.cs.princeton.edu/courses/archive/fall11/cos597D/L03.pdf.

Brownlee, J. (2020). *A Gentle Introduction to Information Entropy*. URL: https://machinelearningmastery.com/what-is-information-entropy/.

Campbell, M. (2000). 'Technology and Temporal Ambiguity'. In: *Technology and the Good Life?* DOI: 10.7208/chicago/9780226333885.003.0015.

CCPS (2010). *Guidelines for Risk Based Process Safety*. John Wiley & Sons. URL: https://www.aiche.org/resources/publications/books/guidelines-risk-based-process-safety.

CCPS (2020). *Guidelines for Process Safety During the Transient Operating Mode: Managing Risks during Process Start-ups and Shut-downs*. American Institute of Chemical Engineers, Center for Chemical Process Safety: Wiley. URL: https://www.aiche.org/ccps/resources/publications/books/guidelines-process-safety-during-transient-operating-mode-managing-risks-during-process-start-ups#:~:text=This%20essential%20guide%20for%20plants,safely%20after%20an%20unexpected%20shutdown..

Ceccata, M. et al. (2005). *Ambiguity Identification and Measurement in Natural Language Texts*. URL: https://www.researchgate.net/publication/30530745_Ambiguity_Identification_and_Measurement_in_Natural_Language_Texts.

Cheng, Qinyuan et al. (2023). *Improving Contrastive Learning of Sentence Embeddings from AI Feedback*. arXiv: 2305.01918 [cs.CL]. URL: https://arxiv.org/abs/2305.01918.

Chowdhary, K. (2020). 'Natural Language Processing'. In: *Fundamentals of Artificial Intelligence*. Springer, pp. 603–649. DOI: 10.1007/978-81-322-3972-7_19.

Chronopoulos, C. and N. H. Guzman (2020). 'Is Smartness Risky? A Framework to Evaluate Smartness in Cyber-Physical Systems'. In: *Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference*. DOI: 10.3850/978-981-14-8593-0_3569-cd.

Cooper, G. and E. Herskovits (1992). 'A Bayesian method for the induction of probabilistic networks from data'. In: *Machine Learning* 9, pp. 309–347. DOI: 10.1007/BF00994110.

Cowdrey, D. (2023). *Mitigating the hidden risks of start-up and shutdown operations*. URL: https://www.ishn.com/articles/113870-mitigating-the-hidden-risks-of-start-up-and-shutdown-operations.

Cullen, L. W. (1993). *The public inquiry into the Piper Alpha disaster*. URL: https://www.hse.gov.uk/offshore/piper-alpha-disaster-public-inquiry.htm.

Curley, S. P., J. F. Yates and R. A. Abrams (1986). 'Psychological sources of ambiguity avoidance'. In: *Organizational Behavior and Human Decision Processes* 38.2, pp. 230–256. DOI: 10.1016/0749-5978(86)90018-X.

Developers, Camelot (2021). *Camelot: PDF Table Extraction for Humans*. URL: https://camelot-py.readthedocs.io/en/master/.

Devlin, J. et al. (2018). 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'. In: *arXiv preprint*. DOI: 10.48550/arXiv.1810.04805.

Dougherty, E. (1995). ''Violation'—does HRA need the concept?' In: *Reliability Engineering & System Safety* 47.2, pp. 131–136. DOI: 10.1016/0951-8320(94)00058-V.

Drupsteen, L., J. Groeneweg and G. I. Zwetsloot (2013). 'Critical Steps in Learning From Incidents: Using Learning Potential in the Process From Reporting an Incident to Accident Prevention'. In: *International Journal of Occupational Safety and Ergonomics* 19.1, pp. 63–77. DOI: 10.1080/10803548.2013.11076966.

El-Kassas, W. S. (2011). 'Automatic text summarization: A comprehensive survey'. In: *Expert Systems with Applications* 165. DOI: 10.1016/j.eswa.2020.113679.

Face, Hugging (2023a). *BART*. URL: https://huggingface.co/docs/transformers/en/model_doc/bart.

– (2023b). *BERT*. URL: https://huggingface.co/docs/transformers/en/model_doc/bert.

– (2023c). *Transformers*. URL: https://huggingface.co/docs/transformers/en/index.

Fan, H., H. Enshaei and S. Jayasinghe (2022). 'Human Error Probability Assessment for LNG Bunkering Based on Fuzzy Bayesian Network-CREAM Model'. In: *Journal of Marine Science and Engineering* 10.3. DOI: 10.3390/jmse10030333.

Fantechi, A., S. Gnesi and L. Semini (2023). 'Rule-based NLP vs ChatGPT in ambiguity detection, a preliminary study'. In: *REFSQ Workshops 2023*. URL: https://api.semanticscholar.org/CorpusID:258559142.

Fareri, Silvia et al. (Jan. 2021). *SkillNER: Mining and Mapping Soft Skills from any Text*. DOI: 10.48550/arXiv.2101.11431.

Fenniak, M. et al. (2022). *The PyPDF2 library*. URL: https://pypdf2.readthedocs.io/en/latest/meta/CONTRIBUTORS.html.

Ferson, S. et al. (2015). 'Natural language of uncertainty: numeric hedge words'. In: *International Journal of Approximate Reasoning* 57, pp. 19–39. DOI: 10.1016/j.ijar.2014.11.003.

Foundation, Python Software (2023). *re — Regular expression operations*. URL: https://docs.python.org/3/library/re.html.

French, Simon et al. (2011). 'Human reliability analysis: A critique and review for managers'. In: *Safety Science* 49.6, pp. 753–763. ISSN: 0925-7535. DOI: https://doi.org/10.1016/j.ssci.2011.02.008. URL: https://www.sciencedirect.com/science/article/pii/S0925753511000555.

Friedl, J. (2006). *Mastering Regular Expressions*. O'Reilly Media Inc.

Garrido-Merchan E., Gozali-Brizula R. and S. Gonzalez-Carvajal (2023). 'Comparing BERT against traditional machine learning text classification'. In: *Journal of Computational and Cognitive Engineering* 2.4. DOI: 10.47852/bonviewJCCE3202838.

Geurts, B. (2004). *On an ambiguity in quantified conditionals*. URL: https://api.semanticscholar.org/CorpusID:5864890.

Giarelis N., Mastrokostas C. and N. Karacapilidis (2023). 'Abstractive vs. Extractive Summarization: An Experimental Review'. In: *Applied Science* 13.13. DOI: 10.3390/app13137620.

Gleich B., Creighton O. and L. Kof (2010). 'Ambiguity Detection: Towards a Tool Explaining Ambiguity Sources'. In: *Requirements Engineering: Foundation for Software Quality*, pp. 218–232. DOI: 10.1007/978-3-642-14192-8_20.

Goh, Y. M. and C. U. Ubeynarayana (2017). 'Construction accident narrative classification: An evaluation of text mining techniques'. In: *Accident Analysis & Prevention* 108, pp. 122–130. DOI: 10.1016/j.aap.2017.08.026.

Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing*. Vol. 10. 1. Synthesis Lectures on Human Language Technologies. DOI: 10.2200/S00762ED1V01Y201703HLT

Gonçalves, C. C. and L. G. Trabasso (2018). 'Aircraft Preventive Maintenance Data Evaluation Applied in Integrated Product Development Process'. In: *Journal of Aerospace Technology and Management* 10. DOI: 10.5028/jatm.v10.706.

Goodfellow I., Bengio Y. and A. Courville (2016). *Deep Learning*. MIT Press. URL: http://www.deeplearningbook.org.

Gough, J. and M. Hamrell (2009). 'Standard Operating Procedures (SOPs): Why Companies Must Have Them, and Why They Need Them'. In: *Therapeutic Innovation and Regulatory Science* 43.1, pp. 69–74. DOI: 10.1177/009286150904300112.

Grammarly (2023). *What Is a Homograph? Definition and Examples*. URL: https://www.grammarly.com/blog/homograph/.

Grech, M. R. and A. Smith (2002). 'Human Error in Maritime Operations: Analyses of Accident Reports Using the Leximancer Tool'. In: *Human Factors and Ergonomics Society Annual Meeting*. Vol. 46, pp. 1718–1721. URL: https://api.semanticscholar.org/CorpusID:108577901.

Groth, K. and A. Mosleh (2011). 'Development and Use of a Bayesian Network to Estimate Human Error Probability'. In: *Proceedings of the ANS International Topical Meeting on Probabilistic Safety Assessment and Analysis (PSA 2011)*. Wilmington, N.C., U.S. URL: https://api.semanticscholar.org/CorpusID:16820637.

– (2012). 'A data-informed PIF hierarchy for model-based Human Reliability Analysis'. In: *Reliability Engineering & System Safety* 108, pp. 154–174. DOI: 10.1016/j.ress.2012.08.006.

GUI, HUGIN (2020). *NPC Algorithm*. URL: https://download.hugin.com/webdocs/manuals/8.9/htmlhelp/pages/Manual/Algorithms/NPC_Algorithm.html.

Gunda Y., Gupta S. and L. Singh (2023). 'Assessing human performance and human reliability: a review'. In: *International Journal of System Assurance Engineering and Management* 14, pp. 817–828. DOI: 10.1007/s13198-023-01893-5.

Hannaman, G. W., A. J. Spurgin and Y. Lukic (1985). *Human Cognitive Reliability Model for PRA Analysis: Draft Report*. Tech. rep. NUS-4531, EPRI Project RP2170-3. San Diego, CA: NUS Corporation.

Health and Safety Executive (HSE) (1995). *Improving Compliance With Safety Procedures: Reducing Industrial Violations*. URL: https://www.hse.gov.uk/humanfactors/assets/docs/improvecompliance.pdf.

Heidarysafa M., Kowsari K. Barnes L. and D. Brown (2018). 'Analysis of Railway Accidents' Narratives Using Deep Learning'. In: *IEEE International Conference on Machine Learning and Applications (IEEE ICMLA)*. Orlando, FL, U.S., pp. 1446–1453. DOI: 10.1109/ICMLA.2018.00235.

Hendrycks, Dan et al. (2020). *Pretrained Transformers Improve Out-of-Distribution Robustness*. arXiv: 2004.06100 [cs.CL]. URL: https://arxiv.org/abs/2004.06100.

Hoffstaetter, S. (2023). *pytesseract - A Python wrapper for Google Tesseract*. URL: https://github.com/madmaze/pytesseract/graphs/contributors.

Hollman S., Frohmne M. Endrullat C. Kreme A. D'Elia D. Regierer B. and A. Nechyporenko (2020). 'Ten simple rules on how to write a standard operating procedure'. In: *PLoS Computational Biology* 16.9. DOI: 10.1371/journal.pcbi.1008095.

Hollnagel, E. (1998). *Cognitive Reliability and Error Analysis Method (CREAM)*. Elsevier Science. DOI: 10.1016/B978-0-08-042848-2.X5000-3.

– (2017). 'Why is Work-as-Imagined Different from Work-as-Done?' In: *Resilient Health Care*. Vol. 2, pp. 279–294. DOI: 10.1201/9781315605739.

Honnibal, M. and I. Montani (2017). *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing*. URL: https://spacy.io/.

Hudson, P. T., G. C. van der Graaf and W. L. Verschurr (1998). 'Perceptions of Procedures by Operators and Supervisors'. In: *SPE International Conference on Health, Safety, and Environment in Oil and Gas Exploration and Production*. Caracas, Venezuela. DOI: 10.2118/46760-MS.

Hughes, P., M. Figueres-Esteban and C. Van Gulijk (2016). 'From negative statements to positive safety'. In: *26th European Safety and Reliability Conference (ESREL 2016)*. Glasgow. DOI: 10.1201/9781315374987-286.

IAEA, International Atomic Energy Agency (1996). *Human Reliability Analysis in Probabilistic Safety Assessment for Nuclear Power Plants: A Safety Practice*. URL: https://www.iaea.org/publications/5200/human-reliability-analysis-in-probabilistic-safety-assessment-for-nuclear-power-plants-a-safety-practice.

Iliev, R., M. Dehghani and E. Sagi (2014). 'Automated Text Analysis in Psychology: Methods, Applications, and Future Developments'. In: *Language and Cognition*. DOI: 10.1017/langcog.2014.30.

Inc., The MathWorks (2023). *MATLAB (version R2023a)*. Natick, MA. URL: https://www.mathworks.com.

Institute, Energy (2020). *Guidance on human factors safety critical task analysis*. London. URL: https://publishing.energyinst.org/__data/assets/file/0012/697917/Pages-from-Guidance-on-human-factors-safety-critical-task-anlaysis-jk.pdf.

Intrator, Y. et al. (2024). 'Breaking the Language Barrier: Can Direct Inference Outperform Pre-Translation in Multilingual LLM Applications?' In: *arXiv*. DOI: 10.48550/arXiv.2403.04792.

IOGP (2024). *IOGP Safety Zone*. URL: https://safetyzone.iogp.org/Main.asp.

Isin, A. (2012). 'Standard Operating Procedures (What Are They Good For?)' In: *Latest Research into Quality Control*. DOI: 10.5772/50439.

ISO (2018). *Ergonomics of human-system interaction. Part 11: Usability: Definitions and concepts*. URL: https://www.iso.org/standard/63500.html.

James, Y. C. et al. (2014). 'The SACADA database for human reliability and human performance'. In: *Reliability Engineering & System Safety* 125, pp. 117–133. DOI: 10.1016/j.ress.2013.07.014.

Jang, I., Y. Kim and J. Park (2021). 'Investigating the Effect of Task Complexity on the Occurrence of Human Errors observed in a Nuclear Power Plant Full-Scope Simulator'. In: *Reliability Engineering & System Safety* 214. DOI: 10.1016/j.ress.2021.107704.

Jiao, Q. et al. (2024). 'Enhancing Multimodal Large Language Models with Vision Detection Models: An Empirical Study'. In: *arXiv*. DOI: 10.48550/arXiv.2401.17981.

Jing, X. et al. (2023). 'BERT for Aviation Text Classification'. In: *AIAA Aviation 2023 Forum*. San Diego. DOI: 10.2514/6.2023-3438.

Johnson, K., C. Morais and E. Patelli (2023a). 'AI Tools for Human Reliability Analysis'. In: *UNCECOMP 2023 - 5th ECCOMAS Thematic Conference on Uncertainty Quantification in Computational Sciences and Engineering*. Athens, Greece. DOI: 10.7712/120223.10348.20039.

