TÉCNICO LISBOA

Prediction of Human Factors in Aviation Incident Reports using Machine Learning and Natural Language Processing

Tomás Moitinho de Almeida Andrade Madeira

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisor(s): Prof. Duarte Pedro Mata de Oliveira Valério
Prof. Mário Rui Melício da Conceição

Examination Committee

Chairperson: Prof. Carlos Baptista Cardeira
Supervisor: Prof. Mário Rui Melício da Conceição
Members of the Committee: Prof. João Paulo Baptista de Carvalho
Prof. Luís Filipe Ferreira Marques Santos

January 2021

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my two supervisors, Prof. Duarte Valério and Prof. Rui Melício, for their unconditional support and cooperation throughout the dissertation period. Their guidance and knowledge was absolutely determinant for the success of the developed work.

Besides my supervisors, I would like to thank Harro Ranter for kindly providing me access to the ASN Aviation Safety Database for research purposes.

Finally, I would like to express my profound gratitude to my supportive parents, friendly colleagues and lovely Joana, whom each in their own way, were crucial during the course of this challenging year, believed in me and supported me in every way they could. This accomplishment would not have been possible without all of them. Thank you.

Resumo

No setor da aviação, os fatores humanos são a principal causa de incidentes de segurança. Sistemas inteligentes de previsão, capazes de avaliar o estado humano e gerir riscos, têm vindo a ser desenvolvidos ao longo dos anos para identificar e prevenir fatores humanos. No entanto, a falta de grandes conjuntos de dados rotulados tem frequentemente dificultado o desenvolvimento destes sistemas. O objetivo desta dissertação é propor e implementar um modelo preditivo que consiga identificar e classificar categorias de fatores humanos com base em relatórios de incidentes de aviação. Os dados de estudo, fornecidos pela rede de segurança da aviação (ASN), contemplam 1.674 relatórios de incidentes entre 2000 e 2020. Estes relatórios contêm detalhes da aeronave, rota planeada, causa provável e outras informações do voo.

Um novo sistema de classificação de fatores humanos é proposto e um conjunto diversificado de dados rotulados é desenvolvido, com base nos dados adquiridos. Para a extração de atributos, é desenvolvido um pipeline de pré-processamento de texto e Processamento de Linguagem Natural (NLP). Para a modelagem de dados, são consideradas a técnica semi-supervisionada propagação de rótulo (LS) e a técnica supervisionada máquina de vetor de suporte (SVM). Métodos de busca aleatória e otimização Bayesiana são aplicados para a análise de hiperparâmetros e melhoria do desempenho do modelo, medido pela pontuação Micro F1.

Os melhores modelos preditivos alcançaram pontuações Micro F1 de 0.900, 0.779 e 0.875, para cada nível do sistema de classificação, respectivamente. A solução proposta indica que pontuações de previsão favoráveis podem ser atingidas na classificação de fatores humanos com base em dados de texto. Não obstante, é recomendada a utilização de um conjunto maior de dados em futuras investigações.

Palavras-chave: Aprendizagem Automática, Processamento de Linguagem Natural, Classificação, Fatores Humanos, Segurança da Aviação.

Abstract

In the aviation sector, human factors are the primary cause of safety incidents. Intelligent prediction systems, capable of evaluating human state and managing risk, have been developed over the years to identify and prevent human factors. However, the lack of large useful labelled data has often been a drawback to the development of these systems. The aim of this dissertation is to propose and implement a predictive model that can identify and classify human factor categories from aviation incident reports. The study data, provided by the Aviation Safety Network (ASN), comprises 1674 incident reports between 2000 and 2020. These reports consist of aircraft details, planned route, probable cause, and other flight information.

A novel human factor classificatory framework is proposed and a diversified labelling set is developed, based on the acquired data. For feature extraction, a text pre-processing and Natural Language Processing (NLP) pipeline is developed. For data modelling, semi-supervised Label Spreading (LS) and supervised Support Vector Machine (SVM) techniques are considered. Random search and Bayesian optimization methods are applied for hyper-parameter analysis and improvement of model performance, measured by the Micro F1 score.

The best predictive models achieved a Micro F1 score of 0.900, 0.779 and 0.875, for each level of the proposed framework, respectively. The proposed solution indicates that favourable predicting performances can be achieved for the classification of human factors based on text data. Notwithstanding, a larger data set would be recommended in future research.

Keywords: Machine Learning, Natural Language Processing, Classification, Human Factors, Aviation Safety.

Contents

List of Tables

List of Figures

Nomenclature

Symbols

v_w	Vector projection of the wth word
P	Embedding dimensional
d_i	Vector projection of the ith document
T_i	Set of ordered words from the ith document
N	Number of documents
V	Set of distinct words from the corpora
$q_i(w)$	Weight of the wth word in the ith document
$f_i(w)$	Frequency of the wth word in the ith document
$f_N(w)$	Number of documents the wth word appears
$C(w)$	Set of context words of the wth word
T	Set of ordered words from the corpora
L	Set of context word pairs
W	Word embedding matrix
W'	Context matrix
v_{w_I}	Vector projection of the wth input word
v_{w_O}	Vector projection of the wth output word
y	Softmax prediction vector
t	True one-hot encoded label vector
e	Error vector
μ	Learning rate
Z	Number of Negative Sampling words

$p(w)$ Probability of choosing the wth word for Negative Sampling

$f(w)$ Frequency of the wth word in the corpora

$n(w, j)$ Hierarchical Softmax jth node on the path from the root to the wth word

$L(w)$ Length of Hierarchical Softmax from the root to the wth word

σ Sigmoid function

D Document embedding matrix

S Cosine similarity

Th Word similarity threshold

α Label Spreading regularization parameter

L Number of labelled documents

$Gamma$ RBF kernel parameter

K RBF kernel affinity

TP_a Number of true positives from the ath category

FP_a Number of false positives from the ath category

TN_a Number of true negatives from the ath category

FN_a Number of false negatives from the ath category

A Set of main HFACS-ML categories from a certain level of the framework

Ts Train size

Acronyms

ICAO International Civil Aviation Organization

SMS Safety Management System

MRM Maintenance Resource Management

HRA Human Reliability Assessment

PROCOS Probabilistic Cognitive Simulator

PSFs Performance Shaping Factors

NN Neural Networks

HFACS Human Factors Analysis and Classification System

ASN Aviation Safety Network

NLP Natural Language Processing

D2V Doc2Vec

LS Label Spreading

RBF Radial Basis Function

TF-IDF Term Frequency–Inverse Document Frequency

fANOVA functional Analysis of Variance

IoT Internet of Things

ASRS Aviation Safety Reporting System

ASI Air Safety Institute

HFACS-ML Human Factors Analysis and Classification System - Machine Learning

HFACS-MA Human Factors Analysis and Classification System - Maintenance Audit

VSM Vector Space Model

BoW Bag of Words

W2V Word2Vec

DG Skip-Gram

CBoW Continuous Bag of Words

NS Negative Sampling

HS Hierarchical Softmax

PV-DM Distributed Memory for Paragraph Vectors

PV-DBoW Distributed Bag of Words for Paragraph Vectors

iD Identification Digit

t-SNE t-Distributed Stochastic Neighbor Embedding

knn K nearest neighbours

GP Gaussian Process

SMBO Sequential Model-Based Optimization

EI Expected Improvement

SVM Support Vector Machine

LDA Latent Dirichlet Allocation

Abbreviations

n/a Not Available

und Undetermined

Precondition Precondition for Unsafe Act

Physical Env. 1 Physical Environment 1

Physical Env. 2 Physical Environment 2

Planned Inap. Oper. Planned Inappropriate Operations

Failed Known Prob. Failed Known Problem

Technological Env. Technological Environment

Doc Document

Chapter 1

Introduction

1.1 Motivation for Designing Aviation Safety Predictive Models

1.1.1 The Current Welfare of Aviation Safety

Over the last decades, air transport has been considered as one of the fastest and safest methods of transportation for long-distance travel, being one of the largest contributors to the growth of political, social, and economic globalization.

Before the current coronavirus pandemic, the commercial aviation sector was annually deploying over 37 million airplane departures and 4 billion passengers worldwide, with the International Civil Aviation Organization (ICAO) expecting these numbers to reach 90 million and 10 billion, respectively, by 2040 [1]. Although these numbers have been strongly affected by the pandemic, some projections estimate that global air traffic could reach 2019 levels as early as 2024 [2]. Figure 1.1 shows the continuous growth of worldwide commercial air traffic over the past decade, in billions of passengers.

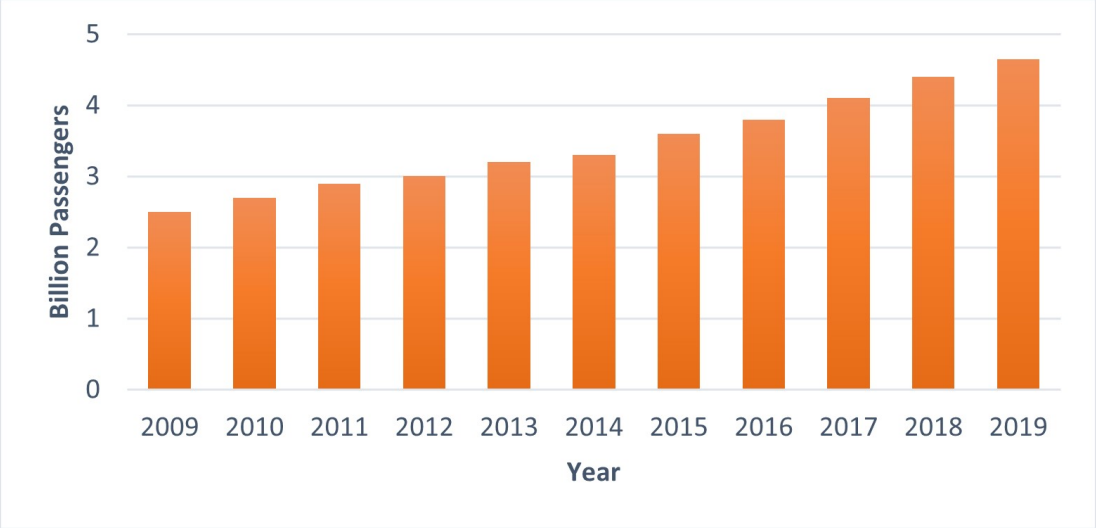

Figure 1.1: Worldwide commercial air traffic evolution in the last decade (source [3]).

Hence, to keep up with the increasing demand generated by the economic growth in emerging mar-

kets, rise in population, and introduction of low-cost carriers, airlines have been on a fast large-scale expansion, through increasing their assets, personnel, and infrastructure, while maintaining competitive prices. Some instances representative of this growth can be found in [4, 5].

This fast expansion has brought safety concerns into numerous sectors of the aircraft industry. Oftentimes the increase in workload is not accompanied by a proportional gain in personnel, and as a consequence, employees are subjected to an enormous quantity of rigorous requests, under pressured time frames and complex environments. Moreover, compounded factors such as career uncertainty and frequent demand for overtime, have contributed to an increasing risk of personnel capabilities detriment, and hence mistakes in safety sensitive tasks [6]. These circumstances have been further aggravated by the effects of the current pandemic, where companies had to forgo large portions of their employees in order to reduce cash burns and regain profit margins, while maintaining contractual obligations and preparing for a resurgence of traffic demand.

In fact, although the total number of yearly accidents has had a significant decrease over the past decades due to rapid technological developments, human factors have taken the lead as the main latent cause of the overall incidents (Figure 1.2). Studies such as [7–9] have found pressure, fatigue, miscommunication, and lack of technical knowledge on crucial personnel - such as maintenance workers, air crew, and air traffic controllers - to be some of the main probable causes for aviation mishaps.

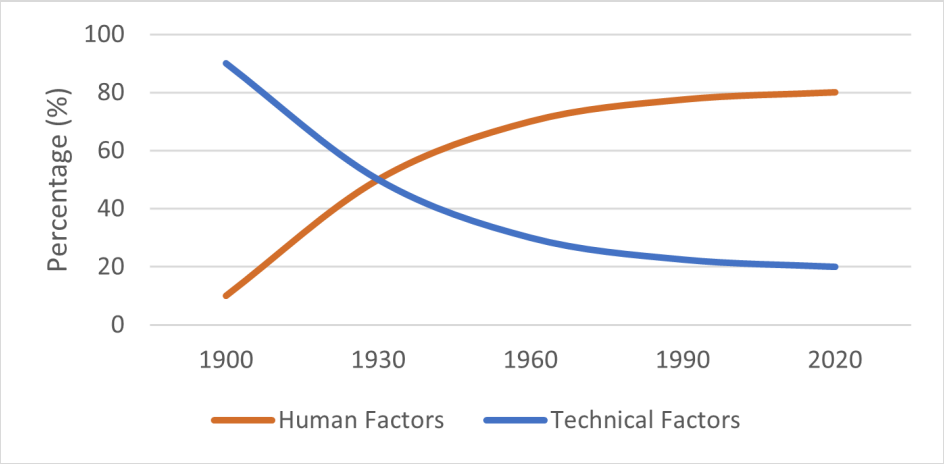

Figure 1.2: Evolution of accident causes (adapted from [6]).

To address these issues, international regulatory agencies compel airlines to use frameworks such as Safety Management System (SMS) and Maintenance Resource Management (MRM), which use periodic inspections, standardized audits, and performance-based approaches to identify safety breaches [10]. Notwithstanding, according to ICAO, in 2018 there were 98 aircraft accidents for scheduled commercial air transport operations, of which 11 were fatal accidents, resulting in 514 passenger fatalities [1]. This report still reflects a long way for aviation safety improvement. Consequently, The aviation industry should strongly adopt the strategy that ”one accident is already too much”. In addition to the existing reactive tools, there is also a high need for the implementation of predictive human factor safety models that can detect and prevent high-risk situations.

1.1.2 Predictive Human Factor Safety Models

The arguments presented in Section 1.1.1 justify the research that has been done over the past decade, related to the development of proactive safety systems that can quantify human performance and measure the potential for human error, before accident occurrence.

In the field of human factors, there are several Human Reliability Assessment (HRA) tools for measuring the probability of human errors. A detailed overview of the different methods developed over the years can be found in [11]. In sum, these can be divided into two main categories: model-based and data-based.

Some of the most recent HRA model-based developments, used in high-risk industries like nuclear power plants, involve cognitive simulators such as the PROCOS [12], which combine a realistic plant design of the workspace with theoretical mental process computational reproductions, capable of modelling Performance Shaping Factors (PSFs) of human behaviour. Although the interest in these solutions to provide qualitative results for complex environments, they often either oversimplify static frameworks with limited depth and practical value, or require a full integration, tailored to very specific applications and environments, being therefore costly to reproduce for different systems.

Recently, it is notable an increase in a common effort within the research community to develop data-based HRA processes that can prove more accessible to implement, while availing value from already acquired data. Some examples of these solutions, specifically designed towards predictive safety models, are [13, 14]. These approaches seek to correlate safety audit findings with posterior incidents, through the use of Neural Networks (NN), in order to quantitatively evaluate the urgency of detected issues, based on their future expected impact.

Unfortunately, some of these methods, especially those which rely on the contents of descriptive reports, require manual categorization of human factor categories, often retracted from the established Human Factors Analysis and Classification System (HFACS), shown in Figure 1.3. Note that the HFACS framework has been widely used to categorize human causes of accidents and serve as a means to establishing safety training and prevention efforts [15]. This slow and human error-prone process impairs the scalability of these methods towards other data sets and industries.

Consequently, there is a strong need for an automation of this process, which enables to generate pseudo-labels that can be used for aviation safety and work health research, as well as to provide immediate statistics of which human factors lead to a higher number of incidents in a given field and should therefore be more urgently mitigated.

The general problem of inferring taxonomic information from text data is not novel and has been extensively explored in other fields of research, such as healthcare and journalism. Some examples of successful applications have been the prediction of patient illness based on medical notes [16, 17] and automated fake news detection from internet pages [18]. Surprisingly, to our knowledge, only a few studies have tried to infer information from aviation safety reports using NLP [19].

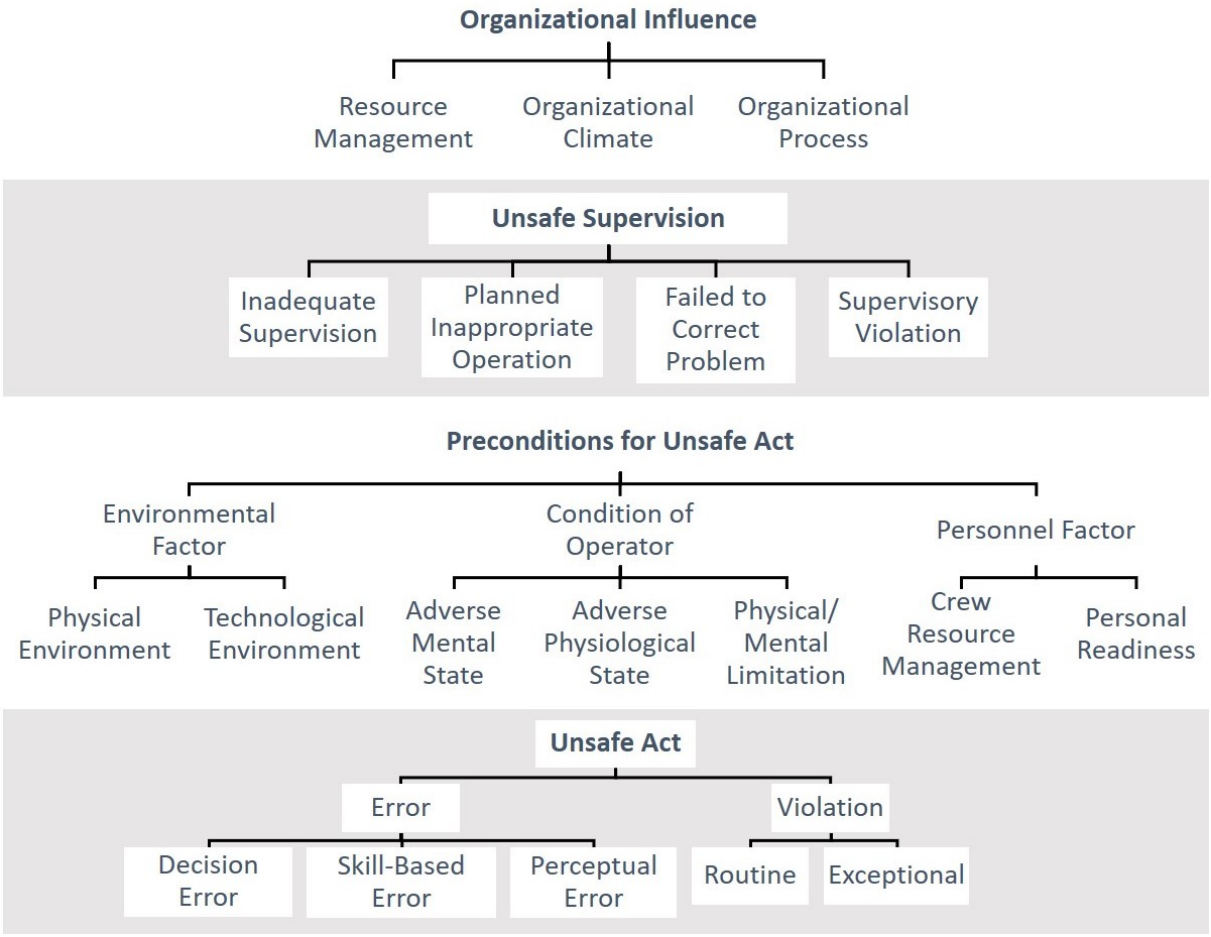

Figure 1.3: HFACS framework (adapted from [15]).