– (2023b). 'Natural Language Processing Tool for Identifying Influencing Factors in Human Reliability Analysis and Summarizing Accident Reports'. In: *33rd European Safety and Reliability Conference (ESREL 2023)*. Southampton. DOI: 10.3850/978-981-18-8071-1_P294-cd.

– (2024). *Identifying Ambiguity And Potential Violations In Standard Operating Procedures Using Natural Language Processing Tools*. Cracow, Poland. URL: https://esrel2024.com/wp-content/uploads/articles/ea1/identifying-ambiguity-and-potential-violations-in-standard-operating-procedures-using-natural-language-processing-tools.pdf.

Johnson, K. et al. (2022). 'A Data Driven Approach to Elicit Causal Links between Performance Shaping Factors and Human Failure Events'. In: *32nd European Safety and Reliability Conference (ESREL 2022)*. Dublin, Ireland: Research Publishing, Singapore. DOI: 10.3850/978-981-18-5183-4_R12-12-372.

Jones, S., C. Kirchsteiger and W. Bjerke (1999). 'The importance of near miss reporting to further improve safety performance'. In: *Journal of Loss Prevention in the Process Industries* 12.1, pp. 59–67. DOI: 10.1016/S0950-4230(98)00038-2.

Jurafsky, D. and J. Martin (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall. URL: https://www.researchgate.net/publication/200111340_Speech_and_Language_Processing_An_Introduction_to_Natural_Language_Processing_Computational_Linguistics_and_Speech_Recognition.

Khanna, C. (2021). *Byte-Pair Encoding: Subword-based tokenization algorithm*. URL: https://towardsdatascience.com/byte-pair-encoding-subword-based-tokenization-algorithm-77828a70bee0.

Kim, S.-H., N. Lee and P. E. King (2020). 'Dimensions of Religion and Spirituality: A Longitudinal Topic Modeling Approach'. In: *Journal for the Scientific Study of Religion* 59.1. DOI: 10.1111/jssr.12639.

Kim, Yochan et al. (2022). 'Empirical study on human error probability of procedure-extraneous behaviors'. In: *Reliability Engineering and System Safety* 227, p. 108727. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2022.108727. URL: https://www.sciencedirect.com/science/article/pii/S0951832022003519.

Kingma, D. and J. Ba (2014). 'Adam: A Method for Stochastic Optimization'. In: *arXiv*. DOI: 10.48550/arXiv.1412.6980.

Kirwan, B. (1994). *A Guide to Practical Human Reliability Assessment*. London: CRC Press. DOI: 10.1201/9781315136349.

Kiyavitskaya, N. et al. (2008). 'Requirements for tools for ambiguity identification and measurement in natural language requirements specifications'. In: *Requirements Engineering* 13, pp. 207–239. DOI: 10.1007/s00766-008-0063-7.

KNKT (2019). *Aircraft Accident Investigation Final Report Boeing 737-8 (MAX) Lion Mentari Airlines KNKT.18.10.35.04*. URL: https://www.aaiu.ie/sites/default/files/FRA/2018%20-%20035%20-%20PK-LQP%20Final%20Report.pdf.

Kosko, T. and J. Noble (2009). *Bayesian Networks: An Introduction*. Wiley. DOI: 10.1002/9780470684023.

Kızılırmak, E. (2023). *Text Summarization: How To Calculate Rouge Score*. URL: https://medium.com/@eren9677/text-summarization-387836c9e178.

Lakretz, Y., S. Dehaene and J.-K. King (2020). 'What Limits Our Capacity to Process Nested Long-Range Dependencies in Sentence Comprehension?' In: *Entropy* 22.4. DOI: 10.3390/e22040446.

Laumann, K. (2020). 'Criteria for qualitative methods in human reliability analysis'. In: *Reliability Engineering & System Safety* 194. DOI: 10.1016/j.ress.2018.07.001.

Lavie, A. and A. Agarwal (2007). 'METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments'. In: *Proceedings of the Second Workshop on Statistical Machine Translation*. Prague, Czechia: Association for Computational Linguistics, pp. 228–231. URL: https://aclanthology.org/W07-0734.

Lawton, R. (1998). 'Not working to rule: Understanding procedural violations at work'. In: *Safety Science* 28.2, pp. 77–95. DOI: 10.1016/S0925-7535(97)00073-8.

Leung, K. (2022). *Micro, Macro & Weighted Averages of F1 Score, Clearly Explained*. URL: https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f.

Leveson, N. (2020). *Safety III: A Systems Approach to Safety and Resilience*. URL: http://sunnyday.mit.edu/safety-3.pdf.

Leveson, Nancy G. (2011). *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, Massachusetts; London, England: The MIT Press. ISBN: 978-0-262-01662-9.

Lewis, M. et al. (2019). 'BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension'. In: *arXiv*. DOI: 10.48550/arXiv.1910.13461.

Li, G. (2009). *NPC algorithm for learning DAG in Bayesian Network*. URL: https://www.mathworks.com/matlabcentral/fileexchange/24456-npc-algorithm-for-learning-dag-in-bayesian-network.

Li, M. Y. et al. (2024a). 'A Taxonomy of Ambiguity Types for NLP'. In: *arXiv*. DOI: 10.48550/arXiv.2403.14072.

Li, Y. et al. (2008). 'Regular Expression Learning for Information Extraction'. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*. Honolulu. DOI: 10.3115/1613715.1613719.

Li, Z. et al. (2024b). 'Quantifying Multilingual Performance of Large Language Models Across Languages'. In: *arXiv*. DOI: 10.48550/arXiv.2404.11553.

Lima, N. E. et al. (2019). 'A Methodology to Use Multi-Objective Optimization Criteria for an Offshore Topside Production System Since the Early Design Stages, and for The Unit Life Cycle'. In: *29th European Safety and Reliability Conference (ESREL 2019)*. Hannover, Germany. DOI: 10.3850/978-981-11-2724-3_0675-cd.

Lin, C.-Y. (2004). 'ROUGE: A Package for Automatic Evaluation of Summaries'. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, pp. 74–81. URL: https://aclanthology.org/W04-1013.

Lin, T. et al. (2022). 'A Survey of Transformers'. In: *AI Open* 3, pp. 111–132. DOI: 10.1016/j.aiopen.2022.10.001.

Liu, Bo et al. (2024). *How Good Are LLMs at Out-of-Distribution Detection?* arXiv: 2308.10261 [cs.CL]. URL: https://arxiv.org/abs/2308.10261.

Liu, Y. et al. (2023). 'On Learning to Summarize with Large Language Models as References'. In: *arXiv*. DOI: 10.48550/arXiv.2305.14239.

Macedo, J. et al. (2023). 'Human Factor Identification in Aviation Accidents Using Contextual Word Embeddings'. In: *33rd European Safety and Reliability Conference*. Southampton. DOI: 10.3850/978-981-18-8071-1_P233-cd.

Macêdo, J. et al. (2022). 'Identification of risk features using text mining and BERT-based models: Application to an oil refinery'. In: *Process Safety and Environmental Protection* 158, pp. 382–399. DOI: 10.1016/j.psep.2021.12.025.

Manghani, K. (2011). 'Quality assurance Importance of systems and standard operating procedures'. In: *Perspectives in Clinical Research* 2.1, pp. 34–37. DOI: 10.4103/2229-3485.76288.

Maples, S. (2017). *The ROUGE-AR: A Proposed Extension to the ROUGE*. URL: https://api.semanticscholar.org/CorpusID:34483154.

Marks, S. and A. L. Dahir (2020). *Ethiopian Report on 737 Max Crash Blames Boeing*. URL: https://www.nytimes.com/2020/03/09/world/africa/ethiopia-crash-boeing.html.

McCallum, A. and K. Nigam (1998). 'A Comparison of Event Models for Naive Bayes Text Classification'. In: *AAAI Conference on Artificial Intelligence*. URL: https://api.semanticscholar.org/CorpusID:7311285.

Merlo, P. and E. E. Ferrer (2006). 'The Notion of Argument in Prepositional Phrase Attachment'. In: *Computational Linguistics* 32.3, pp. 341–378. DOI: 10.1162/coli.2006.32.3.341.

Miller, G. (1995). 'WordNet: A Lexical Database for English'. In: *Communications of the ACM* 38.11, pp. 39–41. DOI: 10.1145/219717.219748.

Mkrtchyan, L., L. Podofillini and V. Dang (2015). 'Bayesian belief networks for human reliability analysis: A review of applications and gaps'. In: *Reliability Engineering & System Safety* 139, pp. 1–16. DOI: 10.1016/j.ress.2015.02.006.

Morais, C. et al. (2020). 'Analysis and Estimation of Human Errors From Major Accident Investigation Reports'. In: *ASCE-ASME J Risk and Uncertainty in Engineering Systems Part B: Mechanical Engineering* 6.1. DOI: 10.1115/1.4044796.

Morais, C. et al. (2022a). 'Identification of human errors and influencing factors: A machine learning approach'. In: *Safety Science* 146. DOI: 10.1016/j.ssci.2021.105528.

Morais, C. et al. (2022b). 'Robust data-driven human reliability analysis using credal networks'. In: *Reliability Engineering & System Safety* 218. DOI: 10.1016/j.ress.2021.107990.

Mosleh, A., V. Bier and G. Apostolakis (1988). 'A critique of current practice for the use of expert opinions in probabilistic risk assessment'. In: *Reliability Engineering & System Safety* 20.1, pp. 63–85. DOI: 10.1016/0951-8320(88)90006-3.

Mosleh, A. and K. Groth (2012). 'Deriving causal Bayesian networks from human reliability analysis data: A methodology and example model'. In: *Proceedings of the Institution of Mechanical Engineers, Journal of Risk and Reliability* 226.4, pp. 361–379. DOI: 10.1177/1748006X11428107.

Moura, R. (2023). 'Human Errors in Engineering'. In: *Understanding Human Errors in Construction Industry*. Springer, pp. 1–7. DOI: 10.1007/978-3-031-37667-2_1.

Moura, R. et al. (2016). 'Learning from major accidents to improve system design'. In: *Safety Science* 84, pp. 37–45. DOI: 10.1016/j.ssci.2015.11.022.

Moura, R. et al. (2017a). 'Learning from accidents: Interactions between human factors, technology and organisations as a central element to validate risk studies'. In: *Safety Science*, pp. 196–214. DOI: 10.1016/j.ssci.2017.05.001.

– (2017b). 'Learning from major accidents: Graphical representation and analysis of multi-attribute events to enhance risk communication'. In: *Safety Science* 99.2, pp. 58–70. DOI: 10.1016/j.ssci.2017.03.005.

Ng, J.-P. A. (2015). 'Better Summarization Evaluation with Word Embeddings for ROUGE'. In: *arXiv*. DOI: 10.48550/arXiv.1508.06034.

NRC, U.S. (2006). *Evaluation of Human Reliability Analysis Methods Against Good Practices (NUREG-1842)*. Albuquerque, N.M., U.S. URL: https://www.nrc.gov/reading-rm/doc-collections/nuregs/staff/sr1842/index.html.

Nunes, J. et al. (2018). 'Issues in the Probability Elicitation Process of Expert-Based Bayesian Networks'. In: *Enhanced Expert Systems*. Ed. by P. Vizureanu. Rijeka, Croatia: IntechOpen. DOI: 10.5772/intechopen.81602.

Oehm, D. (2019). *Simulating data with Bayesian networks*. URL: https://gradientdescending.com/simulating-data-with-bayesian-networks/.

Office, Home (2007). *Guidelines for Standard Operating Procedures (SOPs)*. URL: https://www.gov.uk/government/publications/standard-operating-procedure-guidelines.

Oliaee, A. et al. (2023). 'Using Bidirectional Encoder Representations from Transformers (BERT) to classify traffic crash severity types'. In: *Natural Language Processing Journal* 3. DOI: 10.1016/j.nlp.2023.100007.

Oram, P. (2001). 'WordNet: An electronic lexical database'. In: *Applied Psycholinguistics*. Ed. by C. Fellbaum. Vol. 22. Cambridge University Press, pp. 131–134. DOI: 10.1017/S0142716401221079.

Papineni, K. et al. (2002). 'BLEU: a Method for Automatic Evaluation of Machine Translation'. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL 2002)*. Philadelphia, PA: Association for Computational Linguistics, pp. 311–318. DOI: 10.3115/1073083.1073135.

Park, J. et al. (2004). 'Analysis of operators' performance under emergencies using a training simulator of the nuclear power plant'. In: *Reliability Engineering & System Safety* 83.2, pp. 179–186. DOI: 10.1016/j.ress.2003.09.009.

Patelli, E. (2016). 'COSSAN: A Multidisciplinary Software Suite for Uncertainty Quantification and Risk Management'. In: *Handbook of Uncertainty Quantification*. DOI: 10.1007/978-3-319-11259-6_59-1.

Patil, R. et al. (2023). 'A Survey of Text Representation and Embedding Techniques in NLP'. In: *IEEE Access* 1. DOI: 10.1109/ACCESS.2023.3266377.

Peters, M. E. et al. (2018). 'Deep contextualized word representations'. In: *arXiv*. DOI: 10.48550/arXiv.1802.05365.

Phipps, Denham L., Paul C.W. Beatty and Dianne Parker (2015). 'Standard deviation? The role of perceived behavioural control in procedural violations'. In: *Safety Science* 72, pp. 66–74. ISSN: 0925-7535. DOI: https://doi.org/10.1016/j.ssci.2014.08.009. URL: https://www.sciencedirect.com/science/article/pii/S0925753514001982.

Podofillini, L. and V. N. Dang (2013). 'A Bayesian approach to treat expert-elicited probabilities in human reliability analysis model construction'. In: *Reliability Engineering & System Safety* 117, pp. 52–64. DOI: 10.1016/j.ress.2013.03.015.

Podofillini, Luca, Bernhard Reer and Vinh N. Dang (2021). 'Analysis of recent operational events involving inappropriate actions: influencing factors and root causes'. In: *Reliability Engineering and System Safety* 216, p. 108013. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2021.108013. URL: https://www.sciencedirect.com/science/article/pii/S0951832021005226.

Porter, S. (2023). *Understanding Cosine Similarity and Word Embeddings*. URL: https://spencerporter2.medium.com/understanding-cosine-similarity-and-word-embeddings-dbf19362a3c.