1.2 Aims and Contributions

1.2.1 Aims

In a wide perspective, the aim of this research is to contribute to a better knowledge about how to enhance aviation safety. In a more confined sense, the aim is to develop a comprehensive framework based on data mining and machine learning techniques, inspired by comparable solutions from other fields, to make predictions over the main human factors causal of aviation incidents, based on descriptive text data. In this context, the scientific questions which are intended to be solved can be expressed in the following terms:

- Can machine learning techniques be used to classify aviation incident reports, based on their latent human factors?
  - If yes, can this classification be used for pseudo-label generation?
  - If yes, can this classification be used for statistical analysis?
- Can semi-supervised classification approaches outperform supervised techniques, for small labelled data sets?

From a practical perspective and in order to contribute to the development of a structured data-driven solution, we also aim to shape our approach to follow a standard text classification project development workflow. Figure 1.4 presents a high-level view of the workflow sough out for this dissertation.

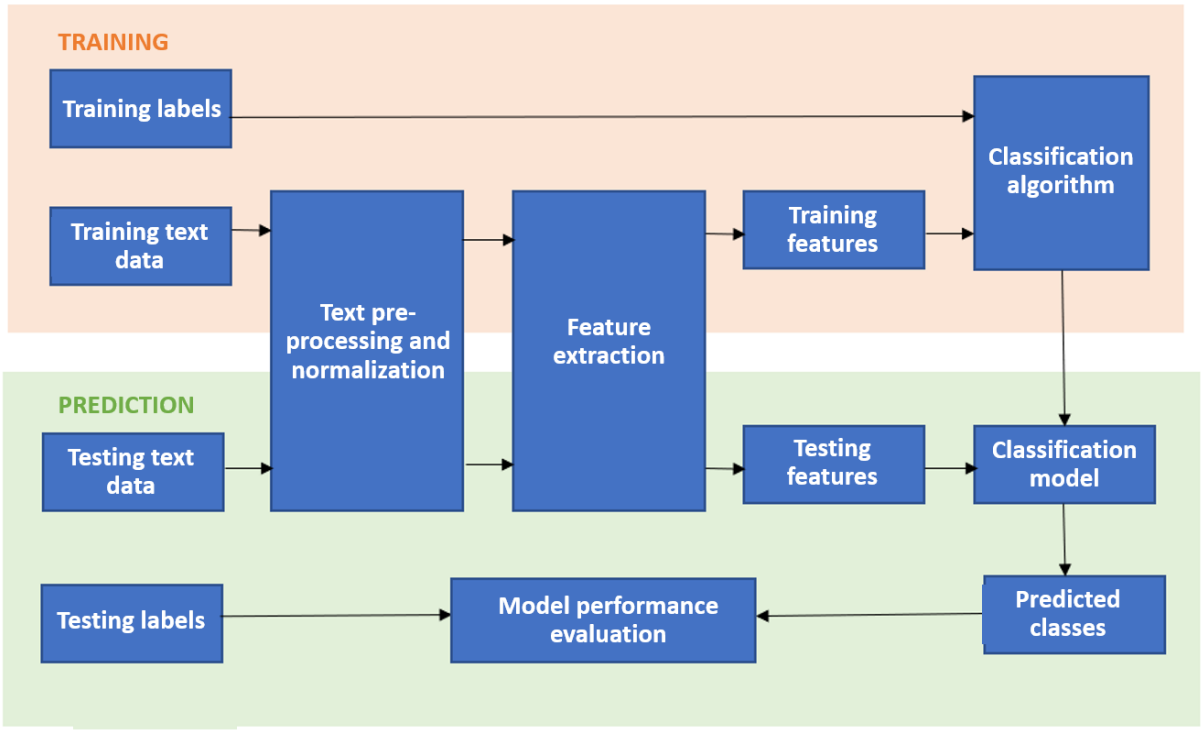

Figure 1.4: Text classification project development workflow (adapted from [20]).

1.2.2 Contributions

The data set in which this application was developed comprises incident text reports, kindly made available by the Aviation Safety Network (ASN). To exploit the specific characteristics of the ASN database, we started by performing a comprehensive analysis of the data, followed by a tailored pre-processing which included data selection and NLP data preparation. In addition, an adapted HFACS framework is proposed for this particular study, and a small labelled set is constructed for later use.

Differently from [14], where each document is categorized manually, in this work, we directly convert the pre-processed documents into numerical vectors, representative of their content, through the use of feature extraction techniques. In the process, we take some preliminary conclusions as to the efficiency of various commonly used approaches, by evaluating the resulting vectors based on their relative position in the state space.

To avail both the labelled and unlabelled data, we apply a semi-supervised transductive label propagation algorithm (LS) to the developed models. The advantage of this algorithm is that it enables to classify the documents with their most likely human factor, by measuring vector similarities through a non-linear Gaussian RBF kernel and by spreading known taxonomic information through graph propagation. The transductive algorithm is tested by running multiple simulations on different optimizers, within predefined simulation scenarios, and compared to the supervised SVM. The results regarding

performance, robustness, and computation time are discussed.

The classification algorithm was developed in Python3. The author's main contributions to this platform were the management of a data preparation and NLP pipeline, as well as a Bayesian optimization integration for graph classification. Beyond the models described in this dissertation, a variety of unsupervised clustering methods (namely K-Means, Bisecting K-Means and Spectral Clustering) were also considered and implemented during an early stage of the investigation. However, due to their low effectiveness and applicability to the proposed objectives, these were omitted from most of the work.

The results of this dissertation are expected to make a contribution on the current body of knowledge in the relevant field of study.

1.3 Dissertation Outline

The dissertation is organized as follows:

Chapter 2 - Tailored Data Analysis of Aviation Incident Reports

In this chapter, a tailored data analysis is conducted. Topics such as data comprehension, data preprocessing and label generation, are explored.

Chapter 3 - Feature Extraction with Natural Language Processing

A principled approach for converting text excerpts into numerical vector representations is highlighted in this chapter, through the use of Natural Language Processing feature extraction techniques. The importance of spacial distribution is presented, and the modeling procedure for both document embeddings (W2V, D2V) and vector space models (TF-IDF) is carefully described.

Chapter 4 - Human Factor Label Propagation

In this chapter we propose an integration of a Label Propagation approach (LS) that, together with the human factor labels extracted in Chapter 2 and the vector representations presented in Chapter 3, spreads taxonomic information throughout the vector space. Additionally, the classification performance is tested and improved, through the implementation of Bayesian optimization, supported by an fANOVA hyper-parameter importance evaluation. A comparison of the metric results with other alternative models is also presented.

Chapter 5 - Conclusions and Future Work

The final chapter summarizes the most pertinent conclusions, aims to answer the initial research questions presented in Chapter 1 and expands on some ideas for future work.

Chapter 2

Tailored Data Analysis of Aviation Incident Reports

This chapter describes a tailored data analysis of the aviation incident reports carried out for this dissertation. Section 2.1 starts with a description of the reports and a selection of information relevant to the study. After that, Section 2.2 expands on the construction of a labelled set, to be used for later training and testing of predictive models. Then, Section 2.3 addresses the various pre-processing steps applied to the raw data before being used for feature extraction. Finally, Section 2.4 debates over the main data changes and comments on their possible impact on the development of predictive models.

2.1 Raw Data Collection

A common challenge in text data mining is the acquisition of corpora that is representative enough to distinctively capture semantic information relevant to the pursued taxonomy, and large enough to allow for various patterns to be revealed. Over the last decade, it is notable an increase in synergies, between the research community and the industry, to improve data acquisition systems through the development of complex database platforms and Internet of Things (IoT) technologies.

In the aviation safety sector, there may be found various incident databases, such as the ASRS and the ASI. Notably, we found the publicly available ASN database to be the most appropriate for the current study since, to date, it possesses more complete and consistent information regarding the incidents and their contributing factors, whilst covering official accidents and safety issues of airliners, military transport planes, and corporate jets from all over the world [21]. Note that airliners are considered here as aircrafts capable of carrying at least 12 passengers. Figures 2.1 and 2.2 illustrate the type of data available in this database, using as an example Tassili Airlines' 2004 accident.

Status:	Final
Date:	**Wednesday 28 January 2004**
Time:	21:01

Type:	Beechcraft 1900D
Operator:	Tassili Airlines
Registration:	7T-VIN
C/n / msn:	UE-365
First flight:	1999
Total airframe hrs:	1742
Engines:	2 Pratt & Whitney Canada PT6A-67D
Crew:	Fatalities: 1 / Occupants: 3
Passengers:	Fatalities: 0 / Occupants: 2
Total:	Fatalities: 1 / Occupants: 5
Aircraft damage:	Destroyed
Aircraft fate:	Written off (damaged beyond repair)
Location:	5 km (3.1 mls) S of Ghardaïa-Noumérat Airport (GHA) (Algeria)
Phase:	Approach (APR)
Nature:	Domestic Non Scheduled Passenger
Departure airport:	Hassi R'Mel-Tilrempt Airport (HRM/DAFH), Algeria
Destination airport:	Ghardaïa-Noumérat Airport (GHA/DAUG), Algeria

Narrative:

The Sonatrach company chartered one of Tassili Airlines' Beechcraft 1900D planes to fly two employees from the oil fields near the Algerian Sahara town of Hassi R'Mel to Ghardaïa. The copilot was Pilot Flying. The Beech took off at 20:30 and arrived near Ghardaïa fourteen minutes later.

At 20:44 the flight was cleared for a right hand circuit in preparation for an approach to runway 30. At that moment a Boeing 727 was on long finals. The copilot stated that he intended to carry out an NDB/ILS approach to runway 30. The captain however preferred a visual approach. The copilot carried out the captain's course and descent instructions with hesitation. At 20:57, the EGPWS alarm sounded. Power was added and a climb was initiated from a lowest altitude of 240 feet above ground level.

The captain then took over control and assumed the role of Pilot Flying. The airplane manoeuvred south of the airport until 21:01 when the copilot saw the runway. The captain rolled left to -57° and pitched down to -18.9° in order to steer the airplane towards the runway. Again the EGPWS sounded but the descent continued until the airplane impacted the ground and broke up.

The five occupants survived the impact, but the co-pilot died a day later of his injuries.

Figure 2.1: Example of an incident report from the ASN database, part 1 (source [21]).

Probable Cause:

Causes of the accident (translated from French):
The Commission believes that the accident can be explained by a series of several causes which, taken separately, would not lead to an accident.

The causes are related to:
1 - the lack of rigor in the approach and landing phase evidenced by a failure to follow standard operating procedures, including the arrival checklist.
2 - the failure to strictly comply with the holding, approach and landing procedures in force for the aerodrome of Ghardaïa.
3 - the fact that the captain seemed occupied by the visual search maneuvers that put him temporarily out of the control loop.
He was so focused on the visual search for the runway and abandoned the monitoring of parameters that are critical for the safety of the flight. This concentration completely disoriented him.
4 - the fact that the crew did not respond appropriately to different alarms that occurred, indicating a lack of control in the operation of the aircraft in that kind of situation.
Lack of control was apparently due to his lack of training on this aircraft type.
5 - The activities in the southern part of Algeria may cause a certain routine that can promote the tendency to conduct visual approaches.
It seems, indeed, that the crew is more experienced in visual flights.
6 - A lack of coordination and communication between the crew members flying together for the first time.

Classification:

Controlled Flight Into Terrain (CFIT) - Ground

Figure 2.2: Example of an incident report from the ASN database, part 2 (source [21]).

To make the best use of the information available in the database, while considering the most recent threats to aviation safety for posterior statistical analysis and pseudo-label extraction, in the present study, we narrowed the time frame to the last two decades (from 2000 to 2020) of aviation incidents, amounting a total of 3276 reports. However, since not all the available descriptive segments (shown in Figures 2.1 and 2.2) were considered to be relevant to the study, only some details were extracted. The extracted segments and their respective descriptions follow bellow.

- Narrative: Description of the occurrence;
- Probable Cause: Main contributing factors which lead to the cause of the incident, as established by the accident investigators;
- Classification: Keywords related to the cause, type of collision, and/or consequences of the occurrence.

Although these three segments may provide contextual relevance to the study of human factors, they still encompass very different characteristics and, as such, were examined individually.

We noted that while the ”Narrative” segment was mostly descriptive of technical aspects and sequential deeds, the ”Probable Cause” segment often explored in greater depth the contributing factors which lead to the cause of the incidents, and provided clearer indicators of human interference, which could be linked to human factor categories. This led us to chose the second segment as the main source of information for incident/document classification and the first segment as complementary evidence, for clarification purposes. Lastly, although the keywords found in the ”Classification” segment did not present available or relevant human factor information for most occurrences, we found that it could still provide a valuable tool for extracting information correlated to some specific categories.

2.2 Construction of a Labelled Data Set

In data science, labelled data is a designation for pieces of data, in this case the ”Probable Cause” documents, that have been tagged with one or more labels identifying certain properties or characteristics in their content. Labelled data is especially useful in the development of certain types of machine learning models, as it can act on the orientation of training and validating simulations. Considering that we do not possess a reliable labelled set which can accurately correlate incident reports to their latent human factors, in this section, we diligently construct a small labelled set and define onto which human factor categories it may be reasonable to base the classification on, taking into consideration the specific characteristics of the ASN database.

2.2.1 Human Factor Classification Framework

After a comprehensive examination of a high number of ”Probable Cause” documents, it was concluded that it would not be appropriate to force a classificatory algorithm to make predictions regarding every

single HFACS category from the original framework (Figure 1.3). In the first place, causal factors referring to Organizational Influences are rarely mentioned in the incident investigations reports. This may be due to the information gap between the knowledge provided to investigators and the real upper-level management practices. This justifies the exclusion of this level of the framework from the study. Secondly, most subcategories in the original framework retract to very specific situations, whose information is often not evident in the descriptions, or is biased to subjectivity. An example of these instances is the differentiation between skill-based and decision-based errors. For this reason, a higher-level outlook of the framework has been taken, retracting the previous example to be simply classified as ”Error”. Lastly, we observed that the ”Physical Environment” category encompassed two frequent types of Preconditions for Unsafe Act with very different core vocabularies: weather and animals. On this account, we found fit to subdivide this category into two: ”Physical Environment 1”, referring to weather-related preconditions; and ”Physical Environment 2”, referring to animal-related preconditions. In-depth descriptions of the remaining HFACS categories, common to both frameworks, can be found in [15].

As a result, Figure 2.3 presents a revised HFACS framework, adapted for machine learning (ML) research and based on the current data set. Its distinctiveness lies on the potential to optimize the classification task by grouping documents that have common human factor characteristics in the same class, while distinguishing between documents of clearly different classes. This newly proposed framework will be used as the main classificatory guide for the rest of the study, being referred to as HFACS-ML. It is worthwhile to note that this study is not the first to propose domain specific modifications of the original HFACS. See [13] for the HFACS-MA framework, adjusted to the maintenance sector.

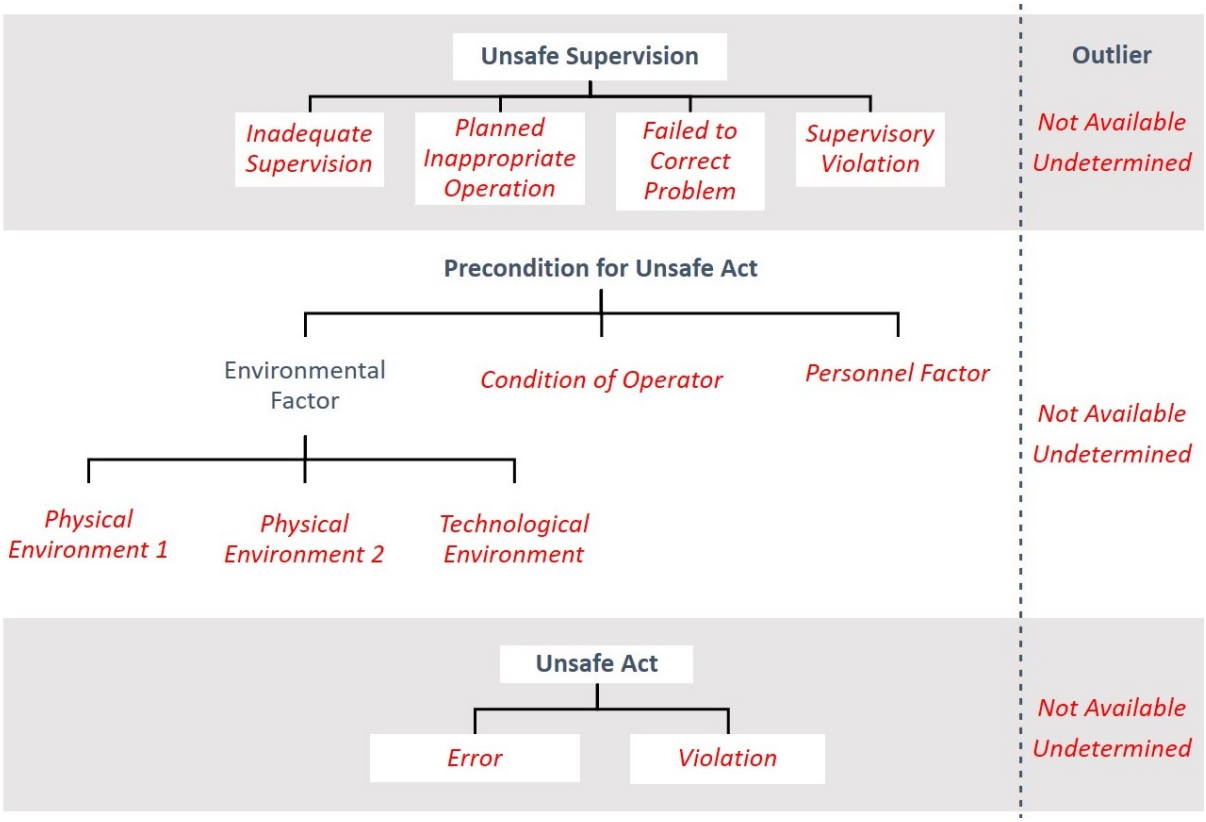

Figure 2.3: Revised HFACS-ML framework.

Similarly to the original HFACS framework, in the proposed categorization model, each document may have a minimum of zero labels, in the case of those that did not participate in the labelling process, and a maximum of three labels (with at most one label per level) for those that were found descriptive enough to be classified for all three levels of the framework. The adopted categories are represented in red, in Figure 2.3.

A last modification that has been included in the revised HFACS-ML framework was the introduction of outlier categories: ”Not Available” and ”Undetermined”. To deal with the lack of information frequently encountered in the comprehensive analysis, documents which were observed not to belong to any category, for a certain level, were considered ”Not Available” (or ”n/a”), if the information was not sufficient to conclude about their category. However, should the report explicitly inform that the cause could not be determined in the course of the investigation, then the document would be considered ”Undetermined” (or ”und”). Note that the outlier categories were introduced separately for each level of the framework. By adding these categories, we aimed to use the outlier labels for the detection of other unlabelled outlier documents, which should not be classified into the main HFACS-ML categories, therefore improving predictive reliability.