Qasim, R. et al. (2022). 'A Fine-Tuned BERT-Based Transfer Learning Approach for Text Classification'. In: *Journal of Healthcare Engineering* 1. DOI: 10.1155/2022/3498123.

Ramos, P. et al. (2022). 'Combining BERT with numerical variables to classify injury leave based on accident description'. In: *Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*. DOI: 10.1177/1748006X22114019.

Ratsaby, J. and S. S. Venkatesh (1995). 'Learning From A Mixture Of Labeled And Unlabeled Examples With Parametric Side Information'. In: *Proceedings of the eighth annual conference on Computational learning theory*. DOI: 10.1145/225298.225348.

Rayson, P. (2009). *Wmatrix: a web-based corpus processing environment*. URL: https://ucrel.lancs.ac.uk/wmatrix/.

Reason, J., D. Parker and R. Lawton (1998). 'Organizational controls and safety: The varieties of rule-related behaviour'. In: *Journal of Occupational and Organization Psychology* 71.4, pp. 289–304. DOI: 10.1111/j.2044-8325.1998.tb00678.x.

Ribeiro, L. C. et al. (2020). 'Evolving Neural Conditional Random Fields for drilling report classification'. In: *Journal of Petroleum Science and Engineering* 187. DOI: 10.1016/j.petrol.2019.106846.

Robinson, R. D. et al. (2015). 'Application of machine learning to mapping primary causal factors in self-reported safety narratives'. In: *Safety Science* 75, pp. 118–129. DOI: 10.1016/j.ssci.2015.02.003.

Salazar, J. et al. (2020). 'Masked Language Model Scoring'. In: *58th Annual Meeting of the Association for Computational Linguistics*, pp. 2699–2712. DOI: 10.48550/arXiv.1910.14659.

Sarkar, S. and J. Maiti (2020). 'Machine learning in occupational accident analysis: A review using science mapping approach with citation network analysis'. In: *Safety Science* 131. DOI: 10.1016/j.ssci.2020.104900.

Scutari, M., C. Graafland and J. Gutierrez (2019). 'Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms'. In: *International Journal of Approximate Reasoning* 115, pp. 235–253. DOI: 10.1016/j.ijar.2019.10.003.

Sedlar, Nejc et al. (2023). 'A qualitative systematic review on the application of the normalization of deviance phenomenon within high-risk industries'. In: *Journal of Safety Research* 84, pp. 290–305. ISSN: 0022-4375. DOI: https://doi.org/10.1016/j.jsr.2022.11.005. URL: https://www.sciencedirect.com/science/article/pii/S0022437522001827.

Sennrich, R., B. Haddow and A. Birch (2015). 'Neural Machine Translation of Rare Words with Subword Units'. In: *arXiv*. DOI: 10.48550/arXiv.1508.07909.

Sharma, A. (2024). *How Part-of-Speech Tag, Dependency and Constituency Parsing Aid In Understanding Text Data?* URL: https://www.analyticsvidhya.com/blog/2020/07/part-of-speechpos-tagging-dependency-parsing-and-constituency-parsing-in-nlp/.

Sharma, P. (2023). 'Advancements in OCR: A Deep Learning Algorithm for Enhanced Text Recognition'. In: *International Journal of Inventive Engineering and Sciences* 10.8, pp. 1–7. DOI: 10.35940/ijies.F4263.0810823.

Shen, K. et al. (2023). 'A Study on ReLU and Softmax in Transformer'. In: *arXiv*. DOI: 10.48550/arXiv.2302.06461.

Shi, H. and Y. Liu (2011). 'Naïve Bayes vs. Support Vector Machine: Resilience to Missing Data'. In: *Artificial Intelligence and Computational Intelligence - Third International Conference, AICI 2011*. Taiyuan, China. DOI: 10.1007/978-3-642-23887-1_86.

Shung, K. P. (2018). *Accuracy, Precision, Recall or F1?* URL: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

Siegrist, J. (2012). 'Mixing Good Data with Bad: How to Do It and When You Should Not'. In: *Vulnerability, Uncertainty, and Risk: Analysis, Modeling, and Management*. DOI: 10.1061/41170(400)45.

Smith, R. (2007). 'An Overview of the Tesseract OCR Engine'. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Curitiba, Brazil, pp. 629–633. DOI: 10.1109/ICDAR.2007.4376991.

Song, X. et al. (2020). 'Fast WordPiece Tokenization'. In: *arXiv*. DOI: 10.48550/arXiv.2012.15524.

Steck, H. (2001). *Constraint-based structural learning in Bayesian networks using finite data sets*. URL: https://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss2001051816888.

Steen, Riana et al. (2024). 'Dark knights: Exploring resilience and hidden workarounds in commercial aviation through mixed methods'. In: *Safety Science* 175, p. 106498. ISSN: 0925-7535. DOI: https://doi.org/10.1016/j.ssci.2024.106498. URL: https://www.sciencedirect.com/science/article/pii/S0925753524000882.

Steen-Tveit, Kristine, Bjørn Erik Munkvold and Kjetil Rustenberg (Oct. 2024). 'Use of Standard Operating Procedures for Supporting Cross-Organizational Emergency Management: Challenges and Opportunities Identified from a Tabletop Exercise'. In: *International Journal of Disaster Risk Science*. ISSN: 21926395. DOI: 10.1007/s13753-024-00583-5.

Sujan, M. A., D. Embrey and H. Huang (2020). 'On the application of Human Reliability Analysis in healthcare: Opportunities and challenges'. In: *Reliability Engineering & System Safety* 194. DOI: 10.1016/j.ress.2018.06.017.

Sun, Hao, Ming Yang and Haiqing Wang (Mar. 2024). 'An integrated approach to quantitative resilience assessment in process systems'. In: *Reliability Engineering and System Safety* 243. ISSN: 09518320. DOI: 10.1016/j.ress.2023.109878.

Sun, Z. et al. (2012). 'Estimating Human Error Probability using a modified CREAM'. In: *Reliability Engineering & System Safety* 100, pp. 28–32. DOI: 10.1016/j.ress.2011.12.017.

Swain, A. D. and H. E. Guttmann (1983). *Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications: Final Report*. Tech. rep. NUREG/CR-1278, SAND80-0200. Prepared for the United States Department of Energy. Albuquerque, New Mexico and Livermore, California: Sandia National Laboratories.

Tabaie, A. et al. (2023). 'A natural language processing approach to categorise contributing factors from patient safety event reports'. In: *BMJ Health & Care Informatics*. DOI: 10.1136/bmjhci-2022-100731.

Team, Pandas Dev (2024). *Pandas-v2.2.2*. DOI: 10.5281/zenodo.10957263.

TensorFlow (2015). *TensorFlow*. URL: https://www.tensorflow.org/.

Traffis, C. (2020). *What Is a Subordinate Clause?* URL: https://www.grammarly.com/blog/subordinate-clause/#:~:text=A%20subordinate%20clause%20is%20a,to%20as%20a%20dependent%20clause..

Trinh, V. et al. (2023). 'The use of natural language processing in detecting and predicting falls within the healthcare setting: a systematic review'. In: *International Journal for Quality in Health Care* 35.4. DOI: 10.1093/intqhc/mzad077.

Valcamonico, Dario et al. (Jan. 2024). 'Combining natural language processing and bayesian networks for the probabilistic estimation of the severity of process safety events in hydrocarbon production assets'. In: *Reliability Engineering and System Safety* 241. ISSN: 09518320. DOI: 10.1016/j.ress.2023.109638.

Vaswani, A. et al. (2017). 'Attention Is All You Need'. In: *arXiv*. DOI: 10.48550/arXiv.1706.03762.

Venkataramana, A., K. Srividya and R. Cristin (2022). 'Abstractive Text Summarization Using BART'. In: *IEEE*. Seoul. DOI: 10.1109/MysuruCon55714.2022.9972639.

Vinnem, J. (2018). 'FPSO Cidade de São Mateus gas explosion – Lessons learned'. In: *Safety Science* 101. DOI: 10.1016/j.ssci.2017.09.021.

Wang, J. et al. (2024). 'Utilizing BERT for Information Retrieval: Survey, Applications, Resources, and Challenges'. In: *ACM Computing Surveys* 56.7, pp. 1–33. DOI: 10.1145/3648471.

Wang, S. and C. D. Manning (2012). 'Baselines and Bigrams: Simple, Good Sentiment and Topic Classification'. In: *Proceedings of 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, pp. 90–94. URL: https://aclanthology.org/P12-2018.

Waykole, R. and A. Thakare (2018). 'A Review of Feature Extraction Methods for Text Classification'. In: *International Journal of Applied Engineering Research and Development*.

Yadav, H., N. Patel and D. Jani (2023). 'Fine-Tuning BART for Abstractive Reviews Summarization'. In: *Computational Intelligence*, pp. 375–385. DOI: 10.1007/978-981-19-7346-8_32.

Yamamoto, Y., Y. Matsumoto and T. Watanabe (2022). 'Dependency Patterns of Complex Sentences and Semantic Disambiguation for Abstract Meaning Representation Parsing'. In: *Journal of Natural Language Processing* 29.2, pp. 515–541. DOI: 10.5715/jnlp.29.515.

Yazgan, E. and D. E. (2024). 'Human Error Assessment and Reduction Technique in Aircraft Maintenance'. In: *Solutions for Maintenance Repair and Overhaul. ISATECH 2021. Sustainable Aviation*. Ed. by T. Karakoc et al. Springer. DOI: 10.1007/978-3-031-38446-2_12.

Zhang, M. et al. (2022). 'A Comprehensive Survey of Abstractive Text Summarization Based on Deep Learning'. In: *Computational Intelligence and Neuroscience* 1. DOI: 10.1155/2022/7132226.

Zhang, M. et al. (2024). 'ROUGE-SEM: Better evaluation of summarization using ROUGE combined with semantics'. In: *Expert Systems with Applications* 237. DOI: 10.1016/j.eswa.2023.121364.

Zhang, W., T. Yoshida and X. Tang (2008). 'Text classification based on multi-word with support vector machine'. In: *Knowledge-Based Systems* 21.8, pp. 879–886. DOI: 10.1016/j.knosys.2008.03.044.

Zhao, Z. et al. (2020). 'Summarization of Coal Mine Accident Reports: A Natural-Language-Processing-Based Approach'. In: *Cyberspace Data and Intelligence, and Cyber-Living, Syndrome, and Health*, pp. 103–115. DOI: 10.1007/978-981-33-4336-8_9.

Zhu, C. et al. (2021). 'Enhancing Factual Consistency of Abstractive Summarization'. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. DOI: 10.48550/arXiv.2003.08612.

Zmandar, N., E.-H. M. and P. Rayson (2023). 'A Comparative Study of Evaluation Metrics for Long-Document Financial Narrative Summarization with Transformers'. In: *Natural Language Processing and Information Systems (NLDB 2023)*. Vol. 13913. Springer, pp. 391–403. DOI: 10.1007/978-3-031-35320-8_28.

Zubrinic, K., M. Milicevic and I. Zakarija (2013). 'Comparison of Naive Bayes and SVM'. In: *International Journal of Computers* 7, pp. 109–116.

Appendix A

Further Methodological Details and Supplementary Code

This appendix provides additional methodological details and supplementary code for each of the tools presented in the main body of the thesis.

A.1 Virtual Human Factors Classifier (First Generation)

The first-generation *Virtual HF Classifier* automates the identification of contributing human errors and PSFs in accident reports, presented in Chapter 3. The classifier used a BoW representation of preprocessed text data and applied linear SVM models for classification.

Reports were matched with labels using a classification table derived from the MATA-D dataset. Important textual sections are extracted using rule-based keyword detection, before preprocessing which includes punctuation removal, tokenization, stop word elimination, and word normalization.

It was implemented in MATLAB and makes us of the built-in functions and toolboxes given in Table A.1.

Function(s)	Toolbox / Source
`tokenizedDocument`, `removeWords`, `normalizeWords`, `bagOfWords`, `removeInfrequentWords`, `encode`	Text Analytics Toolbox
`fitcecoc`, `cvpartition`	Statistics and Machine Learning Toolbox
`readtable`, `xlsread`	Built-in Spreadsheet Functions
`extractFileText`	Built-in (PDF Text Extraction)
`regexprep`, `strsplit`, `contains`, `lower`, `isempty`	Core MATLAB String Functions

Table A.1: Functions and their required toolboxes or sources.

A.1.1 Report-Label Matching and Extraction

This script serves as the starting point, by aligning accident reports with labels from the MATA-D classification table. It attempts to match each PDF with an entry in the MATA-D based on the available metadata (location, year), before extracting the text from the matching files and attaching the corresponding CREAM labeling.
The key operations include,

- Cleaning metadata (location, year) and handling fuzzy string matches.
- Extracting text from the PDFs using `extractFileText`.
- Section filtering using `getImportantSection.m`

The function outputs a composite cell array, where each row represents a report and includes, a report ID, filename, extracted text, and the associated classification labels.

A.1.1.1 Metadata Cleaning (Location and Year)

The code first cleans metadata by trimming the header and selecting relevant columns from the Excel sheet. It processes the location strings by removing punctuation and normalizing spacing. Any `NaN` values are replaced with zero.

```
function [output] = dataCompiler(folderDir,getImpSection)
% Read Classification Table
    [numTable,stringTable,~] = xlsread("MATA-D.xlsx","CREAM Categories");
% Data clean up
    stringTable = stringTable(5:end,2:3);
    splitLocNames = {};
    numTable = numTable(:,[1 4:end]);
    numTable(isnan(numTable)) = 0;
% Create splitLocNames by removing punctuations and multiple spaces
    for i = 1:size(stringTable,1)
        currentLoc = regexprep(regexprep(stringTable(i,2),
            '[^0-9a-zA-Z]',' '),' +',' ');
        splitLocNames{i} = strsplit(string(currentLoc{1}),' ');
    end
```

A.1.1.2 Matching Metadata to PDF Files

The function then loops through each PDF filename to determine if it matches an entry in the metadata using criteria such as location string match, 4-digit year match, and filtering with an ignore word list. The matching algorithm uses a scoring mechanism based based on these factors, if the match confidence exceeds a threshold (3 or more), the file is considered successfully matched.

```
% Try to match PDFs with Classification Table rows
    docDirectory = {};
    for i = 1:length(reports)
      currentName = reports(i).name;
```

```
        ...
    for j = 1:size(numTable,1)
          ...
            if contains(lower(currentName),lower(stringTable(j,2)))
            matchConfidence = matchConfidence + 1;
           end
          ...
            if matchConfidence >= 3 && wrongYearConfidence == false
               docDirectory = [docDirectory {numTable(j,1)
               ;string(currentName);matchConfidence}];
         end
        end
    end
```

A.1.2 Targeted Section Filtering

This function identifies and extracts the most informative part of a report, typically sections titled “Recommendations,” “Lessons Learned,” or similar. The goal is to focus classification only on sections most likely to contain relevant insights. This script,

- Scans all lines and assigns confidence scores to those matching target phrases.
- Identifies valid start-end pairs while attempting to avoid common non-informative sections (e.g., “Appendices” or “References”).
- Selects the highest confidence and longest section for use.