2.2.2 Data-Driven Automated Labelling

Having defined the categorization framework, we proceeded to construct the labelled set. Different approaches could have been taken to this end, such as crowd-sourcing and the various types of automated data labelling. However, due to lack of resources and prioritization over ground truth control, we chose two simple and efficient methods: Data-driven automated labelling and manual labelling. Note that ground truth is referred here to the initial labels that are attributed to the documents during either labelling process. Likewise, this process might be exposed to subjectivity, as it is derived from direct text interpretation.

For the automated labelling method, it seemed appropriate to analyse if, from the ”Classification” segment keywords (whose nature is described in Section 2.1), some associations could be made regarding HFACS-ML categories. During this process, we realised that not all keywords provided consistent patterns that could be correlated to a single HFACS-ML category. In these cases, no label was attributed. However, few particular keywords were indeed found to be consistent with a single category for one level of the framework, and in some cases, up to two levels of the framework were found to be satisfied. Keyword consistency was defined here as the observation of at least 15 random documents, all belonging to the same category, with the exception of at most 3 documents which were allowed to be out of the norm. In the cases where consistency was satisfied, all documents which possessed this keyword were equipped with the respective human factor label, except for the irregular samples which were manually corrected. Naturally, in the cases where keywords appeared less than 15 times, the consistency criterion count was reduced to the size of the set and, in these cases, all instances were analysed.

Due to the high number of keywords, and in order to avail their semantic meaning, only those that could potentially hold a rational correlation to any of the HFACS-ML categories were considered. Table

2.1 describes all keyword associations that were found to satisfy the consistency criterion, and hence contributed to the data-driven automated labelling process. Moreover, the full list of keywords extracted from the database, in which the development of this process was based, can be found in Appendix A.

Table 2.1: Automated labelling table.

	HFACS-ML Categories		
Keyword	**Unsafe Supervision**	**Precondition**	**Unsafe Act**
Weather - (all)	-	Physical Env. 1	-
ATC & navigation - VFR flight in IMC	-	Physical Env. 1	-
ATC & navigation - Language/communication	-	Personnel Factor	-
Collision - Object - Bird	-	Physical Env. 2	-
Collision - Object - Person, animal	-	Physical Env. 2	-
Airplane - Engines - Fuel exhaustion	Supervisory Violation	Personnel Factor	-
Airplane - Engines - Fuel starvation	Supervisory Violation	Personnel Factor	-
Flightcrew - Alcohol, drug usage	-	Condition of Operator	-
Flightcrew - Incapacitation	-	Condition of Operator	-
Flightcrew - Disorientation, sit. awareness	-	Condition of Operator	Error
Flightcrew - Insufficient rest / fatigue	-	Personnel Factor	-
Flightcrew - Non adherence to procedures	-	-	Violation
Cargo - Overloaded	-	-	Violation
Flightcrew - Un(der)qualified	Supervisory Violation	Personnel Factor	-
Security - Suicide	-	Condition of Operator	Violation

2.2.3 Manual Labelling

Although a considerable amount of labels was attained in the first labelling method, the distribution of the resulting set proved very imbalanced and not much diversified. As such, in order to add variety, as well as a possible balance to the labelled set, we saw fit to add manual labelling to the process. To this end, and for the general document analysis, some guidelines were established.

A document was considered to belong to a certain category if the respective human factor was evident to be the cause of the incident and no other human factors were mentioned in the text. In the cases where multiple human factors from the same level were mentioned, the label for that document appertained to the most prevalent factor, that which revealed to be the most consequential to the accident and appeared with more frequency in the text. Should this feature not be evident, the first chronological mishap identified, that which triggered the chain of human misbehaviour, was defined as the prevalent category.

Another procedure that was established, in this case exclusive to the manual labelling process, was the attribution of outlier labels ”n/a” or ”und”, depending on the situation, to those documents that did not contain any conclusive information regarding their category for a certain level of the framework. This was done only for the manual process since all documents were assured to be individually assessed.

Throughout the course of this study, more than 60 documents were individually analyzed and classified onto their respective HFACS-ML categories. The result of both labelling processes lead to a total classification of 107 Unsafe Supervision labels, 370 Precondition for Unsafe Act labels and 119 Unsafe Act Labels. Table 2.2 displays the entire distribution of labels over the different levels and categories of the revised framework. Concerning this process, Table 2.3 further illustrates, as an example, the labels

given to the previously shown 2004 Tassili Airlines accident. Note that the identification number ("iD") corresponds to the document's place in the extracted database.

Table 2.2: Total label distribution.

HFACS-ML Level	HFACS-ML Category	Label count
Unsafe Supervision	Inadequate Supervision	20
	Planned Inap. Oper.	6
	Failed Known Prob.	3
	Supervisory Violation	52
	n/a	22
	und	4
Precondition for Unsafe Act	Physical Env. 1	169
	Physical Env. 2	46
	Technological Env.	10
	Condition of Operator	55
	Personnel Factor	78
	n/a	8
	und	4
Unsafe Act	Error	54
	Violation	47
	n/a	14
	und	4

Table 2.3: Example of manual labelling.

	HFACS-ML Level		
iD	Unsafe Supervision	Precondition	Unsafe Act
361	Inadequate Supervision	Personnel Factor	Error

The reasoning behind the classification of this report is as follows: Due to his/her lack of rigor and training, the pilot in command exhibited characteristics of an inadequate supervisor, by generating inefficient and unsafe oversight to the procedures and rest of the team. Furthermore, the lack of coordination and communication between crew members, intensified by the fact that it was their first flight together, demonstrated to be one of the main preconditions for the accident, as it reflected poor crew resource management and lack of personnel readiness. Lastly, the main unsafe act, causal of the accident, could be discussed here between a violation of standard operating procedures by the pilot, or skill-based and perceptual-based errors by the pilot and crew. However, taking into account the latter is more frequently mentioned in the report and was interpreted to have a broader impact on the sequence of occurrences, the unsafe act was considered here an error.

2.3 Pre-Processing

In data mining, the presence of irrelevant information, often found in raw text data, is known to substantially condition the performance of predictive models. Pre-processing analyses the syntactical aspects of sentences and words to improve data quality, reduce noise and optimize the vocabulary for subsequent feature extraction. Since, to our knowledge, no studies have tried to explore which pre-processing tools result most efficient for aviation incident report analysis, we took inspiration from applications to other settings, such as [22, 23], for the implementation of our tailored pipeline.

Notably, in [22], a comparative study was carried out to analyze the impact of common pre-processing techniques across various studies and data sets on the task of text classification. This study revealed that lower-casing presented positive remarks over most data sets and that stemming also generally presented a positive impact, specially for English texts. In contrast, other common pre-processing tools such as alphanumeric parsing and stop-word removal were found to be more domain dependant and should therefore be adapted for each context. A more detailed description regarding these and other pre-processing tools, as well as their practical application, can also be found in [23].

After a comprehensive analysis over the ”Probable Cause” information and the various pre-processing tools, we understood that a robust pre-processing scheme, tailored to the current data set, could be divided into three stages: Data cleaning, Normalization and Tokenization. The entire scheme is presented in Figure 2.4 and an in-depth explanation of each stage follows in the next subsections.

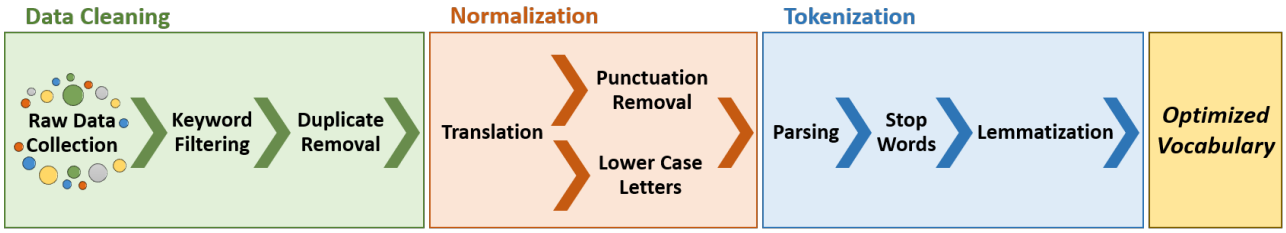

Figure 2.4: Data pre-processing pipeline.

Note that although the labelling process consisted of a parallel procedure that did not influence pre-processing, in practice, it was carried out right after translation, so as to avoid redundant work on excluded reports while remaining interpretable to the labeler.

2.3.1 Data Cleaning

In this first stage, the data was filtered based on the utility it could provide. For this reason, all documents which did not contain any ”Probable Cause” information were immediately removed. Moreover, after a keyword analysis from the ”Classification” segment, it presented clear that not all described occurrences should be included. The documents linked to the following keywords were also excluded:

- Security - Shot - Surface-to-air
- Security - Sabotage (bomb)

The reason behind the exclusion of terrorist related accidents is based on the principle that personnel performance under malicious external threats is not representative of their professional behaviour under conventional circumstances, and therefore should not be classified into the HFACS-ML framework.

After that, all duplicate samples, verified through the ”Narrative” segment, were also removed. At the end of this stage, the the total sample pool had been decreased from 3276 to 1674 occurrences.

2.3.2 Normalization

The second stage of pre-processing was designed to homogenise the texts before parsing. With this intent, all non-English documents were first identified and translated into English, using *googletrans*' Python library [24]. This tool was applied exactly three times to the entire data set, as a number of documents would pass unnoticed for some iterations. Next, all letters were lower-cased and punctuation was removed.

2.3.3 Tokenization

In the third stage of pre-processing, the text from each document was parsed (or tokenized), converting each word into a single entity (or token). For this step, we chose to apply alphabetic parsing and removed all digits. Although significant information may at times be derived from these characters, we found them not to provide any additional value regarding human factors, as the main relevant semantic meaning was often found in word descriptions. The same justification applies to punctuation removal.

After parsing, we considered the removal of stop-words. Since, as described in [22], it is not always possible to predict if this action will result beneficial or disadvantageous for a particular application, various empirical trials were performed to test both scenarios. The results confirmed that stop-word removal was indeed favorable for the current application. As such, these are described next.

Two lists of stop-words were used in the context of this study. The first list (Figure 2.5) was extracted from a publicly available source [25] and is commonly used in the treatment of natural English data. This list contains high frequency words, such as “a”, “the”, “of”, “and” or “an”, which in some cases corrupt the feature extraction process.

{ourselves, hers, between, yourself, but, again, there, about, once, during, out, very, having, with, they, own, an, be, some, for, do, its, yours, such, into, of, most, itself, other, off, is, s, am, or, who, as, from, him, each, the, themselves, until, below, are, we, these, your, his, through, don, nor, me, were, her, more, himself, this, down, should, our, their, while, above, both, up, to, ours, had, she, all, no, when, at, any, before, them, same, and, been, have, in, will, on, does, yourselves, then, that, because, what, over, why, so, can, did, not, now, under, he, you, herself, has, just, where, too, only, myself, which, those, i, after, few, whom, t, being, if, theirs, my, against, a, by, doing, it, how, further, was, here, than}

Figure 2.5: List of commonly used English stop-words (source [25]).

The second list, however, was tailored to the current data set and created to improve robustness. Frequently, ”Probable Cause” reports contain phrases that introduce the text, but provide no relevant human factor information. Two examples can be seen just from Figure 2.2, as ’Causes of the accident (translated from French)’ and ’The causes are related to:’. Since these phrases are often repeated, with some variations, and may appear in different parts of the text, an additional stop-word vocabulary was created (Figure 2.6) for mitigating the unwanted noise.

{summary, probable, cause, accident, contributing, factor, finding, findings, conclusion, conclusions, translate, translated, spanish, italian, french, german}

Figure 2.6: List of added stop-words.

In the final phase of this stage, words could undergo one out of two morphological processes: stemming and lemmatization. The first, stemming, trims inflected words into a root form, called a stem. For example, the words 'automatic,' 'automate,' and 'automation' would each be converted into the same stem, 'automat'. Stemming is known for efficiently providing word simplifications. The second process, lemmatization, simplifies words into their syntactic base form, the lemma. For example, the words 'plays,' 'played,' and 'playing' would each be converted into the same lemma, 'play'. However, while words "caring" and "cars" would be reduced to "car" in a stemming process, a lemmatizer would correctly reduce them to "care" and "car" respectively, as it leverages dictionary information to find the most probable base form. Hence, even though requiring more computing resources, the latter method has proved to provide more accurate results in various scenarios [26, 27]. For this reason, lemmatization has been chosen as the main inflection removal tool and implemented using the *nltk* library [25]. Additionally, after this process, words appearing 5 or less times throughout the corpora were ignored, as these would prove too rare to form meaningful patterns.

2.4 Discussion

For a high-level assessment of the pipeline, in this section we compare word clouds derived from the corpora of "Probable Cause" texts, before and after pre-processing. Note that word clouds are visualization tools that represent words (or expressions) in a size proportional to their frequency in the corpus.

Figures 2.7 and 2.8 display word clouds before and after pre-processing, respectively. From Figure 2.7, we may observe that the raw corpora is full of irregular forms, such as different case letters and even different languages. It may also be observed that the vocabulary is constituted mostly of generic terms, such as "PROBABLE CAUSE" or "the accident", which provide little insight into the latent causes of the incidents. In contrast, Figure 2.8 clearly displays a much more homogeneous vocabulary, with less noise and more descriptive information. In fact, with the removal of extremely rare words, along with the other data cleaning tools, we were able to reduce the complete vocabulary from 10.136 words in the raw corpora, to 6.855 words in the processed texts, hence, contributing significantly to the simplification of data, much relevant for the subsequent feature extraction process.

However, there is still little information in these word clouds regarding which terms are more representative of each class. Additionally, the variance of word sizes evident in Figure 2.8 demonstrates a substantial imbalance of word appearance throughout the corpora, with only a few words occupying most of the illustration. This justifies the use of feature extraction tools which, amongst other things, may take into account a word's context as an indicator for its importance in a particular document. This topic is further explored in Chapter 3.

One last remark concerning the topics explored in this section that is worth noting is the imbalanced nature of the labelled set. In Table 2.2 it may be observed that for some levels of the framework, specifically Unsafe Supervision and Precondition for Unsafe Act, the two most frequent categories account for more than 50% of the total labels from that level. Although we do not consider this a problem for the outlier categories, as these provide less importance to the study, it might prove an obstacle for the prediction of primary HFACS-ML categories such as "Planned Inap. Oper." and "Failed Known Prob.". We address the topic of label imbalance in greater depth in Chapter 4.

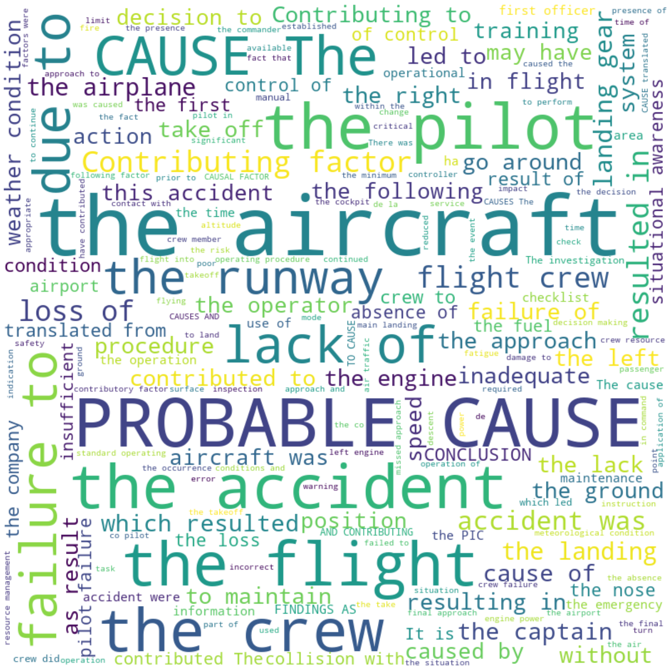

Figure 2.7: Word Cloud visualization of the corpora, before normalization.

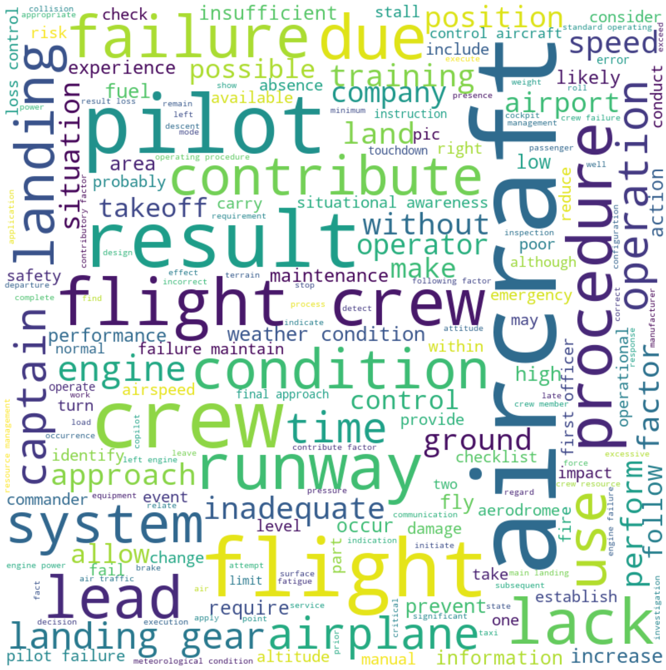

Figure 2.8: Word Cloud visualization of the corpora, after tokenization.

Chapter 3

Feature Extraction with Natural Language Processing

This chapter describes the feature extraction process carried out for this dissertation. Section 3.1 provides a brief introduction to the field of Natural Language Processing (NLP). After that, Section 3.2 expands on the TF-IDF, a classic method for representing document vectors, as well as its implications. Sections 3.3 and 3.4 develop on alternative approaches, which operate more complex embedding techniques. Then, Section 3.5 conducts a series of preliminary local experiments, that assess the various vector representations. Finally, Section 3.6 summarizes the obtained results and reflects on some considerations regarding both the algorithms and the classificatory framework.

3.1 Introduction to Natural Language Processing

3.1.1 Current applications

This field of research explores how computers can be used to understand and manipulate natural language text or speech. The potentialities of NLP are extensive and include tasks such as text classification, sentiment analysis, conversational agents, among many others. Figure 3.1 illustrates how some core tasks developed in this field are related to well known industry specific applications. For instance, a text classification algorithm can be developed as a ”spam detector”, which is able to identify scam accounts. Additional examples of NLP and its applications can be found in [28].

To serve the purpose of this study, and due to the nature of the data set, text classification tools will be considered in greater depth throughout the chapter.

3.1.2 Document Classification

In the context of NLP text classification, a wide range of applications can be derived. In this study, we examine the assignment of document classification, which consists of classifying unknown documents into categories, relative to their content, without the need for human intervention amid the categorization

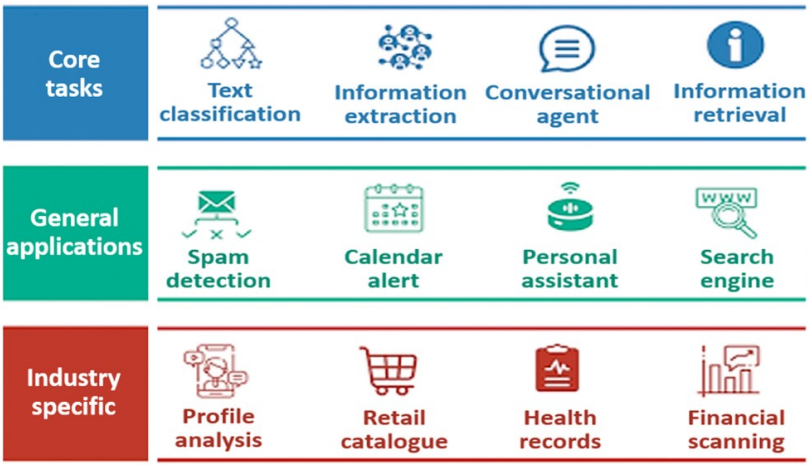

Figure 3.1: NLP Applications (adapted from [23]).

process. The automation of this task can potentially improve cost-efficiency, especially during revision of large corpora, an usually monotonous and time consuming assignment. However, this approach also comes with a compromise of control over the labelling process, as classification accuracy may vary substantially with data quality, taxonomic complexity and type of predictive model.

In a broad sense, document classification machine learning approaches can be grouped into three types, referred to as supervised learning, unsupervised learning, and semi-supervised learning. While supervised learning approaches take advantage of existent large amounts of up-front labelled data to train classifier algorithms that are able to produce regressions (or boundaries) within the projected data, unsupervised learning approaches directly use raw data to find patterns (or clusters) within the corpora, without the need for previously labelled reference samples. However, the first group does not take advantage of unlabelled data to improve predictions, making it a more expensive approach for raw environments, whereas the second group does not avail any small labelled set that might be available, and as such, solely relies on the data distribution. As to the third group, semi-supervised learning, it finds a balance between the previous two, using both labelled and unlabelled data to predict unknown documents. This approach is discussed in greater detail in Chapter 4.

For now, a schematic comparison between these three approaches can be found in Figure 3.2. Note that in this figure, red circles, green stars and blue crosses are considered labelled samples from different categories, while the blue circles and black rhombs are considered unlabelled samples. Additionally, the dashed lines are representative of the boundaries predicted by the respective algorithms.

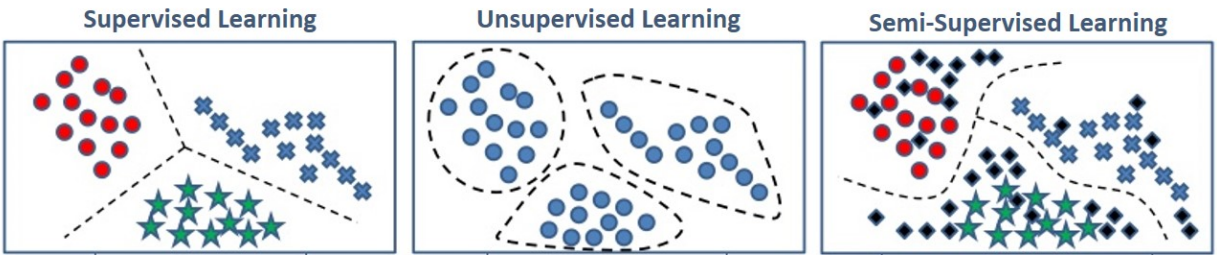

Figure 3.2: Types of classification methods (adapted from [29]).

3.1.3 Feature Extraction

In order to enable computers to read, decipher, and understand the semantic meaning of language data in a valuable manner, mathematical models are used to convert text segments into numerical vector projections. This process is referred to as feature extraction and the resulting algebraic format is often known as Vector Space Model (VSM). Although a multidimensional vector might not always have relevant meaning by itself, it can provide valuable insight when compared to other vectors, projected in the same feature space, with the same number of dimensions. A brief description of some feature extraction algorithms developed over the past decades follows bellow, whilst practical comparisons are exhibited later, in Section 3.5.

One simple and commonly used feature extraction technique is the Bag of Words (BoW). This method represents text under a collection (bag) of words, taking their frequency into account while ignoring the order and context in which they appear. The basic assumption behind BoW is that texts which possess similar words and word frequencies should belong to the same class (bag). However, this method assumes that all words have the same importance and does not consider that some rare terms might be potentially more descriptive of certain categories. For this reason, a variation from this method, TF-IDF, is often preferred and has been considered here as one of the main feature extraction techniques [30]. This algorithm is carefully described in Section 3.2 and has been computationally implemented with the aid of scikit-learn's Python library [31].

Although the interest of the previous methods, other more complex approaches have also been gaining traction over the past years, namely, the use of embeddings. As opposed to previous models, embeddings can create word (or document) projections of any custom length. Even though these representations may not be as human-interpretable, they seem to work well in practice and be able to capture text semantics in dense vectors [32]. Two examples of how a word embedding algorithm may project different terms in relation to each other can be found in Figure 3.3. In the left image, the state space suggests that a "woman" is to a "man", what a "queen" is to a "king", similarly to what a human interpreter might infer from a gender descriptive text. The same analogy applies to the image on the right.

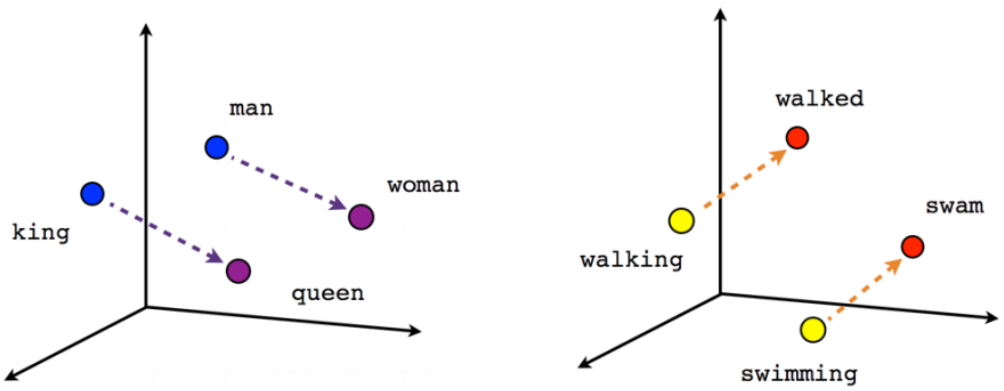

Figure 3.3: Word embedding examples: gender (left) and morphology (right) projections (source [33]).

In the scope of this study and with the goal of deriving document embeddings from the various probable cause records, two types of embedding techniques have been considered: transformed word embeddings and direct document embeddings. For practical use of the first, various operators such as *average*, *concatenate*, and *sum* could have been employed. However, we decided to use *average* to transform word embeddings into document embeddings because, as suggested in [34], it allows for stable vector length and spacial magnitude, facilitating feature tracking. Other reasons for this decision are less memory consumption and faster computational speed. A mathematical description of this operator follows.

Let v_w be the vector projection of word w, with custom dimensionality P, and d_i the vector projection of an arbitrary document i. Let also T_i be the set of ordered words found in that the same document, and N the total number of documents. Equation 3.1 shows the mathematical equivalent for the step of averaging word embeddings, to form document embeddings.

$$d_i = \frac{1}{|T_i|} \sum_{w \in T_i} v_w, \quad \text{for } i = 1, 2, ..., N. \tag{3.1}$$

An in-depth description of the transformed word embedding (W2V) and direct document embedding (D2V) algorithms, along with their efficiency improvements, can be found in Sections 3.3 and 3.4, respectively. Note that in these methods, each numerical vector is a function of both the algorithm and the text corpus and, as such, vectors should not be mixed across algorithms or corpora. Note also that both architectures have been implemented computationally with the aid of Gensim's Python library [35].

3.2 TF-IDF

In information retrieval, the Term Frequency–Inverse Document Frequency (TF-IDF) is a statistical algorithm that is intended to reflect how important a word is to a document, within a given collection of documents. Similarly to the BoW, a TF-IDF feature increases each time a certain word, representative of that feature, appears in the document. However, in this case, this factor is offset by the number of documents in which that word appears throughout the corpora. The latter concept, first conceived in [36], helps to adjust for the fact that, in general, some words appear more frequently than others and should therefore be considered with less importance. For instance, in our study, the word "flight" is so common that its term frequency could tend to incorrectly emphasize documents which happen to use that word more often, while other more domain-specific terms such as "miscommunication" and "weather" could pass unnoticed. Hence, an inverse document frequency factor has been incorporated to decrease the weight of terms that occur with more frequency in the document set, while increasing the weight of rare terms.

Formally, let $V = \{w_1, w_2, ..., w_V\}$ be the set of distinct words in the vocabulary, each feature $q_i(w)$ in a TF-IDF document vector d_i represents the weight word w possesses for that document. Additionally, let $f_i(w)$ be the frequency of the same word, in the same document, and $f_N(w)$ be the total number of documents in which that word appears. The formal weight computation is shown in (3.2).

$$q_i(w) = f_i(w).\log \frac{N}{f_N(w)} \tag{3.2}$$

Although the simplicity and proven efficiency of this algorithm, in this overview we also present some of its major limitations: the vector dimensionality increases with the size of the vocabulary, which may prove a computational challenge for large corpora; similarly to the BoW approach, it does not capture relationships between words or word orders; and it has problems in handling out of vocabulary words, impairing the classification of new documents [37]. For these reasons, the next sections describe alternative approaches, used in this research, that seek to solve some of the above-mentioned obstacles.

3.3 Word2Vec

This word embedding approach, introduced in [38], uses shallow neural networks to provide a method to establish a word's association with other words and the likelihood of their co-occurrence.

Neural networks consist of a set of algorithms modelled loosely after the human brain, designed to recognize underlying relationships between a given set of input and output data. They are often used to make predictions about unseen data, but other purposes can also be derived from these networks. Figure 3.4 briefly illustrates a high-level view of the typical deep learning process. A further comprehensive foundation can be found in [39].

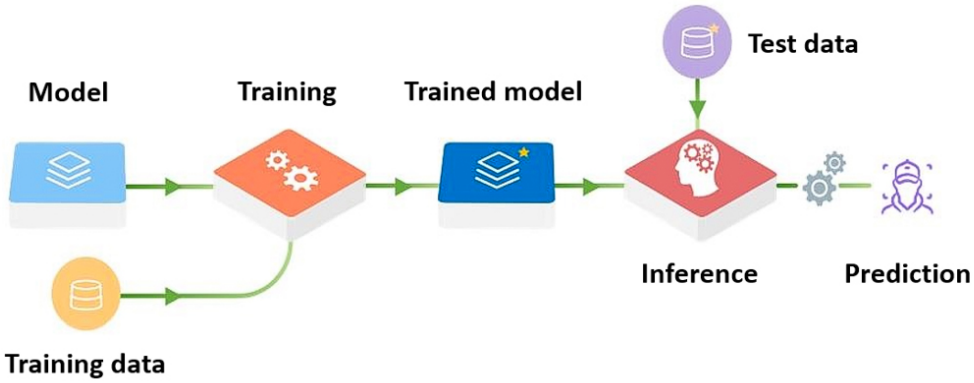

Figure 3.4: High-level view of a deep learning process (adapted from [40]).

In the Word2Vec (W2V) approach, neural networks can be used to derive word embeddings through two different types of architectures: the Skip-Gram (SG) model and the Continuous Bag of Words (CBoW) model. Since the performance of each architecture depends on a series of characteristics intrinsic to the corpus, and no clear indicator was found that could accurately predict which one might perform best against the current data set, both models have been considered in this analysis. Note that, for each model, the extracted word embeddings are posteriorly transformed into document embeddings, as described in equation 3.1.

3.3.1 Skip-Gram

The Skip-Gram model trains a shallow neural network to perform the task of predicting the surrounding context words $c \in C(w)$ of a single target word w, given a custom context size C and the sequence of words in the entire corpus T, by creating a set of context and target word pairs L. For exemplification purposes, Figure 3.5 illustrates how set L is derived from the text. Consider the blue box as the target word and the white boxes as the surrounding context words.

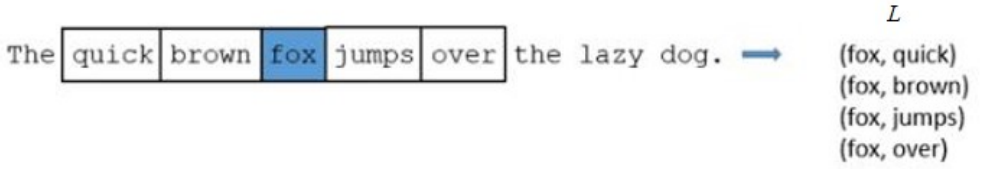

Figure 3.5: Word pair extraction example.

This set of word pairs is used to train and validate the accuracy of the model, with the context words being used as labels for prediction of a given target word. During training, the intrinsic parameters of the model, registered in matrices W and W', which reflect the hidden and output layer of the network, are updated so that the objective function of predicting the context word is maximized. Said function is presented in equation 3.3.

$$\frac{1}{|T|} \sum_{w \in T} \sum_{c \in C(w)} log\, p(c|w; W, W') \tag{3.3}$$

However, the true purpose of the Skip-Gram model is not actually to predict the context words, but to learn the parameters of W, a by-product of the training phase whose rows represent the embedded vectors of the vocabulary words. Additionally, let w_I and w_O be an arbitrary word pair from set L, representing the input and label words, respectively. Figure 3.6 illustrates a high-level view of the network, in which both these words, shown in the far left and far right of the image, respectively, are represented as one-hot encoded vectors of length $|V|$, with the non zero element corresponding to the position of the word in the vocabulary. Furthermore, let $v_{w_I} \in W$ be the embedded vector of the input word and the output of the hidden layer, and let the product of v_{w_I} with W' be used as input for the network prediction.

Knowing that the product of the output layer often does not result in a normalized probability distribution, one common approach for parameterizing the Skip-Gram model is to define the conditional probability from 3.3 as a softmax function that outputs a multinomial distribution, consisting of the prediction for each word in the vocabulary being the correct output word w_O. This conversion is shown in equation 3.4, and a more detailed demonstration for this step can be found in [42].

Note that in equation 3.4, $v'_w \in W'$ is the output weight vector associated with each word, located in the respective column of W'.

$$p(w_O|w_I) = \frac{exp(v'^T_{w_O} v_{w_I})}{\sum_{w \in V} exp(v'^T_w v_{w_I})} \tag{3.4}$$

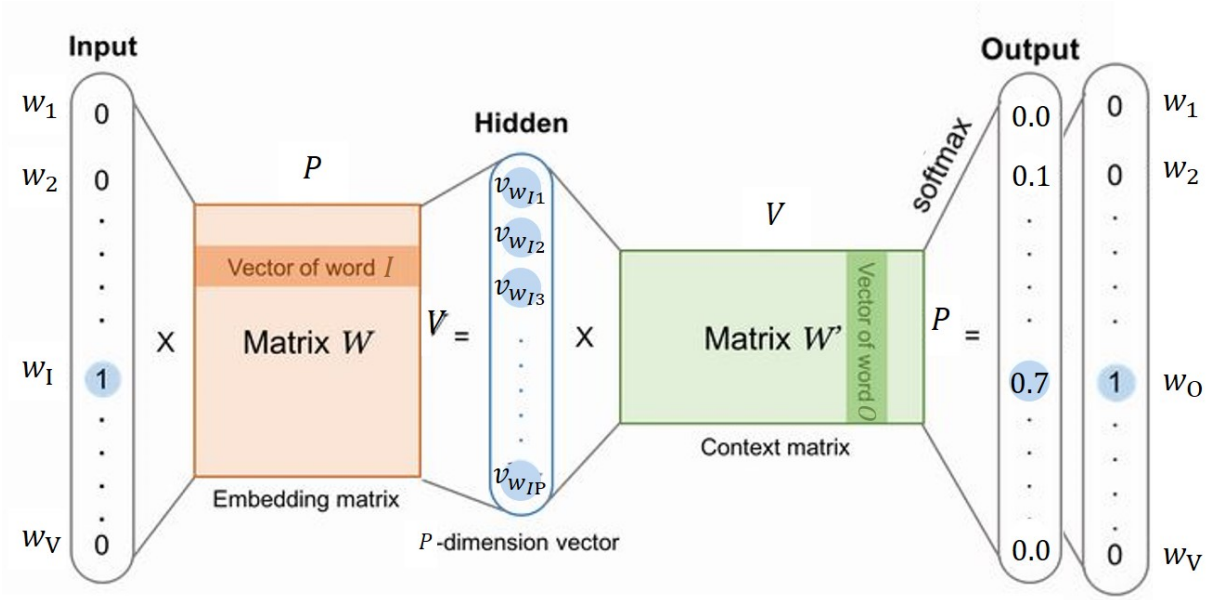

Figure 3.6: Structure and representation of the Skip-Gram model (adapted from [41]).

Weight Update

Let y be the softmax prediction output vector of length V and let t be the one-hot encoded label vector of the same length associated with the correct context word. For each training instance, an error vector e is created from the difference between these two. The computation of this variable is shown in equation 3.5. The error vector reflects the accuracy of the model and, as such, is used to update the inner parameters in the process of backpropagation, so that the conditional probability from equation 3.3 is improved for future predictions. Note that before training, the inner parameters have been set randomly.

$$e = y - t \tag{3.5}$$

Considering v_w and v'_w as two distinct representations of the word w, located in a row of W and column of W', respectively, the following analysis presents a summarized description of how these vectors are updated using stochastic gradient descent. See [43] for the complete in-depth demonstration. Additionally, v_{w_j} and v'_{w_j} will also be referred to as the ”input word vector” and ”output word vector” associated to word w_j.

For each training sample, given a custom learning rate $\eta > 0$, the update equation for the output word vectors is given by:

$$v'^{(new)}_{w_j} = v'^{(old)}_{w_j} - \eta . e_j . v_{w_I}, \quad \text{for } j = 1, 2, ..., V. \tag{3.6}$$

Note that this update equation implies that all V output word vectors of length P are modified in proportion to the size of the prediction error e_j and the learning rate η. Equation 3.6 also implies that when $y_j > t_j$ (“overestimating”), a fraction of the input word vector v_{w_I} is subtracted from v'_{w_j}, thus making v'_{w_j} farther away from v_{w_I}. On the other hand, when $y_j < t_j$ (“underestimating”), which is true only if $w_j = w_O$, a fraction of the input word vector v_{w_I} is added to v'_{w_j}, thus moving v'_{w_j} closer to v_{w_I}.

If y_j is very close to t_j, then according to the update equation, very little change will be made to the weights.

Now, let EH be a P-dimensional vector representing the sum of the output vectors of all words in the vocabulary, weighted by their prediction error. The computation for this vector is shown in equation 3.7.

$$EH = e.W' \tag{3.7}$$

The update equation for the input word vectors is then given by:

$$v_{w_I}^{(new)} = v_{w_I}^{(old)} - \eta.EH^T \tag{3.8}$$

Note that this update equation implies that only one row of W, corresponding to the embedding of the input word, is modified for each training instance. Intuitively, equation 3.8 can be understood as adding a fraction of every output word vector to the input word projection, with a size proportional to the prediction error. Therefore, if the probability of a word w_j being the output word is overestimated ($y_j > t_j$), then the input word vector v_{w_I} will tend to move farther away from the output word vector of v_{w_j}. Conversely, if the probability of w_j being the output word is underestimated ($y_j < t_j$), then the input word vector v_{w_I} will tend to move closer to the output word vector v_{w_j}. Finally, if the probability of w_j is fairly accurately predicted, then it will have little effect on the movement of the input word embedding.