This is done to improve the classifier performance and reduce noise by ignoring irrelevant report content like legal disclaimers or background information.

A.1.2.1 Splitting and Set Up

The document is first split by newline characters into a list of individual lines for analysis.

```
lineArray = strsplit(inputDocument,'\n');
```

Within the function categories of keywords are defined,

```
startTargetWords = ["recommendation", "lessons learned",
\\ "advice to planning authorities"];
endTargetWords = ["reference", "appendix", "annex", "conclusion", ...];
lesserEndTargetWords = ["accordingly", "section", "\figure"];
```

A.1.2.2 Identifying Candidate Start and End Lines

The function `searchForLineNumsAndConfidence` analyzes each line in the document and assigns a confidence score based on keyword prominence and structural patterns.

```
[startLineNumsAndConfidence] = searchForLineNumsAndConfidence(lineArray,
startTargetWords, []);
[endLineNumsAndConfidence] = searchForLineNumsAndConfidence(lineArray,
endTargetWords, lesserEndTargetWords);
```

Each line is cleaned and evaluated as follows,

1. Characters are restricted to letters, digits, hyphens, commas, and periods.
2. If the line contains a target keyword, a base confidence of 2 is awarded.
3. The score is adjusted depending on how much of the line the keyword occupies,
   - 0 points - if the keyword is less than 25% of the line.
   - +1 point - if over 50%.
   - +1 more - if over 70%.

4. If the keyword is “appendix” and followed by a single character (“Appendix A”), an additional point is awarded.
5. Lines with apparent section numbers (“3.2 Recommendations”) get +1.
6. Lines with multiple keyword matches or illegal characters are disqualified (confidence set to 0).
7. Lines surrounded by empty lines (above or below) receive a contextual bonus of +1 for each side.

For example,

```
currentLineArray = regexp(lower(lineArray(i)),'[a-zA-Z0-9-,.]+','match');
currentLine = strjoin(currentLineArray);
if contains(currentLine, targetWords(1,j))
    targetWordsConfidence = targetWordsConfidence + 2;
    ...
    if targetWordLength / currentLineLength > 0.5
        targetWordsConfidence = targetWordsConfidence + 1;
    end
```

A similar but less strict scoring scheme is used for the lesser target words.

A.1.2.3 Selecting Best Section Start and End

From all candidates, the function selects the most confident starting lines using,

```
startRows = getMaxRows(startLineNumsAndConfidence, 2);
```

It then searches for a matching end line that follows each start line. If no strong end is found (confidence is less than 4), it defaults to the end of the document.

```
if isempty(endsForCurrentStart)
    startRows(i,4) = size(lineArray,2);
else
```

```
    startRows(i,4) = endsForCurrentStart(1,1) - 1;
end
```

Each candidate start-end pair is scored by the total number of characters in the range. This helps prefer longer, richer sections.

A.1.2.4 Excluding Table of Contents Matches

To avoid selecting headings from a Table of Contents a check is implemented within the function. If a start line is within 20 lines of a Table of Contents heading, it is discarded.

```
[tableOfContentsNumsAndConfidence] = searchForLineNumsAndConfidence
(lineArray, ["table of content"], ["content"]);
...
if likelyToC(j,1) < startRows(i,1) && likelyToC(j,1) + 20 > startRows(i,1)
    startRowsToRemove = [startRowsToRemove i];
end
```

A.1.2.5 Important Section Output

Among the valid candidates, the section with the most characters is selected as the final output,

```
longestPossibleSectionsInfo = getMaxRows(startRows, 5);
outputSection = lineArray(finalStartLineNum:finalEndLineNum);
```

This enables reliable extraction of meaningful sections from diverse document formats using a combination of lexical cues, structural patterns, and scoring heuristics.

A.1.3 Text Preprocessing

This script prepares raw report text for vectorization. It performs the following steps,

1. Punctuation Removal, using `erasePunctuation()`
2. Tokenization, using `tokenizedDocument()` to structure the text.
3. Stop Word Removal, common non-informative words are discarded.
4. Word Length Filtering, extremely short or long tokens are removed.
5. Word Normalization, stemming/lemmatization to reduce inflected forms.

Using the built-in functions from the Text Analytics Toolbox,

```
output = erasePunctuation(input);
output = lower(output);
output = tokenizedDocument(output);
output = removeWords(output, stopWords);
output = removeShortWords(output, 2);
output = removeLongWords(output, 15);
output = normalizeWords(output);
```

This standardization is critical to ensure consistency and reduce vocabulary sparsity before creating the BoW representation.

A.1.4 Classifier Training and Evaluation

This script drives the model training and evaluation loop. The key steps are,

1. Compile labeled data using `dataCompiler.m`.
2. Preprocess text and convert to BoW model.
3. Partition data into training/test sets using `cvpartition`.
4. Train a classifier for each CREAM label using `fitcecoc`.

5. Predict on test data and calculate performance metrics.
6. Prepare new report, apply trained model and save predictions.

A.1.4.1 Preparing Data, BoW Model, Test and Training Sets

The process begins by compiling labeled text data using the function `dataCompiler.m`.

```
folderDir = pwd + "/Reports";  % Directory containing all reports
getImpSection = false;         % Flag for extracting a specific section (
    optional)
classTable = dataCompiler(folderDir, getImpSection);
```

The result, `classTable`, is a table where:

- Columns 1-3: Metadata (file name, source)
- Column 3: Narrative text data
- Columns 4 onward: Binary labels for CREAM factors

Next, the dataset is partitioned into training and testing subsets using `cvpartition`. The extracted text is then preprocessed and converted into a BoW model.

```
textDataTrain = [dataTrain{:,3}]';
documents = preprocess(textDataTrain);  % Custom function to clean text
bag = bagOfWords(documents);            % Convert to BoW
bag = removeInfrequentWords(bag, 2);    % Remove rare terms
[bag, idx] = removeEmptyDocuments(bag); % Remove empty docs
XTrain = bag.Counts;                    % Feature matrix
```

The resulting BoW model is filtered to eliminate infrequent and empty documents. These steps help reduce dimensionality and noise in the input space.

The data is then split into training and test subsets,

```
cvp = cvpartition(cell2mat(newClassTable(:,4,:)), 'Holdout', 0.1);
```

```
dataTrain = newClassTable(cvp.training,:);
dataTest  = newClassTable(cvp.test,:);
YTrain = cell2mat(dataTrain(:,4:end));
YTest  = cell2mat(dataTest(:,4:end));
YTrain(idx,:) = [];  % Match filtered training docs
```

A.1.4.2 Training the Classifier

The classification model used is a linear SVM wrapped in MATLAB's `fitcecoc` function, which supports multiclass learning via error-correcting output codes. A separate classifier is trained for each CREAM factor.

```
mdlArray = {};
for i = 1:size(YTrain, 2)
    mdl = fitcecoc(XTrain, YTrain(:,i), 'Learners', 'linear');
    mdlArray{end+1, 1} = mdl;
    YPred = predict(mdl, XTest);
    YPreds(:,i) = YPred;
end
```

Each column in `YTrain` corresponds to a binary label for a CREAM category. The loop trains one classifier per label and stores the models in `mdlArray`. Predictions for the test set are collected in `YPreds`.

A.1.4.3 Testing and Performance Metrics

After training, predictions are compared with the ground truth using a custom function `accmetrics.m`.

```
    function [Accuracy,Precision,Recall,F1Score] = accmetrics(x,y)
        TrueP=0; TrueN=0; FalseP=0; FalseN=0;
```

```
    for i = 1:size(x,1)
    for j= 1:size(x,2)
        if x(i,j)==1
            if y(i,j)==1
                TrueP=TrueP+1;
            else
                FalseP=FalseP+1;
            end
        elseif x(i,j)==0
        if y(i,j)==0
    TrueN=TrueN+1;
        else
        FalseN=FalseN+1;
    end end end end
        Accuracy = (TrueP+TrueN)/(TrueP+TrueN+FalseP+FalseN);
        Precision =TrueP/(TrueP+FalseP);
        Recall= TrueP/(TrueP+FalseN);
        F1Score=2*(Precision*Recall)/(Precision+Recall);
end
```

This returns the standard classification performance metrics.

A.1.4.4 Application to a New Report

The trained classifiers are then applied to a new, unseen incident report. The user uploads a PDF file, which is read, preprocessed, and encoded using the previously generated BoW vocabulary.

```
    parentdir = pwd + "/Uploads";
    uploads = dir(fullfile(parentdir, "*.pdf"));
    filepath = parentdir + "/" + uploads.name;
    temp = strsplit(extractFileText(filepath), "\n");  % Extract raw text
    str = strjoin(temp);
    documentsNew = preprocess(str); % Clean text
    XNew = encode(bags{1,1}, documentsNew); % Convert to BoW
```

The vectorized document is passed to each trained model for prediction,

```
for j = 1:size(mdlArray,1)
 labelsNew = predict(mdlArray{j}, XNew);
 result1(j,:) = labelsNew;
end
```

The predicted labels are converted into readable strings and saved to a text file,

```
[Out, ~] = print_string(result1);
OutString = strrep(string(Out(2:end,1)), '_', ' ');
fid = fopen([savepath + "/Results/" + 'factors.txt'], 'wt');
fprintf(fid, '%s\n', OutString);
fclose(fid);
```

This concludes the complete workflow of the first-generation *Virtual HF classifier*. The complete code is available at `https://github.com/VirtualRaphael/Human-Factors-Classifier-1.0`.

A.2 Virtual Human Factors Classifier (Second Generation)

The second-generation, *HF Classifier 2.0* leverages transformer-based models to automate the identification of PSFs from accident reports, presented in Chapter 4, Section 4.2. Each PSF is treated as an independent binary classification task. Unlike the first-generation BoW and SVM model approach, this version incorporates semantic understanding via BERT and addresses common NLP challenges such as sequence length truncation and model scalability.

The process is composed of two primary components,

- A modular training pipeline (`HF_Classifier_2.0_Training.py`)
- A robust inference system for new reports (`HF_Classifier_2.0_NewReport.py`)

A.2.1 Model Training Pipeline

This section will detail the model training pipeline of the *HF Classifier 2.0*, detailing the configuration, text chunking, data manipulation, logit aggregation, training loop and model saving. Similar to the first-generation classifier target sections within the reports are extracted using a keyword matching system, and are paired with the corresponding entries in the MATA-D.

A.2.1.1 Text Chunking

To accommodate BERT's input length limit of 512 tokens, the extracted accident descriptions are split into token chunks. The following function splits each accident description into overlapping chunks.

```
def chunk_text(tokenizer, text, max_len):
    # Tokenize the input text
    tokens = tokenizer.tokenize(text)
    # Initialize a list to hold the resulting text chunks
    chunks = []
    # Loop over the token list in steps of `max_len`
    for i in range(0, len(tokens), max_len):
        # Extracttokens with length up to `max_len`
        chunk = tokens[i:i + max_len]
        # Convert token chunk back to a text string
        chunk = tokenizer.convert_tokens_to_string(chunk)
        # Append the string chunk to the list
        chunks.append(chunk)
    # Return the list of text chunks
    return chunks
```

For every chunk, token IDs, attention masks, and associated labels are generated, along with a report identifier,

```
def chunk_reports(df, label_col, tokenizer, max_len):
    # Initialize lists to hold the results
    input_ids, attention_masks, labels, report_ids = [], [], [], []
    # Iterate over the rows of the DataFrame with an index
    for report_id
    # Extract and chunk the accident description
        chunks = chunk_text(tokenizer, getattr(row, "Accident Description")
            , max_len - 2)
     # Tokenize the chunks together, applying truncation and padding
        encoding = tokenizer(chunks, truncation=True, padding=True,
                             return_tensors='pt', max_length=max_len)

    # Append token IDs and attention masks
        input_ids.append(encoding['input_ids'])              # Shape: (
            num_chunks, max_len)
        attention_masks.append(encoding['attention_mask'])   # Shape: (
            num_chunks, max_len)
        labels.extend([getattr(row, label_col)] * len(chunks))
        # Extend report ID list, same report ID for all its chunks
        report_ids.extend([report_id] * len(chunks))
    # Concatenate all batches of input tensors into single tensors
    return (
        torch.cat(input_ids),
        torch.cat(attention_masks),
        torch.tensor(labels),
        torch.tensor(report_ids))
```

A.2.1.2 Report-Level Logit Aggregation

Each chunk is passed through the BERT model independently to generate chunk-level logits. These logits are then grouped by their original report and averaged to compute a single logit vector per report. This aggregated logit is compared against the report-level ground truth using a cross-entropy loss. This allows the model to be trained with long documents, despite BERT's input size limitations, by enabling report-level supervision from chunk-level processing.

```
# Pass input chunks through the BERT model
outputs = model(input_ids, attention_mask=attention_mask)
# Extract logits from the model
logits = outputs.logits

report_logits = defaultdict(list) # Stores list of logits per report
report_targets = {} # Stores the ground truth label per report

# Group chunk-level logits and labels by their corresponding report ID
for i, rid in enumerate(report_ids):
    report_logits[rid.item()].append(logits[i])
    report_targets[rid.item()] = labels[i]

# Compute loss per report by aggregating its chunk logits
report_losses = []
for rid, logit_list in report_logits.items():
    avg_logit = torch.stack(logit_list).mean(dim=0, keepdim=True)
    label = report_targets[rid].unsqueeze(0)  # Shape: (1,)

    # Compute cross-entropy loss between averaged logits and true label
    loss = F.cross_entropy(avg_logit, label)
    report_losses.append(loss)

# Average all report-level losses to get a final scalar loss
loss = torch.stack(report_losses).mean()
```