Multi-Word Model

A small variation frequently applied to the SG model, used to increase computational efficiency, has also be applied in this research. This variation simultaneously trains all context words for each instance of center word, leading the network to have C multinomial distributions in the output layer, instead of just one. The new softmax function is given by:

$$p(w_{O,1}, ..., w_{O,C}|w_I; W, W') = \prod_{c=1}^{C} \frac{exp(v'^T_{w_{O,c}} v_{w_I})}{\sum_{w \in V} exp(v'^T_w v_{w_I})} \tag{3.9}$$

Where $w_{O,c}$ is the c-th context word of w_I and $v_{w_{O,c}}$ is the output vector representation of the same word.

The derivation of the update equations remains then identical to the single-word model, except for the prediction error e, which is replaced with EI, given by in equation 3.10.

$$EI = \sum_{c=1}^{C} e_c \tag{3.10}$$

Note that the intuitive understanding of this model remains the same, but the hidden layer vector is only updated once per center word, instead of for each word pair.

3.3.2 Continuous Bag of Words

Similar to the Skip-Gram model, the Continuous Bag of Words model (CBoW) trains a neural network on the task of word prediction, though, unlike the Skip-Gram model, it does not try to predict

the nearby word in a given sequence. Instead, it attempts to predict some center word w_O based on the context around the target label $\{w_{I,1}, ..., w_{I,C}\}$, given a context window of size C. Using the example from Figure 3.5, the training sample extracted from the presented window would define the words $\{quick, brown, jumps, over\}$ as input context words and the word $\{fox\}$ as the target word label. Let W and W' be once more the hidden and output layer weight matrices, respectively. For each training sample, the objective function that the network pretends to maximize is described in equation 3.11.

$$log\, p(w_O|w_{I,1}, ..., w_{I,C}; W, W') \tag{3.11}$$

Additionally, let the input and label vectors be represented as the one-hot encoded vectors x_i and t of length V. As opposed to the SG model, in the CBoW model, the output vector v_{w_I} of the hidden layer is given by the average of context word embeddings, as shown in equation 3.12.

$$\begin{aligned} v_{w_I} &= \frac{1}{C} W^T (x_1 + ... + x_C)^T \\ &= \frac{1}{C} (v_{w_I,1} + ... + v_{w_I,C})^T \end{aligned} \tag{3.12}$$

The objective function 3.11 is then defined through a softmax function that outputs the normalized probabilistic distribution, reflecting the respective prediction for each word. This conversion is shown in equation 3.13

$$p(w_O|W_{I,1}, ..., W_{I,C}) = \frac{exp(v'^T_{w_O} v_{w_I})}{\sum_{w \in V} exp(v'^T_w v_{w_I})} \tag{3.13}$$

The structure of this network and its vector representations can be found in Figure 3.7. Remember that y is the prediction output of the network, given in the form of a multinomial distribution vector of size V.

Weight Update

Let again v_{w_j} and v'_{w_j} be referred to as "input word vector" and "output word vector", respectively, both associated to word w_j. Noting the format similarity of the output layer to the one-word SG model, the error e computation between the network prediction and the label vector of the CBoW model can be given by the same equation 3.5. The update equation for the output weights using stochastic gradient descent is then given by:

$$v'^{(new)}_{w_j} = v'^{(old)}_{w_j} - \eta . e_j . v_{w_I}, \quad \text{for } j = 1, 2, ..., V. \tag{3.14}$$

Where v'_{w_j} is the output vector representation of word j and η is a positive custom learning rate. Additionally, the update equation for the hidden layer weights can be given by:

$$v^{(new)}_{w_{I,c}} = v^{(old)}_{w_{I,c}} - \frac{1}{C} \eta . EH^T, \quad \text{for } c = 1, 2, ..., C. \tag{3.15}$$

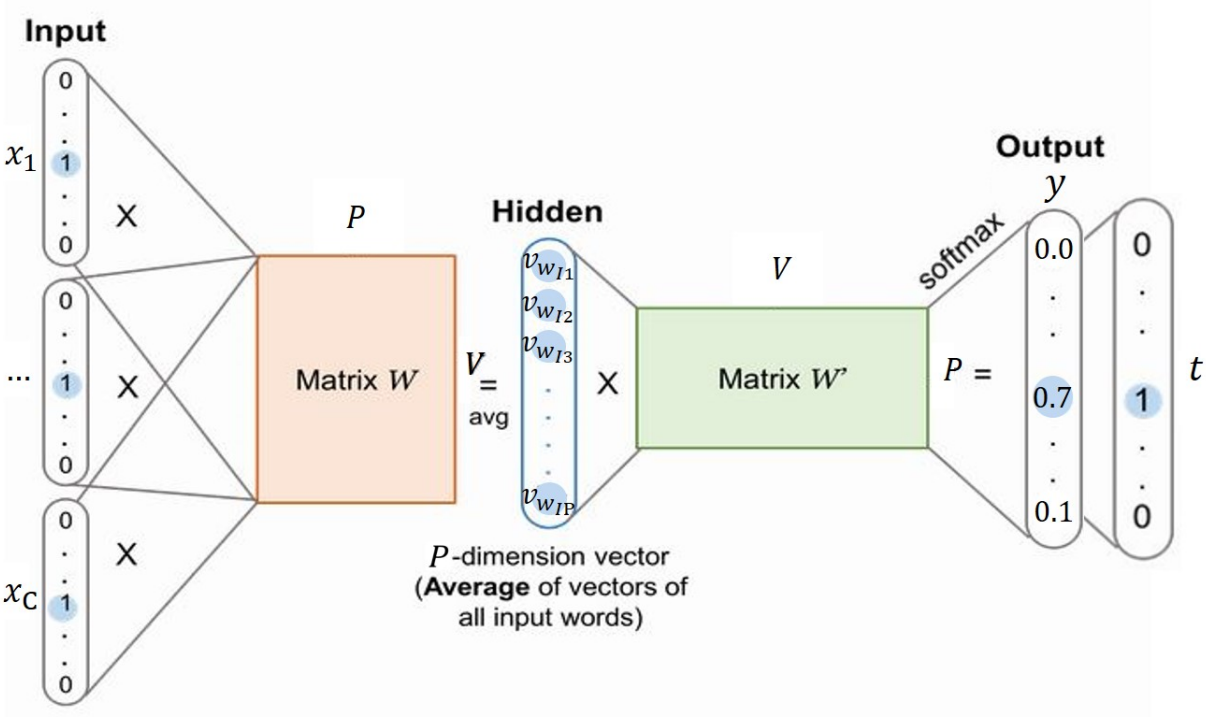

Figure 3.7: Structure and representation of the CBoW model (adapted from [41]).

Where $v_{w_{I,c}}$ is the input word vector for the c-th word in the input context and EH is the P-dimensional vector representing the sum of the output vectors of all words in the vocabulary, weighted by their prediction error. See [43] for the full demonstration of equations 3.14 and 3.15.

The intuitive understanding of the update equations (3.14, 3.15) is equivalent to that of the described for the SG model (in 3.6, 3.8). Note, however, that the extra term $1/C$ in equation 3.15 implies that the bigger the context window, the smoother the impact of the context words on the center word, albeit at a cost of computational complexity [38].

3.3.3 Optimizing Computational Efficiency

The previous sections describe the SG and CBoW models in their basic form. However, alone, these methods are highly inefficient. Learning the input vectors is relatively cheap, only P weights need to be updated for each training sample. However, learning the output vectors is very expensive, since all PxV weights have to be updated for each training instance. Softmax computation scales proportionally to the vocabulary size and accurate results for word-embeddings often require large corpora [43].

To solve this problem, two different commonly used approaches, which restrict the number of vectors to be updated per training instance, will be described in the next subsections. Note that, for simplicity, these applications have mainly been described for the one-word SG model, but can naturally be extended to the other architectures.

Negative Sampling (NS)

This approach, presented in [44], has provided a more efficient way of deriving word embeddings, by having each training sample only modify a small percentage of the weights, rather than all of them.

In this case, the output elements are not described by the large one-hot encoded vector of length V. Instead, they are formed by a single non-zero element and a small number of "negative" (zero output) words. By doing so, only these two-word groups will be corresponding for tuning the respective weights at each iteration. The underlying assumption for this process consists in trading simplicity over computational speed, as long as the desired embeddings retain high-quality representations. Consequently, given a custom number of Z negative words, the weight size of the output layer is reduced to $(Z+1)$xP, from VxP, with $Z << V$.

The negative sample words are chosen using a "unigram distribution", which bases the probability $p(w)$ of choosing a word based on its frequency $f(w)$ of appearance in the text. The calculation for this probability is shown in equation 3.16. Note that the exponent $3/4$ was deduced in [44], as it empirically showed to produce acceptable results across data sets.

$$p(w) = \frac{f(w)^{3/4}}{\sum_{w \in V} f(w)^{3/4}} \tag{3.16}$$

Negative sampling has proven not only to reduce computational costs, but also to improve the quality of the resulting embeddings, especially for frequent words and low dimensional vectors [44, 45].

Hierarchical Softmax (HS)

Another approach for increasing computational efficiency, described in [44], uses a binary Huffman tree representation to describe the output layer. Each leaf unit represents a word from the vocabulary, whereas each branch represents a path probability of travel between nodes. Hence, each leaf unit has a unique path (probability) that is used to estimate the prediction of any word w, as the sampled word pair of the input word. Furthermore, it can be proved that for a binary tree with $|V|$ leaves there are $|V|-1$ inner units [43].

Figure 3.8 highlights, as an example, the path from the root to an arbitrary w_2. Let $n(w,j)$ be the j-th node on the path from the root to word w and let $L(w)$ be the length of this path, so that $n(w,1) = root$ and $n(w, L(w)) = w$.

In the hierarchical softmax model, instead of correlating each output vector with a respective vocabulary word, the output vectors $v'_{n(w,j)}$ are correlated to the $|V|-1$ inner units of the tree. Additionally, this method uses the sigmoid function $\sigma(x) = 1/(1+\exp(-x))$, a binary variation of the softmax, to activate the probability distributions. The probability of moving in a certain direction at each inner unit n is shown in equation 3.17.

$$p(n, left) = \sigma(v'^{T}_{n}.v_{w_I}) \tag{3.17}$$

Which is determined by both the vector representation of the inner unit v'_n, and the hidden layer's

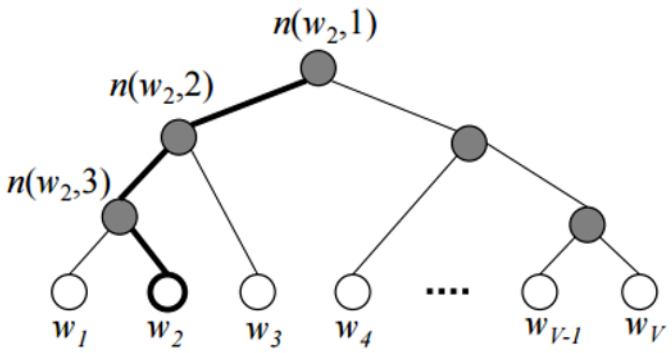

Figure 3.8: Example of a Huffman binary tree used for the hierarchical softmax model (source [43]).

output vector v_{w_I}. The probability of moving in the opposite direction at unit n is then computed through the negation rule:

$$p(n, right) = 1 - \sigma(v_n'^T . v_{w_I}) = \sigma(-v_n'^T . v_{w_I}) \tag{3.18}$$

The above equations exhibit that the prediction value for a certain output word w_O can be computed through the probability of a random walk, starting from the root and ending at the respective leaf unit. The computation for this value is shown in equation 3.19.

$$p(w = w_O|w_I) = \prod_{j=1}^{L(w)-1} \sigma([n(w, j+1) = ch(n(w, j))].v_{n(w,j)}'^T . v_{w_I}) \tag{3.19}$$

Where $ch(n)$ is the left child of unit n and [y] is a Boolean function that reflects the path of going left or right, defined as:

$$[y] = \begin{cases} 1, & \text{if } x \text{ is true.} \\ -1, & \text{otherwise.} \end{cases} \tag{3.20}$$

Now, let us analyse the update equations for the network weights. Let $t_{n(w_O,j)} = 1$ if $[y] = 1$, and $t_{n(w_O,j)} = 0$ otherwise. Equation 3.21 represents the weight update for the output node vectors $v'_{n(w,j)}$, where only the nodes leading up to the output word w_O (for $j = 1, 2, ..., L(w_O) - 1$) are modified. See [43] for the full demonstration.

$$v_{n(w_O,j)}'^{(new)} = v_{n(w_O,j)}'^{(old)} - \eta(\sigma(v_{n(w_O,j)}'^T . v_{w_I}) - t_{n(w_O,j)}).v_{w_I} \tag{3.21}$$

Here, the term $\sigma(v_{n(w,j)}'^T . v_{w_I}) - t_j$ can be understood as the prediction error for the inner unit $n(w, j)$. The "task" for each inner unit is then to predict whether it should follow the left child or the right child in the random walk, so that it finds the correct label word. In the same context, $t_{n(w_O,j)} = 1$ means that the ground truth is to follow the left child, $t_{n(w_O,j)} = 0$ means that the ground truth is to follow the right child, and $\sigma(v_{n(w,j)}'^T . v_{w_I})$ is the prediction result to follow the left child node. For a training instance, if the prediction of the inner unit is very close to the ground truth, then its vector representation $v'_{n(w,j)}$

will change only slightly. Otherwise, $v'_{n(w,j)}$ will move in the appropriate direction, either closer or farther away from v_{w_I}, so as to reduce the prediction error for this instance. Note that when used for the multi-word SG model, we need to repeat this update procedure for each of the C words in the output context.

During back-propagation, EH is computed again. Yet, this time as a P-dimensional vector, which represents the sum of vectors from path nodes that lead the root to w_O, weighted by their prediction error. The computation for this variable is shown in equation 3.22.

$$EH = \sum_{j=1}^{L(w_O)} (\sigma(v'^T_{n(w_O,j)}.v_{w_I}) - t_{n(w_O,j)}).v_{n(w_O,j)} \tag{3.22}$$

Since the format of the inner layer has remained untouched, the update equations for the input word vectors in both SG and CBoW models also remain the same, as shown in 3.8 and 3.12, respectively.

Although the number of parameters in the hierarchical softmax model remains roughly the same ($|V| - 1$ vectors for inner units compared to originally $|V|$ vectors for output words), the computational complexity per training instance, per context word, is significantly reduced, from $O(|V|)$ to $O(\log_2(|V|))$, as only path nodes need to be updated for each iteration.

3.4 Doc2Vec

First proposed in [32], this embedding architecture is a variation from Word2Vec that directly generates document embeddings during the neural network training. Doc2Vec (D2V) uses ”paragraph vectors” that act as memory devices, which contain the topic of the paragraph. Note that the name paragraph vector is to emphasize the fact that the method can be applied to variable-length pieces of texts, anything from a sentence to a large document. In our case, these will portray the information contained in the ”Probable Cause” reports and will often be referred to as document vectors. Moreover, while W2V works on the intuition that word representations should be good enough to predict surrounding context words, the underlying intuition of D2V is that the document representations may be good enough to predict the words or the context of that document.

Similarly to its correlative, this model can be trained through two different types of architectures. Both have been considered for this work and are, therefore, described next.

3.4.1 Distributed Memory for Paragraph Vectors (PV-DM)

This architecture possesses a keen resemblance to the CBoW model (described in Section 3.3.2), in the sense that it trains a shallow neural network to predict a target word, given a context set C. Besides mapping every word w to a unique vector, represented by a column in matrix W, the PV-DM architecture (Figure 3.9) has the benefit of also representing every document d by a column in matrix D, with W and D composing the weights of the hidden layer.

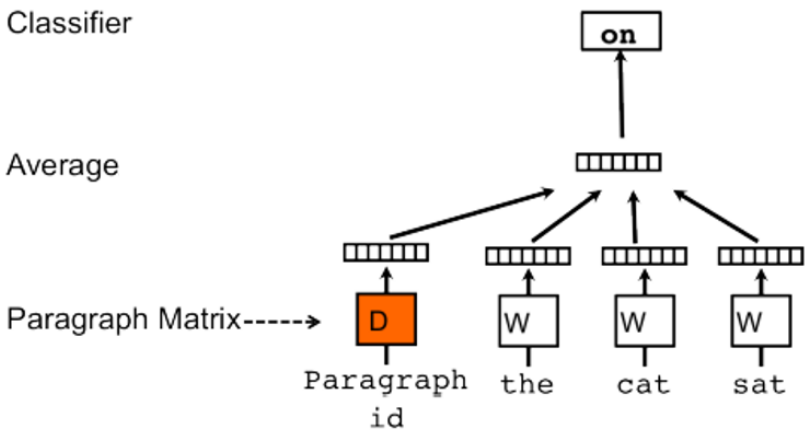

Figure 3.9: Representation of the PV-DM model's structure (adapted from [32]).

More formally, the main change in this model, compared to its precedent, regards the input word vector v_{w_I}, now constructed with both W and D. For exemplification purposes, let D be of size NxP, where P is the projection size and N is the number of documents. The computation of v_{w_I} is shown in equation 3.23. Subsequently, the respective weight update can also be computed analogously to the previously shown (3.15).

$$\begin{aligned} v_{w_I} &= \frac{1}{C}(W^T(x_1 + ... + x_C)^T + D^T(x_d)) \\ &= \frac{1}{C}(v_{w_I,1} + ... + v_{w_I,C} + v_d)^T \end{aligned} \tag{3.23}$$

Where $\{x_1, ..., x_C\}$ are the one-hot encoded vectors representing each context word and x_d is a unique hash-value associated to document d. Equation 3.23 also implies that each paragraph vector v_d is shared across all contexts generated from the same paragraph, but not across paragraphs. Note, however, that the word vector representations v_w are indeed shared between paragraphs.

At the end of the training, it is expected that word vectors and paragraph vectors may be capable of holding mathematical representations that accurately reflect the words in the corpus and the topics in the documents, respectively.

3.4.2 Distributed Bag of Words for Paragraph Vectors (PV-DBoW)

Another way to produce paragraph vectors is through forcing the model to predict words randomly sampled from the text. For each iteration of the stochastic gradient descent, the PV-DBoW samples an output text window, similar to the multi-word SG model, to form a classification task for the Paragraph Vector. Note that the input word vector v_{w_I} can be extracted analogously to the multi-word SG model, using D to contain the document embeddings.

Due to its structure (Figure 3.10), this model is only required to store the output layer weights (PxV) and document vectors (DxP), as opposed to the previous PV-DM, which also had to store the rest of word vectors, leading to an increase in computational speed and a decrease in memory consumption.

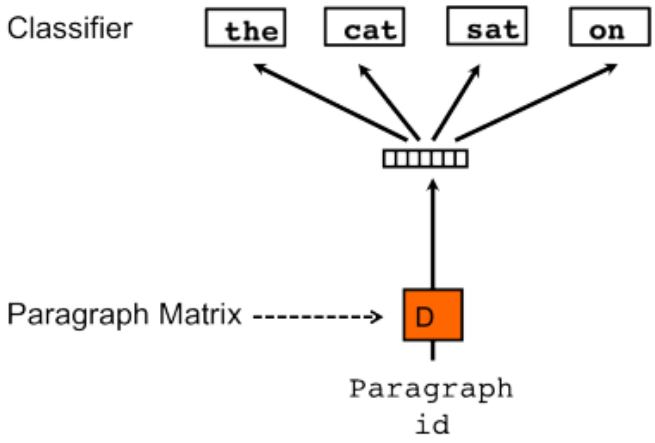

Figure 3.10: Representation of the PV-DBoW model's structure (source [32]).