A.2.1.3 Mixed Precision Training and Optimization

To reduce memory usage and increase speed, the training process leverages PyTorch's automatic mixed precision,

```
# Initialize the gradient scaler for AMP
scaler = torch.cuda.amp.GradScaler()
# Enable autocasting for mixed precision
with torch.cuda.amp.autocast():
    # Forward pass through the model
    outputs = model(input_ids, attention_mask=attention_mask)
    # Get logits
    logits = outputs.logits
    ... # (chunk-level to report-level aggregation and loss computation)
    loss = F.cross_entropy(avg_logit, label)
# Scale the loss to prevent underflow and perform backward pass
scaler.scale(loss).backward()
# Perform optimizer step only at specific intervals to accumulate gradients
if (step + 1) % GRADIENT_ACCUM_STEPS == 0:
    scaler.step(optimizer)
    scaler.update()
    scheduler.step()
```

A.2.1.4 Looping Over Labels

Each contributing factor is modeled independently using a one-vs-rest binary classification approach. The script loops over a predefined list of contributing factor labels, training a separate BERT model for each one,

```
LABEL_COLUMNS = [
    "Wrong Object", "Memory failure", "Fear", ...]
for label_col in LABEL_COLUMNS:
    print(f"\nTraining model for: {label_col}")
```

```
    train_df, val_df = train_test_split(df[[TEXT_COLUMN, label_col]],
        test_size=0.1)
    train_inputs, train_masks, train_labels, train_rids = chunk_reports(
        train_df, label_col, tokenizer, MAX_LEN)
    val_inputs, val_masks, val_labels, val_rids = chunk_reports(val_df,
        label_col, tokenizer, MAX_LEN)
    train_dataset = ChunkedReportDataset(train_inputs, train_masks,
        train_labels, train_rids)
    train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle
        =True)
    model = BertForSequenceClassification.from_pretrained('bert-base-
        uncased', num_labels=2).to(device)
    ...
    # Training loop for this label
    for epoch in range(EPOCHS):
        model.train()
        ...
```

This structure enables the system to,

- Train a separate binary classifier for each contributing factor.
- Use the same data pipeline and model architecture for all labels.
- Produce a dedicated model that can be evaluated and deployed independently.

Each trained model is saved under a unique folder name corresponding to the label,

```
save_path = f"Models/BERT_{label_col.replace(' ', '')}"
model.save_pretrained(save_path)
tokenizer.save_pretrained(save_path)
```

A.2.2 Inference Pipeline

The inference pipeline is responsible for analyzing new accident reports and predicting the presence of each PSF using the models trained in the previous stage. Each label is processed independently using a one-vs-rest approach.

A.2.2.1 Preprocessing and Tokenization

The input report text is chunked in the same way as during training, ensuring that the input format remains consistent,

```
def chunk_text(tokenizer, text, max_len):
    # Tokenize the input text into subword tokens
    tokens = tokenizer.tokenize(text)
    chunks = []
    # Loop through the token list in steps of `max_len`
    for i in range(0, len(tokens), max_len):
        chunk = tokens[i:i + max_len]
        chunk = tokenizer.convert_tokens_to_string(chunk)
        chunks.append(chunk)
    # Return the list of text chunks
    return chunks

# Split the long input text into chunks of up to 510 tokens
chunks = chunk_text(tokenizer, report_text, max_len=510)
# Tokenize the chunks as a batch, add padding and truncation
encoding = tokenizer(
    chunks,
    truncation=True,
    padding=True,
    return_tensors='pt',
    max_length=512)
input_ids = encoding['input_ids'].to(device)
```

```
attention_mask = encoding['attention_mask'].to(device)
```

A.2.2.2 Prediction and Aggregation

Each model is stored under a folder named after the factor it was trained to detect. These are loaded dynamically before inference.

```
from transformers import BertForSequenceClassification, BertTokenizer
# Construct the model directory path
model_path = f"Models/BERT_{label_name.replace(' ', '')}"
# Load trained BERT model
# and move it to the appropriate device (e.g., GPU or CPU)
model = BertForSequenceClassification.from_pretrained(model_path).to(device
    )
# Load the tokenizer
tokenizer = BertTokenizer.from_pretrained(model_path)
# Set the model to evaluation mode
model.eval()
```

Each chunk is passed through the model. The logits from all chunks are averaged to produce a single prediction at the report level,

```
# Disable gradient computation for inference to save memory
with torch.no_grad():
    # Forward pass through the model using the input IDs and attention mask
    outputs = model(input_ids, attention_mask=attention_mask)
    # Extract logits from the model output
    logits = outputs.logits
    # Average logits across all chunks to obtain a single prediction vector
        for the full report
    avg_logits = logits.mean(dim=0)
    # Convert averaged logits to probabilities using the softmax function
    probabilities = torch.softmax(avg_logits, dim=-1)
    # Convert to binary classification
```

```
    predicted_class = int(probabilities[1] > 0.5)
```

This averaging step mirrors the report-level logit aggregation used during training, maintaining consistency in how predictions are interpreted.

A.2.2.3 Iterating Over All Labels

The script loops through all factor models, applying the corresponding model to the same input text. The predictions are stored in a structured dictionary or output file.

```
predictions = {}
for label_name in LABEL_COLUMNS:
    model, tokenizer = load_model_and_tokenizer(label_name)
    chunks = chunk_text(tokenizer, report_text, max_len=510)
    ...
    predicted_class = run_model_on_chunks(model, tokenizer, chunks)
    predictions[label_name] = predicted_class
```

The final predictions are returned as a mapping of labels to binary outcomes.
This architecture enables reliable and interpretable predictions on real-world safety data, supporting automated data collection with potential to support analysis and decision-making in operational environments. The full code is avaialable at `https://github.com/VirtualRaphael/Human-Factors-Classifier-2.0`.

A.3 Human-Centric Summarizer

This section details the *Human-Centric Summarizer*, a hybrid extractive-abstractive pipeline for generating summaries of accident reports, discussed in Chapter 4, Section 4.3. The system is specifically engineered to elevate human roles and contextual narratives typically embedded in lengthy documents.

A.3.1 File Validation and Text Extraction

The summarizer only accepts PDF files, once this is validated the document is read page by page using `PyPDF2`.

```
if not file_path.endswith('.pdf'):
    return 'Please convert file to PDF.'
with open(file_path, 'rb') as file:
    reader = PyPDF2.PdfReader(file)
    extracted_text = [page.extract_text() for page in reader.pages]
```

A.3.2 Semantic Filtering - Section Targeting

This first pass extracts key sections (“recommendation”, “lessons learned”, etc.) by locating them using start and end phrases.

```
# Define phrases that indicate the start of the section
start_phrases = ["recommendation", "lessons learned", "advice to planning
    authorities"]
# Define phrases that indicate the end of the section
end_phrases = ["reference", "appendix", "annex", "list of", "conclusion",
    ...]  # Replace `...` with actual phrases
# Initialize a list to hold the extracted lines
```

```
extracted_part = []
# Flag to indicate whether the start of the target section has been found
found_start = False
# Iterate over each line in the extracted document text
for line in extracted_text:
    # If any start phrase is found in the current line mark the start
    if any(phrase in line.lower() for phrase in start_phrases):
        found_start = True

    # If the target section has started, keep appending lines
    if found_start:
        extracted_part.append(line)
    # If any end phrase is found in the current line, stop extraction
        if any(phrase in line.lower() for phrase in end_phrases):
            break
```

A.3.3 Semantic Filtering - Human-Focused Sentence Mining

In parallel, the system searches for individual sentences that include human roles and pronouns, which often encode decision-making and accountability.

```
# Define keywords
keywords = ["he", "she", "they", "I", "user", "operator", "manager", ...]
# Initialize a list to store sentences
extracted_sentences = []
# Iterate over each line in the extracted document text
for line in extracted_text:
    sentences = re.split(
        r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)\s',
        line)
    # Check each sentence for presence of any defined keyword
    for sentence in sentences:
        if any(keyword in sentence.lower() for keyword in keywords):
            # If a keyword is found, add the sentence to list
```

```
            extracted_sentences.append(sentence)
```

A.3.4 Preprocessing, Chunking and Tokenization

The extracted sections and sentences are concatenated and cleaned for uniformity.

```
combined_text = ' '.join(extracted_part + extracted_sentences)
```

Then, as BART has a max input length (1024 tokens), text is tokenized and split into manageable chunks with an overlap (stride) between each chunk.

```
# Tokenize the combined input text and get the token IDs
tokens = tokenizer(combined_text, return_tensors='pt')['input_ids'][0]
# Define the maximum size of each chunk
chunk_size = 1000
# Define overlap between chunks
stride = 50
# Create overlapping chunks
chunks = [
    tokens[i:i + chunk_size]
    for i in range(0, len(tokens), chunk_size - stride)]
```

A.3.5 Iterative Summarization Using BART

Each chunk is summarized independently. These summaries are then recursively re-summarized until a compressed final summary is under the token limit.

```
def summarize_text(text, max_length=200):
    inputs = tokenizer.encode(
        "summarize: " + text,
        return_tensors='pt',
```

```
        max_length=1024,
        truncation=True)
    # Generate summary token IDs using beam search
    summary_ids = model.generate(
        inputs,
        max_length=max_length, # Maximum length of the summary
        min_length=40, # Minimum length of the summary
        length_penalty=2.0, # Penalty to discourage long outputs
        num_beams=4,
        early_stopping=True)

    # Decode the summary token IDs into text
    return tokenizer.decode(summary_ids[0], skip_special_tokens=True)

# Generate the summary
summaries = [
    summarize_text(tokenizer.decode(chunk, skip_special_tokens=True))
    for chunk in chunks]
```

A.3.6 Final Output

Once the summary is within length constraints, the system returns it as final output.

```
return combined_summary
```

The *Human-Centric Summarizer* uses a multi-phase architecture that integrates,

- Top-down extractive heuristics,
- Fine-grained sentence mining for human narratives,
- BART-powered iterative abstraction and compression.

This layered approach ensures that the resulting summaries capture events, and the human role within them. The complete code for the *Human-Centric Summarizer* is available at `https://github.com/VirtualRaphael/Human-Centric-Summarizer`

A.4 Ambiguity Identifier

Here the implementation of the *Ambiguity Identifier* is broken down by each specific ambiguity type, with code extracts to illustrate key logic in each detection pipeline. The tool's developemnt and application is presented in Chapter 5, Section 5.2.

A.4.1 Undefined Abbreviations and Acronyms

This module identifies abbreviations/acronyms that appear without definition. It uses regular expressions to extract capitalized terms and compares them against long-form definitions.

A.4.1.1 Pattern Definitions

First, the abbreviation/acronym and definition formats are defined:

```
abbreviation_pattern = re.compile(r'\b[A-Z]{2,5}\b')
definition_patterns = [
    re.compile(r'\b([A-Z]{2,5})\s*\(([^)]+)\)'),
    re.compile(r'\b([^)]+)\s*\(([A-Z]{2,5})\)'),
    re.compile(r'\b([A-Z]{2,5})\s*-\s*([^.\n]+)'),
    re.compile(r'\b([^.\n]+)\s*-\s*([A-Z]{2,5})')
]
```

These patterns cover both forward and reverse definitions, including definitions linked with dashes.

A.4.1.2 Extracting Text and Tables

The PDF is processed to extract all content. Regular body text is extracted using "`PyPDF2`", and any structured table content is extracted using "`Camelot`". Table content is converted into plain text so that the same acronym detection logic can be applied consistently.

```
def extract_text_from_pdf(pdf_path):
    # Initialize an empty string to hold the full extracted text
    text = ""
    # Open the PDF file
    with open(pdf_path, 'rb') as file:
        # Create a PDF reader object
        reader = PyPDF2.PdfFileReader(file)
        # Iterate over all pages in the PDF
        for page_num in range(reader.numPages):
            # Get the page object
            page = reader.getPage(page_num)
            # Extract text from the page and append to the overall text
            text += page.extractText()
    # Return the concatenated text from all pages
    return text
def extract_tables_from_pdf(pdf_path):
    # Use Camelot to read tables from all pages of the PDF
    tables = camelot.read_pdf(pdf_path, pages='all')
    # Initialize a list to store the text representation of each table
    table_texts = []
    # Loop through the extracted tables
    for table in tables:
        df = table.df  # Get the table as a pandas DataFrame
        # Convert the DataFrame to plain text
        table_text = df.to_string(header=False, index=False)
        # Append the plain text version of the table to the list
```

```
        table_texts.append(table_text)
    # Return the list of all extracted table texts
    return table_texts
```

A.4.1.3 Abbreviation and Definition Extraction

All uppercase tokens that match the abbreviation pattern are collected. Separately, any abbreviation/acron
definition pairs are extracted using the previously defined patterns and stored in a dictionary.

```
def find_abbreviations(text):
    # Use a Regex pattern to find all abbreviations in the text
    # Return them as a set to remove duplicates
    return set(abbreviation_pattern.findall(text))

def find_defined_abbreviations(text):
    # Initialize an empty dictionary to store abbreviation-definition pairs
    definitions = {}
    # Loop through regex patterns that match definitions
    for pattern in definition_patterns:
        # Find all matches of the current pattern in the text
        matches = pattern.findall(text)
        for match in matches:
            if len(match) == 2:
                key, value = match
                key = key.strip()
                value = value.strip()
                # Check which part of the match is the abbreviation based
                    on the abbreviation pattern
                if abbreviation_pattern.fullmatch(key):
                    definitions[key] = value
                elif abbreviation_pattern.fullmatch(value):
                    definitions[value] = key
    return definitions
```

The dictionary stores known abbreviation/acronym–definition pairs in either direction depending on the format found in the text.