Consequently, since the output layers of the above-mentioned architectures have remained untouched, regarding their W2V counterparts, both PV-DM and PV-DBoW models may operate with both hierarchical softmax and negative sampling classifiers.

3.5 Preliminary Analysis

One main challenge that comes with the analysis of the models described in this chapter is that, due to their multidimensional nature, the resulting document vectors are difficult to visualize and assess individually. However, since these vectors are plotted in an equally dimensional space, it is possible to compare them with each other. This section expands on a series of trials designed to achieve some preliminary conclusions regarding the quality of the resulting vectors, before proceeding to the classification stage.

3.5.1 Similarity Measure

In the current context, our interest lies in the measurement of vector similarity, as this is often one of the main indicators that regression algorithms use to define taxonomic boundaries. However, similarity measures may take various forms. In this preliminary analysis, we chose the commonly used cosine similarity for document comparison. It prioritizes vector orientation rather than magnitude, being therefore less affected by text length [46]. Note that for this particular case, text length was found not to be a relevant feature, as no meaningful correlation between text length and the main categories could be retrieved. The cosine similarity between two arbitrary document vectors, d_1 and d_2, is presented in (3.24).

$$S(d_1, d_2) = cos(d_1, d_2) = \frac{\vec{d_1}.\vec{d_2}}{||\vec{d_1}||.||\vec{d_2}||} \tag{3.24}$$

Where S may range from -1 to 1, with values closer to 1 representing a high degree of collinearity (or similarity) and values closer to 0 representing a high degree of orthogonality (or unrelatedness). Negative values may also outcome from this operation. Although not always presenting a clear interpretation,

they often refer to an opposite or, simply, to a different meaning, as in these cases vectors are placed far apart in the vector space.

Furthermore, since we aim to replicate this similarity measure information for any classification algorithm that may rely on Euclidean distance instead of cosine distance, all document vectors have been normalized to unit length, thus, allowing for vector orientation to be the main descriptor of each document. The normalization procedure is shown in (3.25).

$$\vec{d_i}' = \frac{\vec{d_i}}{||\vec{d_i}||} \tag{3.25}$$

Where $\vec{d_i}$' is the normalized document vector.

3.5.2 Word Similarity Test

Concerning the first set of quantitative trials developed in this research, we propose a method, inspired by [47], to compare the different feature extraction structures, regarding the quality of their produced word vectors. This is done by evaluating how different models perform in a word relatedness evaluation, measured by cosine similarity. It is expected that words which are often interpreted as being synonyms should possess a high evaluation score, while words which are often known as belonging to different contexts should be regarded with lower relatedness (similarity) values.

For this purpose, we trained, with the publicly available corpora, word vectors from two distinct arbitrary W2V models, CBoW HS and SG NS, and their respective D2V counterparts, DM HS and DBoW NS. For the latter model, an additional option from the Gensim integration was included in this trial set, which allowed for word vectors to be extracted in the DBoW training too [35]. The TF-IDF model could not be included in this set of trials because it does not generate single word representations. Moreover, all models from this section have been trained on their baseline versions, with no hyper-parameter tuning. A deeper analysis into the possible modifications of some of these architectures is conducted later, in Chapter 4.

Tables 3.1 and 3.2 retract the results from the various generated trials, exhibiting the cosine similarity scores of each tested model over the respective word pairs. In the first table, word pairs were formed from conventional synonyms, while in the second table, word pairs were formed from random words of different contexts and semantic meaning.

There is no clear consensus within the research community in regard to a fixed threshold for which to validate the accuracy of cosine similarity. This is mostly due to the different manners algorithms approach the task of vector space representation, which are not always comparable with each other. However, a rule of thumb to roughly estimate independent indicators is through averaging the similarity scores over all word pair combinations [48]. See equation 3.26 for the mathematical representation of this step. An overall score has also been added to Tables 3.1 and 3.2 to account for the number of word similarity values which performed either above or below the corresponding validation thresholds.

$$Th = \frac{S(w_1, w_1) + ... + S(w_1, w_V) + ... + S(w_V, w_1) + ... + S(w_V, w_V)}{V^2} \tag{3.26}$$

Table 3.1: Word similarity table composed of synonyms.

Word Pair		**Similarity**			
Word 1	**Word 2**	**W2V CBoW HS**	**W2V SG NS**	**D2V DM HS**	**D2V DBoW NS**
aircraft	airplane	0.708	0.729	0.785	0.678
pilot	captain	0.799	0.834	0.826	0.820
inadequate	poor	0.850	0.834	0.853	0.843
collision	impact	0.877	0.950	0.907	0.927
collision	crash	0.728	0.896	0.745	0.849
Threshold		> 0.65	> 0.83	> 0.66	> 0.80
Score		5/5	4/5	5/5	4/5

Table 3.2: Word similarity table composed of different context words.

Word Pair		**Similarity**			
Word 1	**Word 2**	**W2V CBoW HS**	**W2V SG NS**	**D2V DM HS**	**D2V DBoW NS**
aircraft	teamwork	0.254	0.812	0.311	0.771
pilot	fuel	0.131	0.391	0.144	0.278
inadequate	deer	0.150	0.615	0.077	0.601
collision	paragraph	0.049	0.319	0.088	0.232
collision	culture	-0.025	0.180	0.005	0.122
Threshold		< 0.65	< 0.83	< 0.66	< 0.80
Score		5/5	5/5	5/5	5/5

Although no model was trained with word morphology, from the final scores (Tables 3.1 and 3.2), we may observe that in some cases these unsupervised techniques are indeed able to capture semantic information, by placing words with similar meaning in close orientations and words from different contexts in near orthogonal planes. This is directly interpreted by the fact that most tested synonyms were evaluated with similarity scores above the average threshold, and all randomly sampled words taken from different contexts resulted below the same (threshold), although with a bit less consistency. In fact, it is worth mentionings that not all words should be expected to satisfy this trend, as these models still rely on stochastic processes and only reflect words and their context, based on uneven text data.

We may also observe from both tables that the models which use Negative Sampling as the main classifier tended to perform slightly worse than those employing Hierarchical Softmax. This discrepancy may be linked to a number of reasons. One might be that the vocabulary is mainly comprised by infrequent words, in which case, as noted in the Google study [45], Hierarchical Softmax tends to be a better choice for classifier. Another reason may be that Negative Sampling models require a higher hyper-parameter tuning effort, as opposed to the Hierarchical Softmax which does not contain any specific hyper-parameters. This topic is further explored in Chapter 4.

3.5.3 Document Similarity Test

In the second set of qualitative trials developed in this research, we assess how some of the described models performed at their primary purpose: to place documents of similar human factor categories close to each other and documents of distinct taxonomies in the distant space. We conducted this experiment by taking the reference document, with identification number 361 (shown in Chapter 2), and computing its 5 most similar and 3 less similar documents on different feature extraction architectures. Note that

this measurement is evaluated through cosine similarity too.

For this purpose, we availed the two best trained embeddings (W2V CBoW HS and D2V DM HS) from the previous word similarity test and added a TF-IDF training, on similar conditions, for which the resulting document vectors were also extracted. Tables 3.3, 3.4 and 3.5 illustrate how each feature extraction architecture performed on the document similarity task. In other words, for each level of the framework, we measured how many documents close to the reference report possess the same category as the original. Note that similarity is given here in descending order. Complementarily, Tables 3.6, 3.7 and 3.8 summarize for each architecture how many documents that are place farther from the reference report possess the same category as the original. For these cases, similarity is also exhibited in descending order.

Table 3.3: Most similar documents from the W2V CBoW HS model.

		HFACS-ML Level		
	iD	**Unsafe Supervision**	**Precondition**	**Unsafe Act**
Reference Doc	361	Inadequate Supervision	Personnel Factor	Error
Most Similar Docs (**W2V**)	771	Inadequate Supervision	Physical Env. 2	Error
	1191	Inadequate Supervision	Personnel Factor	Error
	1011	Inadequate Supervision	Personnel Factor	Error
	1411	Inadequate Supervision	Condition of Operator	Error
	857	Inadequate Supervision	Personnel Factor	Error
Score		5/5	3/5	5/5

Table 3.4: Most similar documents from the D2V DM HS model.

		HFACS-ML Level		
	iD	**Unsafe Supervision**	**Precondition**	**Unsafe Act**
Reference Doc	361	Inadequate Supervision	Personnel Factor	Error
Most Similar Docs (**D2V**)	771	Inadequate Supervision	Physical Env. 2	Error
	1191	Inadequate Supervision	Personnel Factor	Error
	1011	Inadequate Supervision	Personnel Factor	Error
	1411	Inadequate Supervision	Condition of Operator	Error
	857	Inadequate Supervision	Personnel Factor	Error
Score		5/5	3/5	5/5

Table 3.5: Most similar documents from the TF-IDF model.

		HFACS-ML Level		
	iD	**Unsafe Supervision**	**Precondition**	**Unsafe Act**
Reference Doc	361	Inadequate Supervision	Personnel Factor	Error
Most Similar Docs (**TF-IDF**)	162	Inadequate Supervision	Personnel Factor	Error
	889	Inadequate Supervision	Personnel Factor	Error
	981	n/a	Physical Env. 1	Error
	1353	Inadequate Supervision	Personnel Factor	und
	1234	Planned Inap. Oper.	Personnel Factor	Violation
Score		3/5	4/5	3/5

From the first table set (3.3, 3.4 and 3.5), some trends can be observed. First and foremost, although W2V and D2V architectures are known to generate different embeddings, as proved in Section 3.5.2, they have performed exactly the same in the document similarity task, by surrounding the reference document with the same closest reports in an identical order of proximity. This may be explained by their close algorithm affinity. Secondly, it may also be noted from the embedding results that, while the

Unsafe Supervision and Unsafe Act levels showed taxonomic consistency for the observed range, the Precondition level did not perform as well, with mixed categories being found in the close state space. Lastly, it may be noted from this table set, that the statistical TF-IDF model underperformed against the embedding models, by correctly identifying two documents very similar to the reference report, but failing at identifying other that could also belong to the same human factor categories. A reason for this behaviour could be related to this model's failure to capture word order and semantics [38]. While the TF-IDF approach allows for documents with very similar word frequencies to be correctly matched, documents which address the same topics, but do so through different vocabularies, may not always be recognized.

Table 3.6: Least similar documents, from the W2V CBoW HS model.

		HFACS-ML Level		
	iD	**Unsafe Supervision**	**Precondition**	**Unsafe Act**
Reference Doc	361	Inadequate Supervision	Personnel Factor	Error
Least Similar Docs (**W2V**)	212	n/a	Physical Env. 2	n/a
	626	n/a	Technological Env.	n/a
	823	n/a	Technological Env.	n/a
Score		0/3	0/3	0/3

Table 3.7: Least similar documents, from the D2V DM HS model.

		HFACS-ML Level		
	iD	**Unsafe Supervision**	**Precondition**	**Unsafe Act**
Reference Doc	361	Inadequate Supervision	Personnel Factor	Error
Least Similar Docs (**D2V**)	212	n/a	Physical Env. 2	n/a
	626	n/a	Technological Env.	n/a
	823	n/a	Technological Env.	n/a
Score		0/3	0/3	0/3

Table 3.8: Least similar documents, from the TF-IDF model.

		HFACS-ML Level		
	iD	**Unsafe Supervision**	**Precondition**	**Unsafe Act**
Reference Doc	361	Inadequate Supervision	Personnel Factor	Error
Least Similar Docs (**VSM**)	98	n/a	Technological Env.	n/a
	30	n/a	Condition of Operator	Error
	23	n/a	Technological Env.	n/a
Score		0/3	0/3	1/3

From the second table set (3.6, 3.7 and 3.8) the analysis is much more straightforward. Across the different architectures, documents located farther from the reference report revealed to generally possess little in common with the original, holding different human factor categories. In one case, however, an exception was found. In Table 3.4, Document 30, with the Unsafe Act level labelled as "Error", was placed very far from the reference document, and was still observed to hold the same category. This could be, again, a signal that the TF-IDF model is not as indicated for the current data set. Or, that the categories are not always distributed into broad context areas, but may instead be allocated to smaller topic islands, which portray different types of sub-contexts. To put it simple, it is possible that instead of a big group of "Error" labelled documents, there are various dispersed groups with the same label, each denoting a specific context in which it is used.

To assess this assumption, we implemented, for the current labelled set and using the previously trained D2V DM HS document embeddings, an effective tool for visualizing high-dimensional data. The t-Distributed Stochastic Neighbour Embedding (t-SNE) algorithm was applied for each level of the framework to project the multi-dimensional samples into a two-dimensional plane, while retaining most information regarding their state space distribution. The complete sample visualization is shown in Figure 3.11, and a simplified version, without legend and unlabelled samples is shown in Figure 3.12. Note that an in-depth explanation of this tool is out of the scope of this study, but can be found in [49]. Note also that the samples represented in these illustrations have gone through a transformation process and, therefore, do not exactly represent the real embedded vectors.

It is evident from Figures 3.11 and 3.12 that, as hypothesized earlier, labels are notably sparse throughout the vector space, in some cases forming smaller common groups instead of big wide clusters. However, these groups are not always evident, and are, at times, disrupted by conflicting categories. Although this factor may prove a drawback for regression algorithms, it also proposes a challenge to better understand how a rearrangement of features, or even categories, may be able to produce more coherent patterns that satisfy the objectives of human factor classification. Therefore, the study will proceed in this direction.

3.6 Preliminary Conclusions

In this chapter, we described various feature extraction techniques applicable to the current data set and carried out some tests to evaluate their effectiveness.

During a preliminary analysis, in Section 3.5, we presented some examples where the embedding techniques proved successful at recognizing literary semantics, by placing words with similar meaning close to each other and words with distinct meaning further away.

In addition, another set of tests was conducted to evaluate the different architectures on the document similarity task. In these trials, we observed that the implemented techniques managed to correctly place documents with at least one common human factor category close to each other. In the same trials, the document embedding structures presented almost identical results and outperformed the statistical TF-IDF. This outcome justifies the greater analysis, which takes place in the next chapter, regarding the potentialities of embeddings, with emphasis on D2V and its variations. We expect this model to perform better than the TF-IDF, while being slightly more efficient than W2V at no performance cost.

Lastly, some remarks may also be made regarding the classificatory framework. The results from Tables 3.3, 3.4, and 3.5 show that not all levels should be expected to behave the same, with a higher expectation being being placed on the Unsafe Supervision and Unsafe Act levels, and less coherence being expected from the Precondition for Unsafe Act level. Chapter 4 elaborates further on the comparison between levels, by evolving the local analysis described in this chapter to a global taxonomic evaluation.

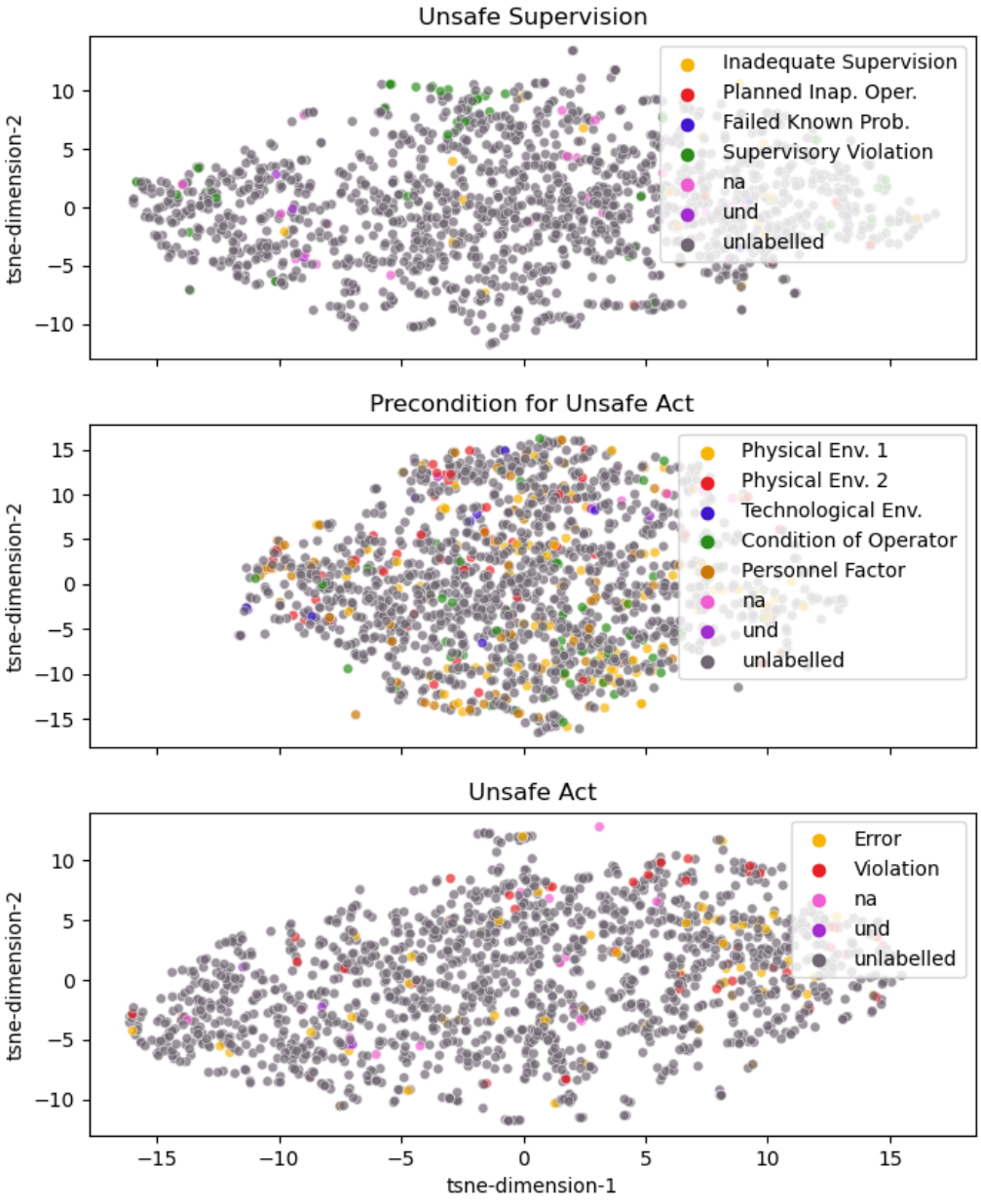

Figure 3.11: Low dimensionality document vector visualization of all samples.

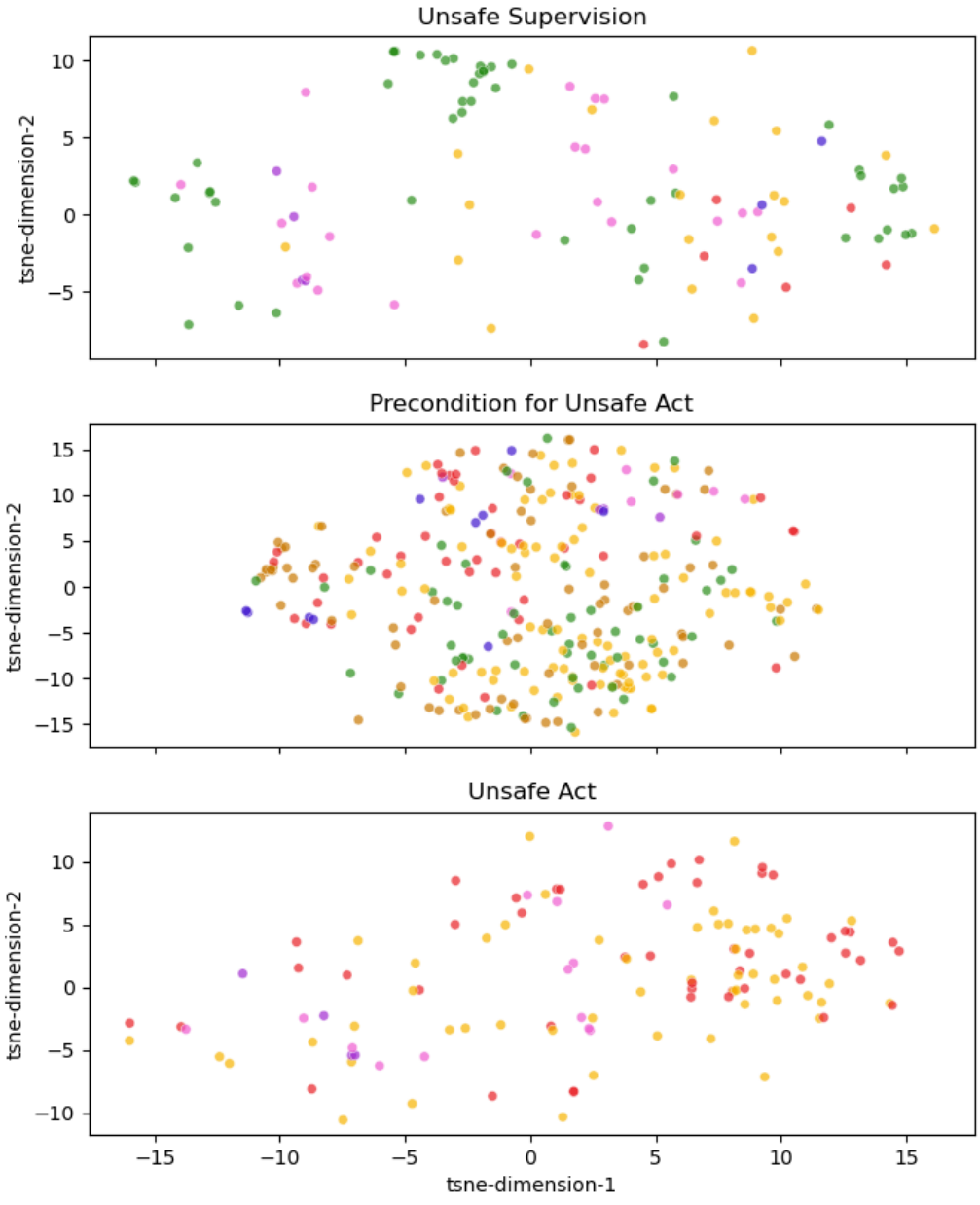

Figure 3.12: Low dimensionality document vector visualization of all labelled samples.

Chapter 4

Human Factor Label Propagation

In this chapter, we propose the use of semi-supervised label propagation techniques to explore the patterns created by the document embeddings described in Chapter 3 and categorize probable cause reports based on their human factors. Section 4.1 opens this chapter with a brief introduction to semi-supervised learning. After that, Section 4.2 expands on the Label Spreading algorithm and used performance measures. Sections 4.3 and 4.4 elaborate on the different stages of model improvement and present some practical observations. Finally, Section 4.5 summarizes the best obtained results from each approach and compares them to other widely used classifiers.

4.1 Semi-Supervised Learning Overview

In this study, we first considered the use of unsupervised learning techniques to group documents of identical categories and the use of previously generated labels to assess prevalence over the established clusters. Various types of methods (K-Means, Bisecting K-Means, Spectral Clustering) were tested. Although the interest of these methods, they revealed unsuitable, in that they did not enable to efficiently distinguish between the desired features appurtenant to each category.

This prompted us to pursue a different general approach. Since supervised learning techniques such as Deep Neural Networks were expected not to be an adequate fit for the current data set - as they often require much larger amounts of labelled data for effective training, validation and testing of each category -, we decided to delve further into the middle ground of semi-supervised learning.

During the last years, semi-supervised learning has emerged as an exciting new direction in machine learning research. It is closely related to profound issues of how to effectively infer from a small labelled set while leveraging properties of large unlabelled data - a challenge often found in real-world scenarios, where labelled data is expensive to acquire [50].

In this chapter, we avail the small labelled set (developed in Chapter 2), as well as the rest of unlabelled samples from the ASN database, to train a classification algorithm, through a specific type of transductive inference referred to as Label Propagation. Proposed in [51], Label Propagation methods are based on the intuition that labelled nodes can propagate their information throughout the state space

and be used to predict other co-existent unlabelled samples. These approaches are often used to detect community structures in large-scale networks, but have also been implemented in the text classification domain for tasks such as prediction of tweet polarity [52] and dialogue utterance inference [53].

The main differences between the various semi-supervised learning algorithms lie in their assumption of consistency. A principled approach to formalize this assumption is through designing a function which is sufficiently smooth with respect to the intrinsic structure of the data [54]. Label Spreading, a type of Label Propagation algorithm, follows this principle through the use of regularization, enabling patterns to be more broadly revealed, while improving robustness to noise.

4.2 Label Spreading

Introduced in [54], Label Spreading uses soft clamping, a technique that allows the algorithm to change the weight of true ground labelled data up to a certain degree (α), in order to increase its confidence over the overall distribution. One of the main advantages of this method is that it enables to use the additional unlabelled data to better capture the shape of the underlying node relations, making it a preferred choice for the current data set. Note that each node is referred here as a single data point, represented by a vector. Note also that the value of $\alpha = 0.2$ will remain a constant, so as to preserve coherence between models. A more detailed description of the algorithm is presented in the following subsection.

4.2.1 Algorithm Description

Given a document vector set $\{d_1, ..., d_L; d_{L+1}, ..., d_N\}$ and a label set $\{l_1, ..., l_L\}$ appurtenant to a single HFACS-ML level, the first L documents (or nodes) from the vector set hold labels for that level and the rest are unlabelled. The labelled nodes interact as seeds which spread their information through the network, following an affinity matrix based on node distance and distribution. During each iteration, each node receives the information from its neighbours, while retaining a part of its initial information. This retention can be regulated through parameter α. The information is spread symmetrically over the state space until convergence is reached, and the label of each unlabelled point is converted to the class which it has received most information during the iteration process.

To define the affinity matrix, two different kernels may be used. We chose the Gaussian Radial Basis Function (RBF), as it proved superior to the K-nearest-neighbours (knn) during empirical trials. Additionally, its Gaussian approach to measure affinity intuitively presented more similarities to the cluster effect which we want to achieve, by prioritizing sample proximity and distribution instead of nearest neighbour count. Note that the RBF kernel affinity criterion (equation 4.1) may be further tuned through hyper-parameter $Gamma$, responsible for the weight with which two nodes may influence each other. The larger the $Gamma$ is, the closer two samples must be to be affected. The impact of this hyper-parameter in the overall objective function is further discussed in Section 4.4, and additional documentation for the description and implementation of the Label Spreading algorithm can be found in [55].

$$K(d_2, d_2) = \exp(-Gamma * ||d_1 - d_2||^2) \tag{4.1}$$

4.2.2 Evaluation Metrics

Multi-class classification metrics compare predicted results to ground truth labels not used during the training process. In this study, we will establish one primary metric, on which the models will be optimized, and two other complementary metrics that will be used to gain deeper insight into the results. Note that the mentioned metrics are not exclusive to the Label Spreading algorithm and have been widely applied to evaluate supervised and semi-supervised machine learning models. To this end, the following metrics will be explored:

- Micro F1 score: This metric has been defined as the primary measure since it takes both precision and recall into account, while considering each sample with equal importance. This choice of using this score for the optimization process is justified by the goal of achieving a maximum number of correct and confident predictions, independently of class imbalance or rare categories. The particular F1 score of each class is given by the harmonic mean of precision and recall, as shown in equation 4.2.

  $$F1 = 2\frac{recall * precision}{recall + precision} \tag{4.2}$$

  Consequently, the total Micro F1 score of an arbitrary multi-class classification, with N categories can be expressed by equation 4.3.

  $$Micro\, F1 = F1_{class1+class2+...+classN} \tag{4.3}$$

- Macro F1 score: This metric has been defined as one of the complementary measures since it possesses the particularity of considering each class with equal importance. This is a key indicator for imbalanced scenarios as it is significantly affected by variations of minority classes. The formula for the Macro F1 score is shown in equation 4.4.

  $$Macro\, F1 = (F1_{class1} + F1_{class2} + ... + F1_{classN})/N \tag{4.4}$$

- Partial Precision: This metric has been defined as the other complementary measure since it specifically penalizes high false positive rates. In this partial approach, the True Positive (TP) rate is given by the sum of correct predictions of the main HFACS-ML categories, whereas the False Positive (FP) rate is given by the sum of all samples wrongly attributed to the main HFACS-ML categories, among themselves and from true outliers. This score prioritizes the detection of outliers being attributed to the main categories, which jeopardizes the validity of the primary pseudo-labels, over more documents being classified as outliers, which mainly reduces the number of usable samples. The respective formula is given by equation 4.5.

$$Precision = \frac{\sum_{a \in A} TP_a}{\sum_{a \in A} (TP_a + FP_a)} \tag{4.5}$$

Where A is, in this context, any set of primary categories from a single level of the framework. Visually, this corresponds to an omission of the two latest columns from the respective confusion matrix, related to the labels predicted with classes ”not available” (n/a) and ”undetermined” (und).

The possible values for the previous metrics range from 0, representing very bad predictions, to 1 representing very good predictions. Complementary documentation regarding some of the used terms can be found in [56].

4.2.3 Data Split

In order to effectively use the labelled data for training and testing the classification models, for each level of the framework, the labelled set has been stochastically split into two subsets. This split was executed in a stratified manner, with a train and a test set encompassing a proportional number of examples of each class, with the restriction of always retaining at least one labelled sample of each category. However, while supervised machine learning algorithms are known to often perform better for higher train sizes, that might not always be the case for the Label Spreading algorithm, as remarked in [54], due to its smoothening properties, and therefore cannot always be optimally estimated. For this reason, although this variable is not usually considered a hyper-parameter, as it is not a direct modifier of the internal kernel, we will still analyse it over a range of values, in an effort to infer its influence. This analysis is presented in the following Section 4.3.

An additional remark regarding the train test data split that is worth to mention is that despite the fact that the test samples are stripped of their labels during the training phase, the correspondent vectors remain present in the state space, unlabelled, thus contributing to the regularization and label propagation process. The predicted outcome of the test nodes is then compared to their true class and used to retrieve the performance of the model.

4.3 Early findings

Having defined the propagation algorithm and the main evaluation metrics, in this section, we performed an initial analysis, using one of our baseline embeddings. The objective was threefold: to better understand how the baseline algorithm performs at distinguishing between categories; which levels of the framework present most reliable results; and if any kind of refining treatment might still be needed. For this task, we chose to use the previously trained D2V DBoW NS model.

For an efficient analysis of our baseline D2V model combined with the LS classifier, without yet knowing which value to use for the Train size, we relaxed this variable as a free hyper-parameter and plotted our primary metric, the Micro F1 score, accordingly, over the different HFACS-ML levels. All remaining relevant hyper-parameters were left to their default values (Table 4.1), following the suggested

from the respective original research or complementary documentation [23, 38, 44, 54, 55]. The results of these tests are displayed in Figure 4.1.

Table 4.1: Default value of each hyper-parameter.

Algorithm	Hyper-Parameter	Default Value
LS	Train size	Ts
	Gamma	20
D2V	Dimensions	100
	Window size	5
	Epochs	10
	Learning rate	0.025
NS	NS words	5

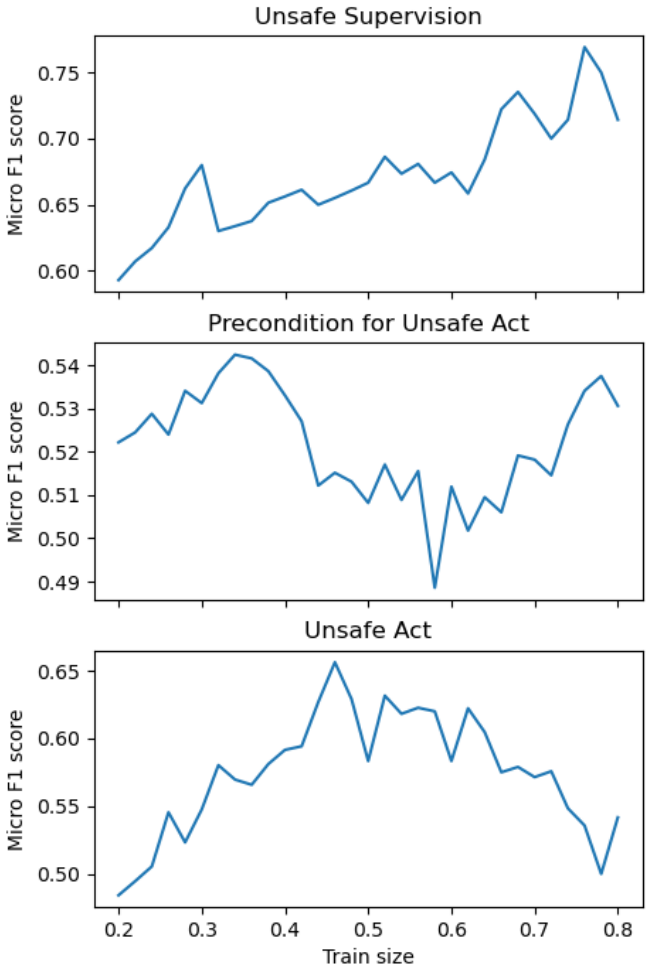

Figure 4.1: Evolution of Micro F1 score over a variable Train size, for each HFACS-ML level.

From the plots presented in Figure 4.1 it is possible to conclude that no single value of Train size is able to optimally suit all three data sets. While the upward trend in the first graph might suggest that the algorithm performs better on a larger labelled set, the downward trend observed in the third graph contradicts this assumption. On the one hand, it could mean that each level presents its own individual behaviour; on the other hand, it could also mean that the parameter is associated with high volatility, being also dependent on the other hyper-parameters. A deeper analysis into the various hyper-parameter associations is explored later, in Section 4.4.

Even though Figure 4.1 describes how the classification algorithm has generally performed at each level, it does not reveal specific information regarding individual categories. For this reason, Figures 4.2, 4.3 and 4.5 further complement the analysis, by reporting the concrete predictions for each category, through the respective confusion matrices, drawn at the best Train size instance of each level.

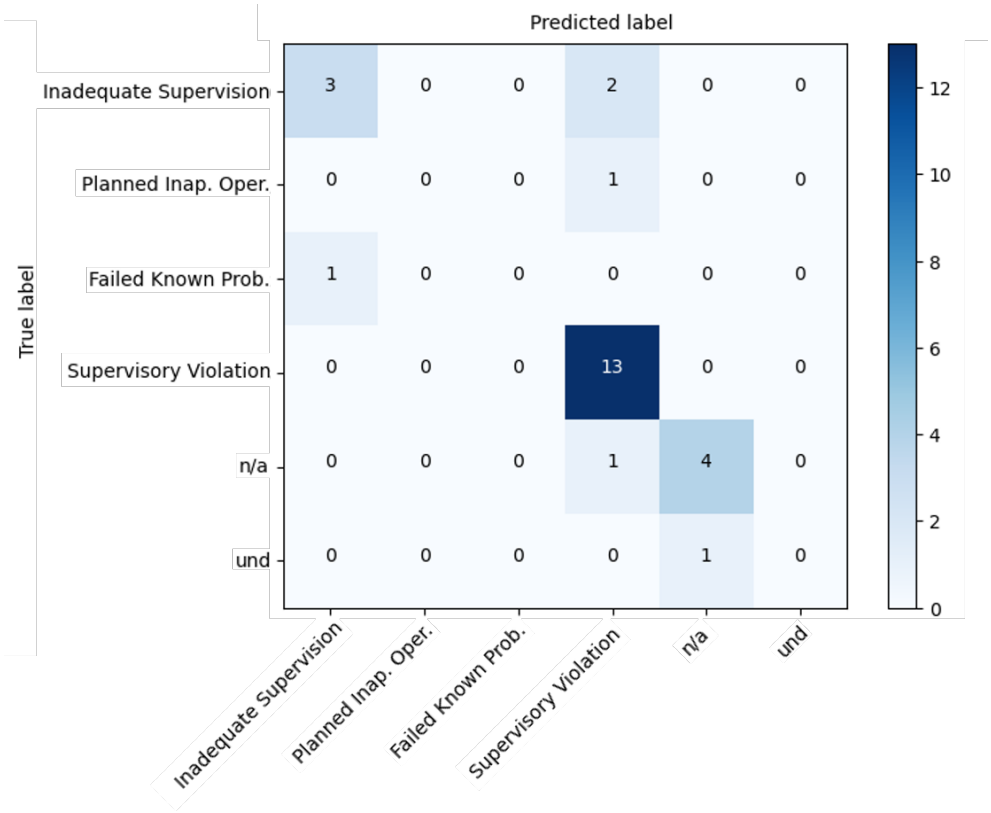

Figure 4.2: Unsafe Supervision confusion matrix, at $Ts = 0.76$.

Although possessing the highest Micro F1 score, Figure 4.2 shows that the Unsafe Supervision level is largely affected by class imbalance, being quite good at classifying the most common category (”Supervisory Violation”), but not as good at classifying the remaining categories. In this case, the shortfall of labels from the minor categories is also shown to be a significant drawback. Meaning that, either train labels are not sufficient to propagate over other nodes or that test labels are not sufficient to accurately portray the accuracy of these results. Undoubtedly, a larger labelled set would be required to provide more solid conclusions about the results of this level. However, it will still be kept in the analysis for comparison purposes.

Similarly, Figure 4.3 shows that the Precondition of Unsafe Act level also faces the same problems as the previous level, with an even clearer superposition of the most frequent class (”Physical Env. 1”) propagating over the less frequent ones, as exhibited in the first column of the matrix. However, in this level, most other classes still possess a significant amount of labels, whose influence on the state space might be significantly obstructed by the first class. To verify if this is indeed the case, we down-sampled the labels from ”Physical Env. 1” to nearly half the original amount, reaching an order of magnitude more similar to the other categories. The corresponding results are shown in Figure 4.4.