A.4.1.4 Identifying Undefined Abbreviations

After collecting all used and defined abbreviations from both the main text and tables, undefined abbreviations are determined by subtracting the set of defined abbreviations from the set of all used abbreviations.

```
def find_undefined_abbreviations(text, table_texts):
    # Combine the main text and all table texts into one large string
    full_text = text + " " + " ".join(table_texts)
    # Find all abbreviations mentioned anywhere in the text or tables
    all_abbrs = find_abbreviations(full_text)
    # Extract all abbreviations that have an explicit definition
    defined_abbrs = find_defined_abbreviations(full_text)
    # Identify abbreviations that are used but not defined
    undefined = all_abbrs - set(defined_abbrs.keys())
    # Return the set of undefined abbreviations
    return undefined
```

A.4.2 Unit of Measurement Ambiguity

This module identifies ambiguity introduced by using multiple unit systems (e.g., metric and imperial) for the same measurement type (e.g., length, weight) in a single SOP. Such inconsistencies can cause confusion, especially in regulated or international contexts.

A.4.2.1 Defining Unit Categories and Systems

A dictionary is constructed to define valid units for different types of physical measurements. Each category (e.g., length, weight) contains units grouped into metric, imperial, etc. systems.

```
units = {
    "length": {
        "metric": ["nm", "µm", "mm", "cm", "m", "km"],
        "imperial": ["inch", "inches", "ft", "feet", "yard", "yards", "mile
            ", "miles"]
    },
    "weight": {
        "metric": ["ng", "µg", "mg", "g", "kg", "tonne"],
        "imperial": ["oz", "lb", "stone", "ton"]
    },
    "volume": {
        "metric": ["ml", "l", "m3"],
        "imperial": ["fl oz", "cup", "pint", "quart", "gallon"]
    },
    ...
}
```

A.4.2.2 Unit Detection and Tokenization

The input text is tokenized using the "`spaCy`" NLP library. Each token is compared against all known units. The system records which units from which system are used under each measurement type.

```
def detect_units_inconsistency(doc):
    # Initialize a nested dictionary to track which units are used for each
        type
    # and which measurement system they belong to
    used_units = {
        measure: {system: set() for system in measure_data}
```

```
        for measure, measure_data in units.items()
    }
    # Iterate over each token in the document
    for token in doc:
        token_text = token.text.lower()
        # Check if the token matches any defined units
        for measure, systems in units.items():
            for system, unit_list in systems.items():
                if token_text in unit_list:
                    # Record the unit under its respective measure and
                        system
                    used_units[measure][system].add(token_text)
```

A.4.2.3 Inconsistency Detection

After scanning all tokens, the system flags any measurement type where for example, metric and imperial units are used. These cases are considered ambiguous.

```
# Initialize a list to collect inconsistencies
inconsistencies = []
# Iterate over each type of measurement and its associated systems
for measure, systems in used_units.items():
    # Check which unit systems have at least one unit used
    non_empty_systems = [s for s, vals in systems.items() if vals]
    # If more than one system is used for the same measure, it's an
        inconsistency
    if len(non_empty_systems) > 1:
        # Record the inconsistent measure and the systems involved
        inconsistencies.append((measure, systems))
# Return the list of all detected inconsistencies
return inconsistencies
```

The result is a list of ambiguous measurement types and the units used from each system.

A.4.3 Temporal Ambiguity

The temporal ambiguity module targets both vague temporal expressions and syntactic constructions involving temporal conjunctions that may result in unclear sequencing or timing of events.

A.4.3.1 Temporal Lexicon Definition

Two curated lists are used to detect potentially ambiguous time references:

- Vague temporal words, words that indicate imprecise frequency or duration.
- Temporal conjunctions, words that connect clauses in time but can introduce ambiguity

```
vague_temporal_words = [
    "frequently", "periodically", "every so often", "sometimes", "
       occasionally",
    "often", "regularly", "from time to time", "at times", "once in a while
       ",
    "rarely", "seldom", "usually", "generally", "normally", "typically", "
       extended time"]
temporal_conjunctions = [
    "before", "after", "during", "simultaneously", "while", "when", "until"
       ,
    "once", "since", "as soon as", "whenever"]
```

A.4.3.2 Detection of Vague Temporal Words

The first detection function flags sentences that contain any of the vague terms above.

```
def detect_vague_temporal_words(doc):
```

```
    # Iterate over each token
    for token in doc:
        # Check if any tokens are in predefined list
        if token.text.lower() in vague_temporal_words:
            # If a vague temporal word is found
            return True
            # else
    return False
```

This method ensures that any sentence using imprecise frequency terms (for example “occasionally monitor the pressure”) is flagged as temporally ambiguous.

A.4.3.3 Detection of Conjunction Based Ambiguity

Temporal conjunctions can introduce ambiguity, especially in complex sentences. This function uses syntactic parsing (via “spaCy”) to check for dependency structures that may indicate unclear temporal relationships.

```
def detect_temporal_conjunction_ambiguities(doc):
    # Iterate over each token in the document
    for token in doc:
        # Check if the token is a temporal conjunction
        if token.text.lower() in temporal_conjunctions:
            # Get the syntactic head of the conjunction
            head = token.head
            # Ensure the head is a verb or auxiliary verb
            if head and head.pos_ in ["VERB", "AUX"]:
                # Inspect the dependents of the head verb
                for child in head.children:
                    # Look for markers of subordinate or prepositional
                    if child.dep_ in ["advcl", "prep", "mark"] and child !=
                        token:
                        # If such a structure is found,
```

```
                        return True
    # If not
    return False
```

This first checks if a temporal conjunction effects a verb, and then looks for other dependent phrases that could compete in scope. If multiple adverbial or clause-like elements exist, the temporal relation may be unclear.

A.4.3.4 Combined Ambiguity Check

The full check processes the input text using "spaCy", and applies both checks to each sentence. Any sentence that triggers one or both heuristics is marked as ambiguous.

```
def check_temporal_ambiguity(text):
    # Load the SpaCy English language model
    nlp = spacy.load("en_core_web_sm")
    doc = nlp(text)

    # Initialize ambiguous list
    ambiguous_sentences = []

    # Iterate over each sentence in the document
    for sent in doc.sents:
        # Check for defined terms
        has_vague_temporal_word = detect_vague_temporal_words(sent)

        # Check for use of temporal conjunctions
        has_temporal_conjunction_ambiguity =
            detect_temporal_conjunction_ambiguities(sent)
        if has_vague_temporal_word or has_temporal_conjunction_ambiguity:
            ambiguous_sentences.append(sent.text)

    # Return all ambiguous sentences
```

```
    return "\n".join(ambiguous_sentences)
```

A.4.4 Syntactic Ambiguity

The syntactic ambiguity module focuses on two common sources, complex subordinate clause nesting and unclear prepositional phrase attachment.

A.4.4.1 Detection of Subordinate Clauses

The first function scans for the presence of multiple subordinate constructions in a sentence. While one or two are typically manageable, three or more can lead to significant ambiguity.

```
def detect_subordinate_clauses(doc):
    # Initialize a counter for subordinate clauses
    subordinate_clauses = 0
    # Iterate over each token in the parsed document
    for token in doc:
        # Check if the token is part of a subordinate clause based on its
            dependency label
        if token.dep_ in ["mark", "advcl", "acl"]:
            # Increment the counter
            subordinate_clauses += 1
    return subordinate_clauses
```

The function identifies subordinate clauses based on three specific dependency labels, `mark`, `advcl`, and `acl`. The label `mark` corresponds to markers that introduce subordinate clauses, such as "although" or "because". The `advcl` label denotes adverbial clause modifiers that typically provide context like time, reason, or condition. The `acl` label refers to clausal modifiers of nouns. If more than two of these structures are found in a single sentence, the sentence is flagged as potentially syntactically ambiguous.

A.4.4.2 Detection of Prepositional Phrase Attachment Ambiguity

This check looks for possible ambiguity in the attachment of prepositional phrases, where it may be unclear what the phrase modifies.

```
def detect_prepositional_phrase_attachment_ambiguities(doc):
    # Initialize a list for storage
    potential_ambiguities = []
    # Iterate over each token in the parsed document
    for token in doc:
        # Check if the token is a preposition (ADP = Adposition)
        if token.pos_ == "ADP":
            # Get the head of the preposition
            prev_token_1 = token.head
            # Get the head of the head
            prev_token_2 = prev_token_1.head if prev_token_1.dep_ != "ROOT"
                else None
            # If the preposition is not directly attached to the root and
               there's a higher-level head, it may be ambiguously attached
            if prev_token_2 and prev_token_1.dep_ != "ROOT":
                # Add this preposition to the list of potential ambiguities
                potential_ambiguities.append(token)
    return potential_ambiguities
```

This function analyzes each token in the parsed document and checks whether its part-of-speech tag is `ADP`, which corresponds to adpositions, typically prepositions. For each such token, it retrieves its syntactic head using `token.head`, referred to as `prev_token_1`. It then attempts to retrieve the head of that head, stored in `prev_token_2`, provided that `prev_token_1` is not the root of the sentence. The conditional check ensures that only nested dependency structures are considered. If both `prev_token_1` and `prev_token_2` exist and the preposition is not directly attached to the root, the function assumes a potentially ambiguous attachment and adds the preposition token to the `potential_ambiguities` list. This logic is designed to capture situations where a prepositional phrase is embedded within a deeper dependency structure, which can obscure the intended referent of the phrase.

A.4.4.3 Combined Ambiguity Detection

The main function applies both checks and flags any sentence meeting either criterion.

```
def check_syntactic_ambiguity(text):
    # Load the English SpaCy model
    nlp = spacy.load("en_core_web_sm")
    # Process the input text
    doc = nlp(text)
    # Initialize a list to collect sentences
    ambiguous_sentences = []
    # Iterate over each sentence in the document
    for sent in doc.sents:
        # Count the number of subordinate clauses in the sentence
        subordinate_clauses = detect_subordinate_clauses(sent)
        # Detect possible prepositional phrase attachment ambiguities
        prepositional_phrase_ambiguities =
            detect_prepositional_phrase_attachment_ambiguities(sent)

        # If the sentence has more than 2 subordinate clauses
        # or contains ambiguous prepositional phrase attachments
        if subordinate_clauses > 2 or prepositional_phrase_ambiguities:
            # Add it to the list
```

```
            ambiguous_sentences.append(sent.text)
    return "\n".join(ambiguous_sentences)
```

The threshold of 2 subordinate clauses can be adjusted depending on SOP style.

A.4.5 Scope Ambiguity

This module identifies scope ambiguity that arises when a verb lacks a clearly defined direct object. In SOPs, missing objects can obscure which component, action, or material the instruction refers to, leading to misinterpretation.

A.4.5.1 Verb Object Detection

The function checks each verb in the sentence to ensure it has an explicitly linked direct object. A verb without a `dobj` (direct object) dependency is considered ambiguous if used in an imperative or instructive context.

```
def detect_scope_ambiguity(doc):
    for token in doc:
    # Check if the token is a verb
    if token.pos_ == "VERB":
        # Check verb's dependency tree for direct object
        has_direct_object = any(child.dep_ == "dobj" for child in token.
            children)
        # If the verb has no direct object
        if not has_direct_object:
            return True
return False
```

This function iterates through each token in the parsed document. If a token has PoS tag `VERB`, it examines that verb's children in the dependency tree. Using a generator expression, it checks whether any child has the dependency label `dobj`, which "`spaCy`" assigns to direct objects. If no such object is found for a verb, the sentence is considered potentially ambiguous and the function returns `True`. Otherwise, it continues checking remaining tokens. The logic assumes that omitting an object for an action verb may leave the scope of the instruction underspecified.

A.4.5.2 Full Sentence-Level Evaluation

The following function wraps the detector above to process an entire text. Each sentence is checked independently, and ambiguous ones are collected.

```
def check_scope_ambiguity(text):
    # Load SpaCy model
    nlp = spacy.load("en_core_web_sm")
    # Process the input text
    doc = nlp(text)
    # List to store sentences
    ambiguous_sentences = []
    # Iterate through each sentence in the document
    for sent in doc.sents:
        # Use detect_scope_ambiguity function
        has_scope_ambiguity = detect_scope_ambiguity(sent)
        # Append the sentence's text to the results
        if has_scope_ambiguity:
            ambiguous_sentences.append(sent.text)
    return "\n".join(ambiguous_sentences)
```

This module is effective in identifying underspecified actions where the target of the action is missing. While some imperative sentences may omit the object intentionally, these can be domain-specific and could benefit from refinement.

A.4.6 Quantity Ambiguity

This module detects quantity related ambiguity caused simply by vague or imprecise quantifiers.

A.4.6.1 Ambiguous Quantity Lexicon

A predefined list of ambiguous quantity terms is used to identify potentially problematic expressions. These include both modifiers (“a little”, “a lot”) and approximators (“approximately”, “more or less”).

```
ambiguous_quantity_words = [
    "some", "many", "enough", "several", "few",...]
```

A.4.6.2 Token-Based Detection of Quantity Terms

The detection function scans each token in a sentence and returns `True` if any token matches an entry in the list above.

```
def detect_ambiguous_quantity_words(doc):
    # Iterate over each token
    for token in doc:
        # Check if token in predefined list
        if token.text.lower() in ambiguous_quantity_words:
            # If a match is found
            return True
    return False
```

This function iterates through the "spaCy"-parsed doc and checks each token. If the token exactly matches any item in the ambiguous_quantity_words list, the function concludes that the sentence contains a vague quantity and returns True. Otherwise, it proceeds through all tokens without raising a flag.

The main function processes the input text by sentence and applies the detection logic to each. Sentences that contain ambiguous quantity expressions are collected and returned.

```
def check_quantity_ambiguity(text):
    # Load the SpaCy model
    nlp = spacy.load("en_core_web_sm")
    # Process the input text
    doc = nlp(text)
    # Initialize a list
    ambiguous_sentences = []
    # Iterate through each sentence in the document
    for sent in doc.sents:
        # Check sentence for defined list
        has_ambiguous_quantity_word = detect_ambiguous_quantity_words(sent)
        # Add to list
        if has_ambiguous_quantity_word:
            ambiguous_sentences.append(sent.text)
    # Join all sentences to one list
    return "\n".join(ambiguous_sentences)
```

This module uses lexical matching, which is fast and effective for known vague expressions.

A.4.7 Lexical Ambiguity

This module detects lexical ambiguity, which occurs when a word has multiple possible meanings and the context does not clearly indicate which sense is intended. This is particularly important in SOPs, where domain-specific terminology may overlap with general vocabulary.

A.4.7.1 POS Tagging and Synset Retrieval

The input text is first processed using "spaCy" to assign POS tags. These tags are mapped to WordNet compatible formats to retrieve sense definitions (synsets) for each word.

```
def get_pos_tags(text):
    # Process the input text using SpaCy
    doc = nlp(text)
    # Return a list of tuples
    return [(token.text, token.tag_) for token in doc]
def get_wordnet_pos(spacy_tag):
    # Map the SpaCy POS tag to WordNet's categories
    if spacy_tag.startswith('J'):
        return wn.ADJ  # Adjective
    elif spacy_tag.startswith('V'):
        return wn.VERB  # Verb
    elif spacy_tag.startswith('N'):
        return wn.NOUN  # Noun
    elif spacy_tag.startswith('R'):
        return wn.ADV  # Adverb
    else:
        return None  # Not supported by WordNet

def extract_top_synsets(word, pos_tag, top_n=3):
    # Convert the SpaCy POS tag to a WordNet POS
    wn_pos = get_wordnet_pos(pos_tag)
    # Lookup synsets for the word with the specific POS
    if wn_pos:
        synsets = wn.synsets(word, pos=wn_pos)
    else:
        synsets = wn.synsets(word)  # Fall back to all POS if mapping is
            unavailable
    # Return the top synsets
    return synsets[:top_n]
```

This step ensures that only relevant synsets are considered by aligning the "`spaCy`" PoS tag with WordNet's expected POS categories. For example, the word "monitor" can serve as both a noun ("a computer screen" or "a person who observes") and a verb ("to observe or track"). By first identifying the word's POS tag using "`spaCy`", and then mapping it to the corresponding WordNet tag, the module filters the synsets so that only definitions appropriate to that grammatical role are retrieved. If "monitor" is tagged as a noun, verb-based meanings are excluded, reducing noise in later similarity comparisons.

A.4.7.2 Sentence and Definition Embedding

Each target word and each of its corresponding WordNet definition texts are embedded using a BERT model. The BERT model produces high-dimensional vector representations that capture semantic context. For the word in question, its contextual embedding is extracted from the full sentence using token-level representations. Separately, each definition from WordNet is encoded as an independent sentence and averaged across tokens to produce a single embedding. Cosine similarity is then computed between the word's in context embedding and each definition embedding. The highest similarity score is interpreted as the best contextual fit. If the highest similarity is still below a defined threshold, the word is considered lexically ambiguous.

```
def encode_sentence(sentence):
    # Tokenize the input sentence and return PyTorch tensors
    inputs = tokenizer(sentence, return_tensors='pt')
    # Pass the inputs through the model
    outputs = model(**inputs)
    # Return the token embeddings for each word
    return outputs.last_hidden_state

def get_sentence_embedding(sentence):
    # Get token embeddings for the sentence
    embeddings = encode_sentence(sentence)
    # Average the embeddings across all tokens to get a single vector
    return embeddings.mean(dim=1).squeeze().detach().numpy()
```

```
def get_definition_embedding(definition):
    # Reuse sentence embedding function for definition string
    return get_sentence_embedding(definition)

def calculate_similarity(embedding1, embedding2):
    # Use cosine similarity to measure how close two sentence vectors are
    return cosine_similarity([embedding1], [embedding2])[0][0]
```

A.4.7.3 Lexical Ambiguity Detection

A sentence is considered lexically ambiguous if a polysemous word appears in it and none of its top senses closely match the usage context.

```
def check_lexical_ambiguity(text, threshold=0.5):
    # Get PoS tags for all words in the input text
    pos_tags = get_pos_tags(text)
    # Initialize a list
    ambiguous_sentences = []
    # Iterate over each word and its POS tag in the sentence
    for i, (word, pos_tag) in enumerate(pos_tags):
        # Retrieve the top synsets from WordNet
        synsets = extract_top_synsets(word, pos_tag)
        # If the word has more than one possible sense, continue
        if len(synsets) > 1:
            sentence = ' '.join([w for w, _ in pos_tags])
            # Get the embedding for the word in its sentence context
            word_embedding = encode_sentence(sentence)[0][i].detach().numpy
                ()
            # Initialize maximum similarity score
            max_similarity = 0
            # Compare the word's context embedding to each synset
                definition
            for synset in synsets:
                # Get the definition text of the synset
```

```
                definition = synset.definition()
                # Encode the definition into an embedding
                definition_embedding = get_definition_embedding(definition)
                # Compute cosine similarity between word and definition
                similarity = calculate_similarity(word_embedding,
                    definition_embedding)
                # Keep the highest similarity found
                if similarity > max_similarity:
                    max_similarity = similarity
            # If none of the meanings match the word's usage closely, store
            if max_similarity < threshold:
                ambiguous_sentences.append(sentence)
    # Return all flagged sentences
    return "\n".join(ambiguous_sentences)
```

This method combines symbolic (WordNet) and distributional (BERT) semantics to detect ambiguity. The threshold (0.5) can be tuned to balance sensitivity and specificity.

A.4.8 Conditional Ambiguity

This module detects conditional ambiguity in sentences where multiple conditions or disjunctive conjunctions (“or”, “either”) may obscure the logic or intended outcome of a procedural instruction.

A.4.8.1 Definition of Conjunction Lists

Two sets of conjunctions are used to identify conditional and disjunctive logic structures,

```
conditional_conjunctions = ["if", "when", "unless",...]
disjunctive_conjunctions = ["or", "either",...]
```

A.4.8.2 Detection of Conditional and Disjunctive Elements

The module includes separate functions for detecting conditionals and disjunctions in a sentence,

```
def detect_conditional_conjunctions(doc):
    # Iterate over each token
    for token in doc:
        # Check if the token in list
        if token.text.lower() in conditional_conjunctions:
            # If found
            return True
    # Else
    return False

def detect_disjunctive_conjunctions(doc):
    # Iterate over each token
    for token in doc:
        # Check if the token is in list
        if token.text.lower() in disjunctive_conjunctions:
            # If found
            return True
    # Else
    return False
```

Each function iterates through the tokens in a sentence and checks whether the lowercase form of any token appears in the corresponding conjunction list. If a match is found, the function returns `True`, indicating that the sentence includes a conditional or disjunctive structure.

A.4.8.3 Detection of Nested Conditions

Nested or repeated conditions can cause ambiguity when the logical relationship between them is not clear. The function below counts the number of conditionals in a sentence and flags it if more than one is present.

```
def detect_nested_conditions(doc):
    # Initialize a counter
    condition_count = 0
    # Iterate over each token
    for token in doc:
        # Check if the token is a conditional conjunction
        if token.text.lower() in conditional_conjunctions:
            # Increment the counter if a condition is found
            condition_count += 1
        # If more than one condition is detected, flag
        if condition_count > 1:
            return True
    # Else
    return False
```

The function maintains a counter for conditional tokens. If more than one such token is found, the sentence is flagged as potentially ambiguous due to multiple overlapping conditions.

A.4.8.4 Overall Conditional Ambiguity Check

This wrapper function combines all three detection routines. A sentence is flagged if it contains a conditional and either a disjunction or a nested conditional structure.

```
def check_conditional_ambiguity(text):
    # Load SpaCy model
    nlp = spacy.load("en_core_web_sm")
    # Process the input text
```

```
    doc = nlp(text)
    # Initialize a list
    ambiguous_sentences = []
    # Iterate over each sentence
    for sent in doc.sents:
        # Check if the sentence contains a conditional conjunction
        has_conditional_conjunction = detect_conditional_conjunctions(sent)
        # Check if the sentence contains a disjunctive conjunction
        has_disjunctive_conjunction = detect_disjunctive_conjunctions(sent)

        # Check if the sentence contains nested or multiple conditions
        has_nested_conditions = detect_nested_conditions(sent)

        # Flag the sentence as ambiguous if it contains both,
        # A condition, and
        # Either a disjunction or multiple nested conditions
        if has_conditional_conjunction and (has_disjunctive_conjunction or
            has_nested_conditions):
             ambiguous_sentences.append(sent.text)
    # Output
    return "\n".join(ambiguous_sentences)
```

This module targets logical ambiguity caused by overlapping or compound conditions. It helps identify instructions where the triggering criteria or logical pathways may be misinterpreted.

A.4.9 Combined Ambiguity Detection Pipeline

This final script integrates all individual ambiguity detection modules into a single pipeline. It processes SOP documents in PDF format, applies the selected ambiguity checks, and writes the results to output files for review and traceability.

A.4.9.1 Import and Setup

All individual ambiguity checkers are imported from their respective modules. The system also uses `PyMuPDF` (imported as `fitz`) to extract text from the PDF.

```
from ambiguity_conditional import check_conditional_ambiguity
from ambiguity_lexical import check_lexical_ambiguity
from ambiguity_quantity import check_quantity_ambiguity
from ambiguity_scope import check_scope_ambiguity
from ambiguity_syntactic import check_syntactic_ambiguity
from ambiguity_temporal import check_temporal_ambiguity
from ambiguity_uomeasurement import check_uomeasurement_ambiguity
from undefined_abbreviations_acronyms import
    check_undefined_abbreviations_acronyms
```

A.4.9.2 Text Extraction from PDF

The text is extracted from all pages of the input PDF using `fitz`. This raw text is passed to each ambiguity detection function.

```
def extract_text_from_pdf(pdf_path):
    # Open the PDF document using PyMuPDF
    pdf_document = fitz.open(pdf_path)
    # Initialize an empty string
    text = ""
    # Iterate over all pages in the PDF
    for page_num in range(pdf_document.page_count):
        # Access the current page
        page = pdf_document[page_num]
        # Extract text from the current page
        text += page.get_text()
    # Return the full extracted text
    return text
```

A.4.9.3 Main Processing Logic

The main function takes in the path to a PDF, a list of ambiguity types to check, and an output directory. It calls the relevant ambiguity detection functions conditionally, depending on the user-specified configuration.

```
def main(pdf_path, output_dir, ambiguity_checks):
    # Extract text from the input PDF
    text = extract_text_from_pdf(pdf_path)

    # Check which ambiguity checks to run
    if "ambiguous_sentences" in ambiguity_checks:
        ambiguous_sentences = ""
        if "conditional" in ambiguity_checks:
            ambiguous_sentences += check_conditional_ambiguity(text)
        if "lexical" in ambiguity_checks:
            ambiguous_sentences += check_lexical_ambiguity(text)
        if "quantity" in ambiguity_checks:
            ambiguous_sentences += check_quantity_ambiguity(text)
        if "scope" in ambiguity_checks:
            ambiguous_sentences += check_scope_ambiguity(text)
        if "syntactic" in ambiguity_checks:
            ambiguous_sentences += check_syntactic_ambiguity(text)
        if "temporal" in ambiguity_checks:
            ambiguous_sentences += check_temporal_ambiguity(text)
        if "uomeasurement" in ambiguity_checks:
            inconsistencies, sentences_with_issues =
                check_uomeasurement_ambiguity(text)

            # If ambiguities are found
            if sentences_with_issues:
                write_output_to_file(
                    "\n".join(sentences_with_issues),
                    os.path.join(output_dir, "
                        Potentially_Ambiguous_Sentences.txt"))
```

A.4.9.4 Handling Abbreviations and Unit Consistency Separately

The check for undefined abbreviations/acronyms and unit of measurements inconsistencies are handled in distinct blocks and written to individual output files.

```
    # If abbreviations requested
if "undefined_abbreviations" in ambiguity_checks:
    undefined_abbreviations = check_undefined_abbreviations_acronyms(
        pdf_path)
    # Write the list of undefined abbreviations to an output file
    write_output_to_file(
        undefined_abbreviations,
        os.path.join(output_dir, "Undefined_Abbreviations_and_Acronyms.txt"
            ))

# If UoM requested
if "inconsistent_units" in ambiguity_checks:
    inconsistent_units = check_uomeasurement_ambiguity(text)
    # Write inconsistencies to an output file
    write_output_to_file(
        inconsistent_units,
        os.path.join(output_dir, "Inconsistent_Use_of_Units.txt"))
```

A.4.9.5 Execution Entry Point

Finally, a configuration block allows it to be run as a standalone program. The ambiguity types to check are defined as a list and passed to the main function.

```
if __name__ == "__main__":
    pdf_path = "SOP_File.pdf"
    output_dir = "output folder"
    ambiguity_checks = [
```

```
        "conditional", "lexical", "quantity", "scope",
        "syntactic", "temporal", "uomeasurement",
        "undefined_abbreviations", "inconsistent_units"
    ]
    main(pdf_path, output_dir, ambiguity_checks)
```

This methodological pipeline enables modular execution of ambiguity checks and generates separate logs for each type of ambiguity. Each component can be toggled on or off depending on the desired focus or type of SOP under review.

A.5 High-Potential Violations Trigger Identification tool

This section describes the development and application of the *Violation Trigger tool*, designed to identify SOP sections that contain directives with high risk potential if regulatory or procedural violations are committed, presented in Chapter 5, Section 5.3. Similar to *HF Classifer 2.0* the system uses a fine-tuned BERT model trained on past incident data.

A.5.1 Training Pipeline

The model is trained using the `bert-base-uncased` architecture from Hugging Face's "`Transformers`" library. The input data is a labeled Excel file containing incident description text and binary violation labels.

A.5.1.1 Data Preparation and Tokenization

Each accident description is processed using a BERT tokenizer.

```
# Load the training data
df = pd.read_excel('Violation_Training_Data.xlsx')
# Split the data into training and validation sets
train, val = train_test_split(df, test_size=0.2)
# Load the BERT tokenizer
# Tokenize the training data
train_encodings = tokenizer(
    train['Combined Text Data'].tolist(),
    truncation=True,
    padding=True,
    return_tensors='pt',
    max_length=512)

# Tokenize the validation data in the same way
val_encodings = tokenizer(
    val['Combined Text Data'].tolist(),
    truncation=True,
    padding=True,
    return_tensors='pt',
    max_length=512)
```

A.5.1.2 Model Training

The classifier is implemented using `BertForSequenceClassification` with two output labels. It uses mixed precision training for efficiency, and optimizes using the AdamW optimizer with a linear learning rate scheduler.

```
# Load BERT model for sequence classification
model = BertForSequenceClassification.from_pretrained(
    'bert-base-uncased',
```

```
    num_labels=2)
# Set up the AdamW optimizer
optimizer = AdamW(model.parameters(), lr=2e-5)
# Set up a linear learning rate scheduler
scheduler = get_linear_schedule_with_warmup(
    optimizer,
    num_warmup_steps=0,
    num_training_steps=len(train_dataloader) * num_epochs)
# Initialize mixed precision gradient scaler
scaler = torch.cuda.amp.GradScaler()
for epoch in range(num_epochs):
    model.train()  # Set model to training mode
    for batch in train_dataloader:
        # Automatic mixed precision
        with torch.cuda.amp.autocast():
            # Forward pass
            outputs = model(
                input_ids,
                attention_mask=attention_mask,
                labels=labels  # Computes loss)
            # Scale loss
            loss = outputs.loss / gradient_accumulation_steps
        # Scale loss and backpropagate gradients
        scaler.scale(loss).backward()
        # ... (gradient accumulation, optimizer step, scaler update,
           scheduler step)
```

Evaluation is performed using accuracy, precision, recall, and F1-score after each epoch.

A.5.1.3 Validation and Model Saving

The model is evaluated on the validation set using standard metrics, then saved along with the tokenizer for later inference.

```
val_acc = accuracy_score(val_labels, val_preds)
torch.save(model.state_dict(), 'bert_model.pt')
tokenizer.save_pretrained('bert_tokenizer')
```

A.5.2 Violation Detection on SOP PDFs

The trained model is applied to full SOP documents in PDF format. The system segments SOPs into finer-grained textual sections for individual classification, to improve granularity and interpretability.

A.5.2.1 Text Extraction and Preprocessing

Text is extracted from each page using `PyMuPDF`, then cleaned by removing repeated content and formatting.

```
def extract_text_from_pdf(pdf_path):
    # Open the PDF file with PyMuPDF
    document = fitz.open(pdf_path)
    # Extract plain text from each page
    return [page.get_text("text") for page in document]
def remove_formatting(text):
    # Remove common footer/header patterns
    text = re.sub(r'Page \d+ of \d+', '', text)
    ... # Additional formatting cleanup
    return text
```

To focus analysis on procedural content, only pages between “Introduction” and “Appendix” are analyzed. Each page is split into smaller sections based on double newlines or heading-like formatting cues.

```
def split_into_sections(text):
    # Split the text using either
    # Two or more consecutive newlines, or
    # A single newline before a line that looks like a heading or starts
        with a label
    sections = re.split(r'\n{2,}|\n(?=\w.+?:)', text)
    # Remove empty entries and space
    return [s.strip() for s in sections if s.strip()]
```

A.5.2.2 Section Classification and Output

Each section is independently tokenized and classified using the fine-tuned BERT model. Sections classified as violations are saved along with their corresponding page number.

```
for section in sections:
    if section.strip():
        # Use a classification model to predict the label
        prediction = classify_text(section, model, tokenizer, device)
        # If the model predicts class 1
        if prediction == 1:
            # Add the section and pg. number to the results
            classified_sections.append(f"Page {page_num}:\n{section}")
```

The system writes all flagged sections to an output file for further review.

```
# Save text to output file
with open('High Potential Violation Directives.txt', 'w') as file:
    for section_text in classified_sections:
        file.write(section_text + "\n\n")
```

The final output can assist compliance officers and reviewers by automatically surfacing potentially risky instructions that may require further consideration, emphasis or extra training. The full code is available at `https://github.com/VirtualRaphael/Violation-Trigger-Tool`.

A.6 Human Factors Causal Relationship Tool

The *HF Relationships tool*, presented in Chapter 6, Section 6.4, constructs the DAG structures for BN based on human reliability data, modeling potential causal dependencies between PSFs.

A.6.1 Inputs and Configuration

The users interactively select,

- Factor Mode, either 53 original CREAM factors or Grouped categories.
- Inference Method,
  - `K2`, score-based greedy structure learning.
  - `NPC`, constraint-based learning using conditional independence.
  - `Expert`, causal structures based on past publications, defined in external Excel sheets.
  - `Aggregated`, includes edges appearing in at least two of the above.
- Other Parameters,
  - Maximum parent count for K2.
  - Significance level for conditional independence tests in NPC.

A.6.2 Main Interface

The entry point is the function `run_causal_inference`, which provides a graphical dialog for users to select model configuration options:

```
% Create a dialog window
d = dialog('Position',[300 300 400 300], 'Name', 'Causal Inference Config')
    ;
...
% Create a dropdown list inside the dialog
methodMenu = uicontrol('Parent', d, ...
                       'Style', 'popupmenu', ...
                       'String', {'k2', 'npc', 'expert', 'aggregated'}, ...
                       ...);
```

After the user confirms selections, the appropriate dataset and expert matrix are loaded:

```
if useGrouped
    % If using grouped data, load the grouped data matrix
    dataMatrix = readmatrix('MATA_D_Matrix_Grouped.xlsx');
    % Load the expert link matrix corresponding to grouped data
    expertMatrix = readmatrix('Grouped_ExpertOp.xlsx');
else
    % Else, load the full data matrix
    dataMatrix = readmatrix('MATA_D_MatFormFull.xlsx');
    % Load the expert link matrix for the full data
    expertMatrix = readmatrix('Expert Links Format.xlsx');
end
```

Based on the selected method (`k2`, `npc`, `expert`, or `aggregated`), different learning functions are invoked to generate the DAG.

A.6.3 Entropy Based Ordering for K2

To guide the K2 structure search, a variable ordering is computed using MI and CRAE. This follows an approach adapted from (Benmohamed et al. 2022)

```
% Calculate mutual information between variables i and j
MI(i,j) = mutual_info(Data(:, i), Data(:, j));
MI(i,j) = mutual_info(Data(:,i), Data(:,j));
...
% Compute conditional entropy ratios in both directions
CR_yx = Hx_y / (Hx * states_x);
CR_xy = Hy_x / (Hy * states_y);
% Determine direction of causality based on the smaller normalized
    conditional entropy ratio
if CR_yx < CR_xy
    % Variable j causes variable i
    Dir(j, i) = 1;
else
    % Variable i causes variable j
    Dir(i, j) = 1;
end
```

A topological sort on this directed matrix provides a causally informed order used in K2.

A.6.4 K2 Structure Learning

The K2 algorithm operates under the assumption that a valid ordering of the nodes is provided in advance, this prevents cycles and ensures acyclicity in the resulting graph. In this implementation, the node ordering is derived automatically using the entropy-based approach described in Section A.6.3.

The main objective of K2 is to identify the optimal parent set for each node in the DAG such that the overall network score, which reflects the likelihood of the data given the structure, is

maximized. For each node X_i, the algorithm considers only the nodes that appear earlier in the specified order as potential parents.

The structure learning process begins by initializing the node with no parents. Then, parents are added one at a time from the pool of allowable candidates if they increase the score provided by the scoring function `GClosedFun`. This process continues until either,

- The maximum number of parents is reached, or
- No additional parent yields an improved score.

The core logic of the algorithm is as follows,

```
for p = 1:d
    i = Order(p); % Get index of the current node from the order
    Parent = zeros(d,1); % Initialize empty parent set
    Pold = GClosedFun(LGObj, i, find(Parent)); % Initial score with no
       parents
    LocalMax = Pold;
    OKToProceed = true;

    while OKToProceed && sum(Parent) < MaxParents
        BestCandidate = -1;
        for q = 1:p-1
            j = Order(q);
            if Parent(j) == 0
                Parent(j) = 1; % Temporarily add j as a parent
                NewScore = GClosedFun(LGObj, i, find(Parent)); % Score with
                    j
                Parent(j) = 0; % Remove candidate

                if NewScore > LocalMax
                    LocalMax = NewScore;
                    BestCandidate = j; % Store best scoring
                end
            end
        end

        if BestCandidate ~= -1
```

```
            Parent(BestCandidate) = 1; % Permanently add best parent
            DAG(BestCandidate, i) = 1; % Record edge: parent → child
            Pold = LocalMax;
        else
            OKToProceed = false; % No further improvements
        end
    end

    Scores(i) = Pold; % Save final score for node i
end
```

This process is repeated for all nodes in the order, gradually constructing a complete DAG. The `GClosedFun` function, explained in the next section, serves as the evaluation metric for comparing different parent combinations.

A.6.4.1 Scoring Function

The scoring function `GClosedFun` evaluates how well a node is explained by its parent configuration using a Bayesian Dirichlet equivalent uniform score. This score is computed using frequency tables derived from the training data and considers both the number of parent configurations and the number of states the child variable can take.

First, the relevant components are extracted,

```
Frequency = LGObj.FreqTable{i};  % Conditional frequency table for node i
ri = LGObj.r(i); % Number of discrete states of node i
qi = size(Frequency, 1) / ri; % Number of unique parent configurations
GFunValue = 0; % Initialize score accumulator
```

Then, for each parent configuration j, the function computes the log-likelihood of observing each possible child state k summed over all states and configurations,

```
for j = 1:qi
```

```
    % Initialize sum of frequencies for the current group j
    Sum = 0;
    for k = 1:ri
        % Accumulate frequency counts
        Sum = Sum + Frequency((j - 1) * ri + k);
        % Increment GFunValue
        GFunValue = GFunValue + gammaln(Frequency((j - 1) * ri + k) + 1);
    end
    % Update GFunValue
    GFunValue = GFunValue + gammaln(ri) - gammaln(Sum + ri);
end
```

Here,

- `Frequency((j-1)*ri + k)` accesses the count of observing value k of X_i under parent configuration j.
- The use of `gammaln` ($\log(\Gamma(x))$) improves numerical stability and avoids underflow issues common in high-dimensional models.

This function returns a score that increases with better model fit (when the child node's behavior is well-explained by its parents). The K2 algorithm uses this score to guide its greedy parent selection process.

Together, `k2_structure_learning.m` and `GClosedFun.m` form an efficient implementation of the K2 algorithm.

A.6.5 NPC Structure Learning

The NPC method is a constraint-based algorithm for learning the structure of a BN. NPC relies on statistical independence tests to determine whether an edge should exist between two nodes. This method proceeds in two stages,

1. Identifying undirected dependencies using pairwise mutual information.
2. Optionally orienting edges based on heuristic assumptions or further constraints.

The algorithm starts by assuming a fully connected undirected graph and then systematically removes edges between nodes that are found to be (conditionally) independent. Independence is determined via mutual information and tested for statistical significance using a chi-squared test.

The core of the algorithm is implemented as follows,

```
% Initialize a fully connected adjacency matrix (size d x d)
% Iterate over all unique pairs of variables (i, j)
for i = 1:d
    for j = i+1:d
        % Compute conditional MI between variables i and j
        [MI, R, M] = ConditionallyIndependent_MutualInformation(LGObj, i, j
            );
        % Perform a chi-squared test
        if CITest_ChiTwoVar(MI, R, M, alpha)
            % If i and j are conditionally independent, remove the edge by
                setting adjacency entries to 0
            Undirected(i,j) = 0;
            Undirected(j,i) = 0;
        end
    end
end
```

Here,

- `MI` is the mutual information between variables X_i and X_j.
- `R` is the degrees of freedom used in the chi-squared test.
- `M` is the number of data samples.
- The edge is removed if the variables are deemed conditionally independent at the significance level α.

After pruning edges, the adjacency matrix is optionally converted into a directed graph. In the current implementation, directions are added heuristically (by assigning direction from lower to higher index) to maintain compatibility with further steps,

```
for i = 1:d
    % Find indices of nodes connected to node i in the undirected graph
    neighbors = find(Undirected(i, :) == 1);
    % Iterate over each neighbor j of node i
    for j = neighbors
        % Only assign direction for edges where i < j
        if i < j
            % Set the edge direction from node i to node j
            DAG(i, j) = 1;  % Heuristic: i → j
        end
    end
end
```

This results in a sparse, acyclic structure that reflects only statistically significant dependencies.

A.6.5.1 Mutual Information

This supporting function computes MI between two variables, optionally conditioned on a third. The MI quantifies how much knowing one variable reduces uncertainty about the other.

For unconditioned MI,

```
% Update MI, add contribution from the joint frequency of (u, v)
MI = MI + (Frequency(u,v)/M) * log2(Frequency(u,v)*Sum / (Fu(u)*Fv(v)));
```

Where,

- `Frequency(u,v)` is the joint frequency count of $X = u$, $Y = v$.
- `Fu(u)`, `Fv(v)` are marginal frequencies.

- M is the total number of observations.

If a third variable is supplied (conditional MI), the function partitions the data according to the conditioning variable and computes the MI for each subgroup.
This approach ensures that even indirect influences can be detected and handled, supporting more robust edge pruning.

A.6.5.2 Chi-Square Test

Once the MI has been computed, the independence test determines whether it is statistically significant. This is implemented using a chi-squared threshold,

```
Threshold = chi2inv(1-a, R); % Critical value for chi-squared distribution
CI = Threshold < 2*M*MI; % Compare MI statistic to threshold
```

Where,

- α is the significance level.
- R is the degrees of freedom (typically $(r_i - 1)(r_j - 1)$).
- M is the total frequency.
- The result `CI` is a Boolean indicating independence (`true`) or dependence (`false`).

This test helps avoid spurious edge inclusion, especially in sparse datasets, and provides statistical grounding for the learned structure.

A.6.6 Learning Object Builders

Both the K2 and NPC methods rely on a common learning object structure, `LGObj`, which stores all required data characteristics. These builder functions perform preprocessing steps to convert the raw MATA-D matrix into the format required for scoring or testing.
The key contents of `LGObj` include,

- `FreqTable`, cell array of conditional frequency tables for each variable.
- `r`, Number of discrete states for each variable.
- `RangeVecs` vectors indicating valid state ranges, used for indexing and table construction.
- `NumCases`, total number of training samples.

This layer supports compatibility across the different learning algorithms.

A.6.7 Expert and Aggregated Models

If `expert` is selected, a DAG is loaded directly from Excel. If `aggregated` is selected, all three models (K2, NPC, Expert) are computed and merged,

```
DAG = (DAG_K2 + DAG_NPC + DAG_Expert) >= 2;
```

This ensures that only relationships agreed upon by a majority are retained.

A.6.8 Final DAG Visualization

The tool visualizes the final directed acyclic graph,

```
G = digraph(DAG);
plot(G);
```

This plot provides an interpretable representation of inferred causal structure among human factors.

The full code for the *HF Relations tool* is available at `https://github.com/VirtualRaphael/Human-Factors-Relationships-Tool`.