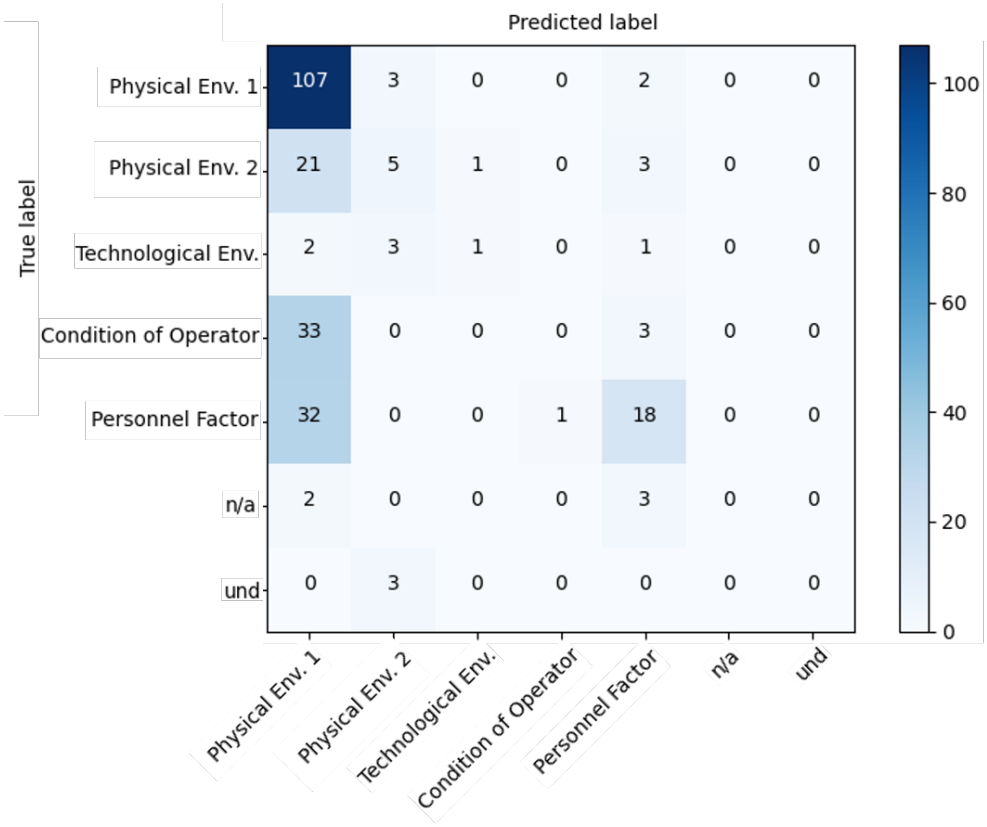

Figure 4.3: Precondition for Unsafe Act confusion matrix, at $Ts = 0.36$.

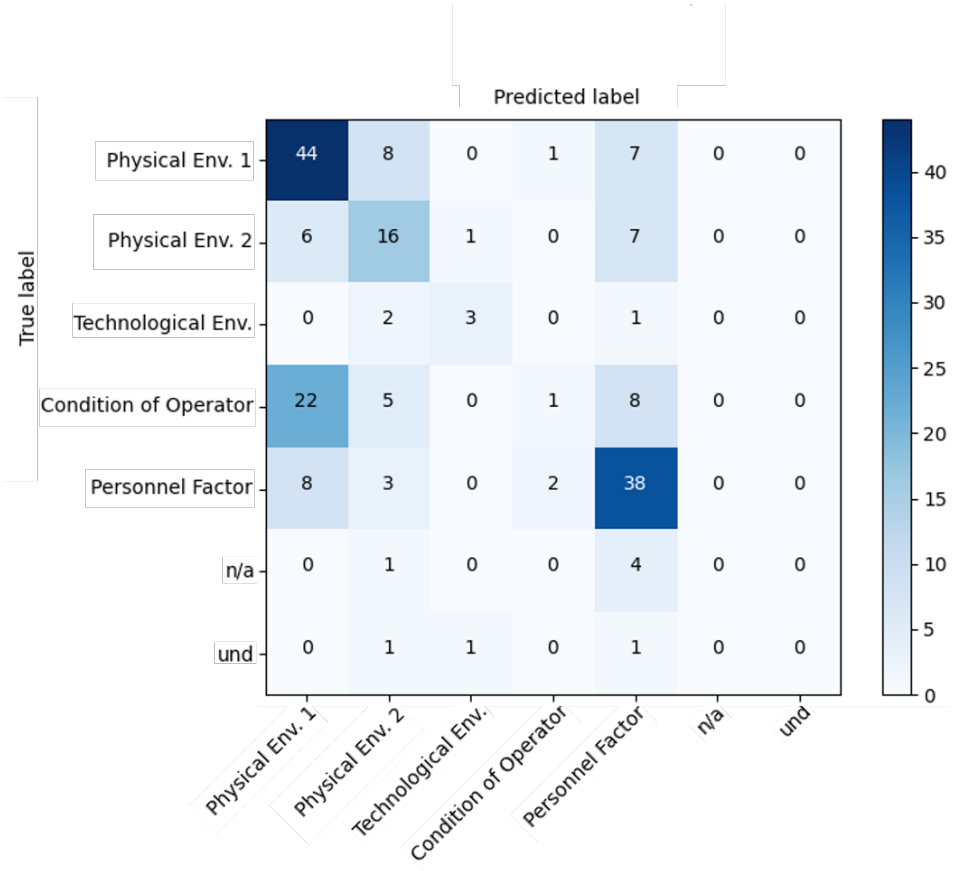

Figure 4.4: Precondition for Unsafe Act confusion matrix, with down-sampled ”Physical Env. 1”.

It is interesting to notice that such change clearly confirms that a significant improvement in class balance and prediction evenness has been accomplished. Although the Micro F1 score from this result remains roughly the same, around 0.54, there is an improvement of the Macro F1 score, which increases from 0.25 to 0.34. Unfortunately, this outcome still presents flawed predictions, especially over the ”Personnel Factor” category. It is possible that this irregularity may derive not from the label propagation or document embedding algorithms, but from the lack of clear distinctions between the text-based categories, since in many cases, probable cause reports happened to describe complementary preconditions for unsafe acts over the same incident, instead of clear distinct ones. Although clear guidelines were defined during manual categorization to deal exactly with these scenarios, there is a possibility that these rules are not mathematically evident if there are not enough labels available to propagate with these characteristics. Nevertheless, we still find worthwhile to explore up to what extent these patterns may be recognized. Therefore, this level of the framework will also continue to be studied.

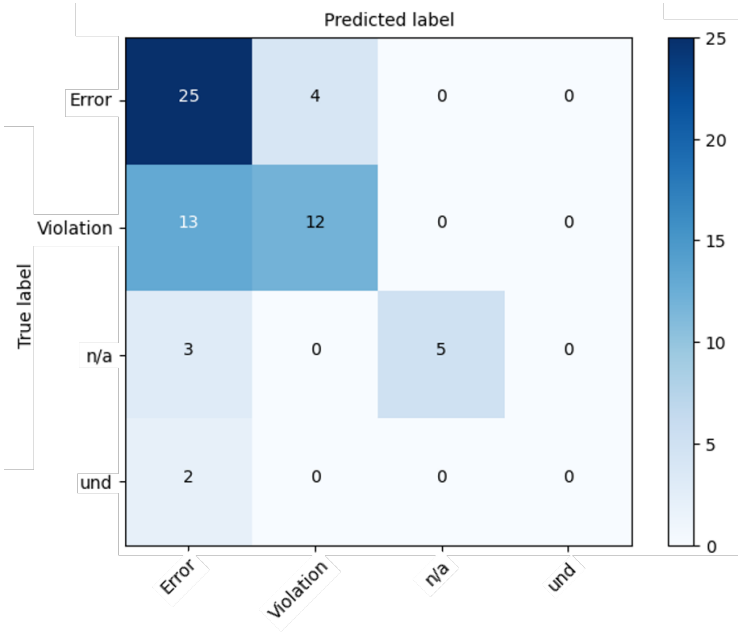

Figure 4.5: Confusion Matrix at $Ts = 0.46$.

Lastly, we may observe from Figure 4.5 that the classification system does not distinguish clearly between the two main categories from the Unsafe Act level (”Error” and ”Violation”), even though it is the most balanced level of the three. This exhibits a clear need for tuning both feature extraction and classification algorithms, so as to improve the developed models and understand the extent of the utility of this approach for the current data set. A final takeaway should also be noted from the confusion matrices shown in this section, regarding the outlier categories. While the ”n/a” class successfully identified most of its counterparts, providing therefore a valuable contribution to outlier identification, the ”und” class seemed to fail at the same task for all tested levels of the framework and only contributed to add noise to the predictor. In view of the fact that the latter category proved to be unsuited, this category as well as its associated documents have been removed from the study and will not be considered in the subsequent optimization process. Note that due to its low label count (shown in Section 2.2), the removal of this category will neither meaningfully impact the total sample size, nor the evaluation scores.

4.4 Model Optimization

As previously mentioned, the developed models rely on a number of hyper-parameters whose optimal values are still unknown. In this section, we describe a two-stage optimization approach used to extract the best performance from each of our four embedded structures, on all three HFACS-ML levels. In the first stage, a continuation of the previous D2V DBoW NS is used on the Unsafe Act classification task, with the purpose of demonstrating how the D2V and Label Spreading hyper-parameters interact with each other. In the second stage, we use this information to efficiently tune an optimizer which finds near-optimal hyper-parameter configurations for our models. A deeper explanation follows below.

There exist a variety of industry-standard optimization approaches, namely grid search and random search [57], which provide easy to implement hyper-parameter tuning solutions. However, these approaches are often based on rules of thumb or brute-force computing, and do not seek to gain a comprehensive understanding of the underlying parameters and their interactions, leading to non-optimal and computationally expensive results. In this work, we consider the automatic Bayesian optimization method, proposed in [58], which uses Gaussian Processes (GP) to generalize a predicted performance for each model state. Note that, in this context, state refers to a hyper-parameter configuration, associated with an objective function value, here measured by the Micro F1 score. This approach falls into a class of optimization algorithms called Sequential Model-Based Optimization (SMBO), which uses previous objective function observations to determine the next optimal hyper-parameter combination to be sampled, while seeking to maximize the Expected Improvement (EI). The internal metric EI also balances exploitation versus exploration of the search space, for either sampling points which are expected to provide a higher score or regions which haven't been explored yet. A comprehensive description of the algorithm and its application in machine learning can be found in [59].

Even though Bayesian optimization is considered a powerful method, as in any optimization problem, its search space complexity grows exponentially with the number of hyper-parameters. Not surprisingly, preliminary tests on the current data set have confirmed that the efficiency of this procedure indeed deteriorates with higher counts of free variables. For this reason, a primary analysis, inspired in [60], has been developed, in order to narrow down which hyper-parameters, from those exhibited in Table 4.1, account for a more significant influence on the objective function, and have therefore a higher need for tuning.

4.4.1 Hyper-Parameter Impact Analysis

Algorithm designers often assess hyper-parameter importance through manual investigation of local neighbourhoods, by varying one hyper-parameter at a time and measuring how performance changes. However, the only information obtained from this analysis is how different hyper-parameter values perform in the context of a single instantiation of the other hyper-parameters. Following [60], the relative importance of our hyper-parameters is evaluated here in the context of all instantiations. The used approach applies random forests to efficiently predicted marginal performance of individual hyper-parameters based on empirical data - consisting of performance scores and their respective hyper-

parameter configurations - and performs a functional Analysis of Variance (functional ANOVA) [61] to analyze how much of the performance variance in the configuration space is explained by each hyper-parameter. Complementary documentation on the implementation of this method can be found in [62].

To build the empirical data set, we ran a random search with 350 different states, registering for each state the performance score (Micro F1) and the respective hyper-parameter configuration. For this assignment, a grid of 20 values per hyper-parameter was defined, within reasonable ranges, both above and below the original default values (Table 4.1). A log-uniform scale was also adopted for most of the variables to uniformly evaluate different orders of magnitude, as hyper-parameters are usually expected to be less sensitive to small changes at higher ranges. The complete set of characteristics attributed to the hyper-parameters for the random search is shown in Table 4.2. A single run of the 350 random search states took around 43 minutes to complete. From these numbers, we conclude that it would be unfeasible to explore the same grid of 20^7 states through a complete grid search, using the same computing power.

Table 4.2: Random search characteristics of each hyper-parameter.

Hyper-Parameter	Min Value	Max Value	Scale	Type
Train size	0.2	0.8	Uniform	Float
Gamma	0.2	200	Log Uniform	Float
Dimensions	10	1000	Log Uniform	Integer
Window size	1	50	Log Uniform	Integer
Epochs	1	100	Log Uniform	Integer
Learning rate	0.0025	0.25	Log Uniform	Float
NS words	1	50	Log Uniform	Integer

Having fit the random search results into the functional ANOVA framework, we obtain the marginal contribution of each hyper-parameter, that is, the relative importance of each observed variable over the final Micro F1 score (Figure 4.6). Remember that, in this context, marginal contribution is given as a relative measure of importance and therefore should be primarily used for comparison purposes.

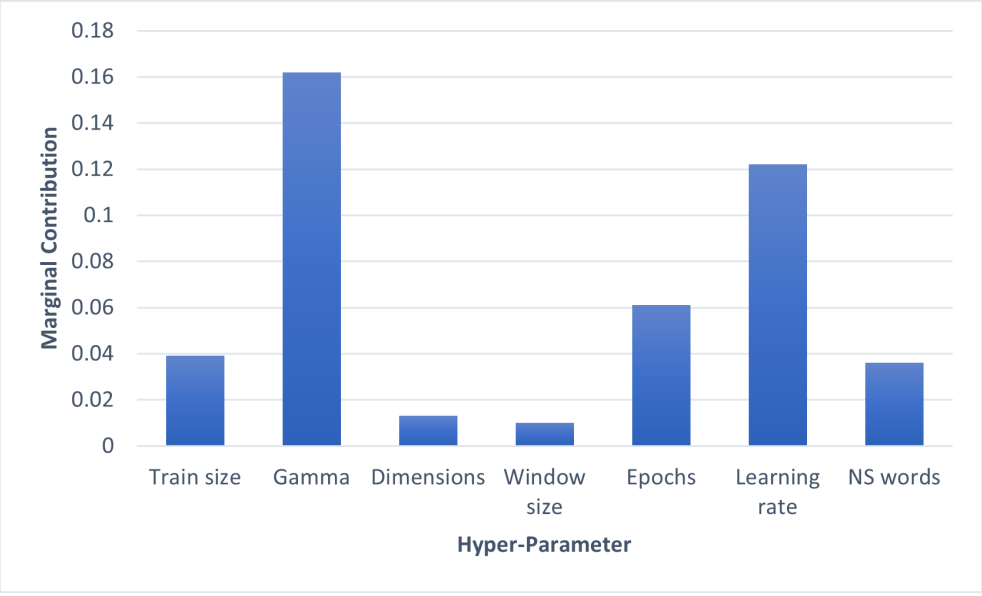

Figure 4.6: Marginal contribution of each hyper-parameter, predicted by the functional ANOVA framework.

From the bar plot exhibited in Figure 4.6 it may be observed that even in high-dimensional cases, most performance variations are attributable to just a few hyper-parameters - in this case Gamma and Learning rate - while other hyper-parameters such as Dimensions and Window size seem to possess a much lower influence.

Even though these results explicitly point out which variables hold a higher need for tuning, it is still unclear which intervals are acceptable, and whether or not the default values are a good fit for our data set. In order to gain a deeper insight into how each parameter might be expected to impact performance along its range, the individual marginal plots are also presented (Figures 4.7 and 4.8). Note that each plot shows the approximate marginal performance achieved when varying the respective hyper-parameter across the x-axis, with the blue line and red area indicating the predicted average and standard deviation, respectively. In other words, these plots illustrate, for each hyper-parameter, how the objective function is predicted to behave at each instantiation of that hyper-parameter, while taking into account all instantiations of the rest of hyper-parameters.

From the individual marginal plots, some assumptions may be inferred. It is possible to see that hyper-parameters Gamma, Epochs, Learning rate and NS words seem to perform worse at higher values. This may be a consequence of over-fitting, as the latter three, appurtenant to the D2V algorithm, are directly related to the training volume, which should not be required in great proportions since the data set is not exceptionally large. Coherently, the hyper-parameter Gamma, appurtenant to LS's RBF kernel, which is responsible for the fit of the decision region, is known to produce strict islands of decision-boundaries when tuned at high values, instead of broad and smoother curves for the lower values, therefore also presenting over-fitting behaviour at higher ranges. On the other hand, hyper-parameters Train size, Dimensions and Window size do not reveal clear behavioural patterns, as their average predicted performances do not vary substantially, and their respective standard deviations demonstrate elevated levels of volatility.

In sum, while the box plot from Figure 4.6 illustrates which variables objectively carry the biggest relative influence - Gamma, Learning rate, Epochs and Train size (in descending order) -, the individual marginal plots from Figures 4.7 and 4.8 suggest what range of values may be acceptable for the parameters which remain fixed. However, fANOVA's documentation clarifies that the framework has been primarily designed for minimizing hyper-parameter search and not to provide a flawless parameter configurator. For this reason, and due to the fact that the default values (Tables 4.1) are already quite close to the extremes observed in the marginal contribution plots (Figures 4.7 and 4.8), no modifications will be made to these values. As for the Train size, whose behavioural analysis did not result completely conclusive, we will initiate it to 0.8, following the commonly used $80/20$ supervised machine learning train test ratio.

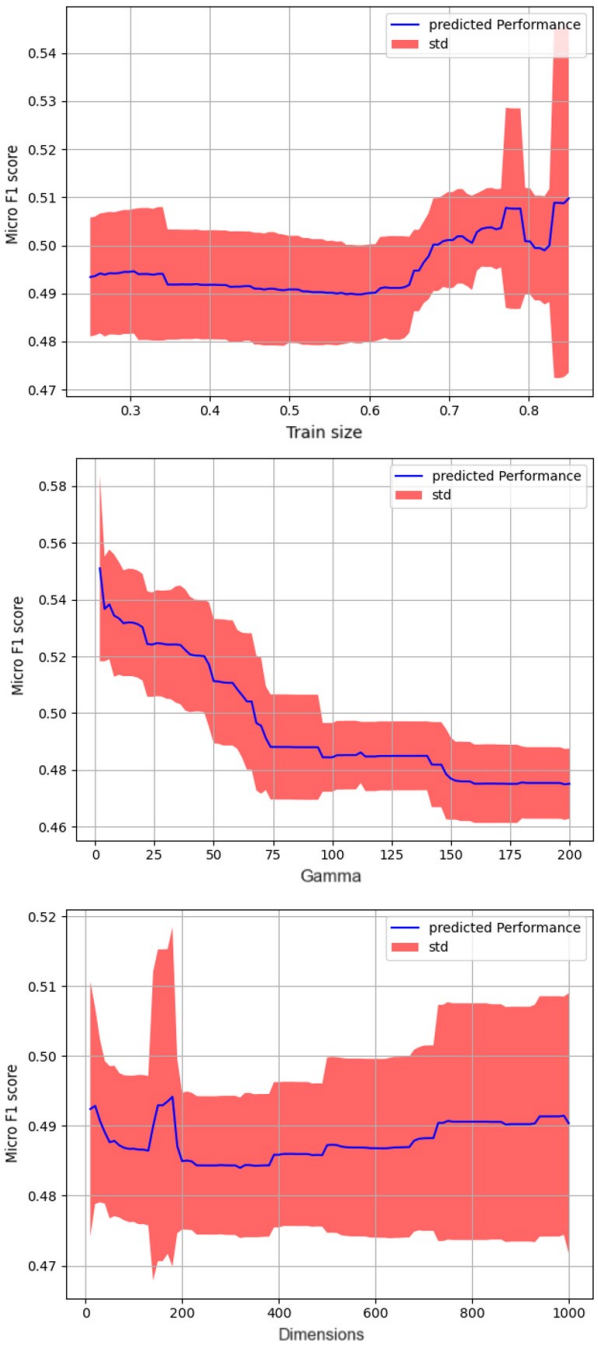

Figure 4.7: Individual marginal contribution plots, predicted by the functional ANOVA framework. Part 1.

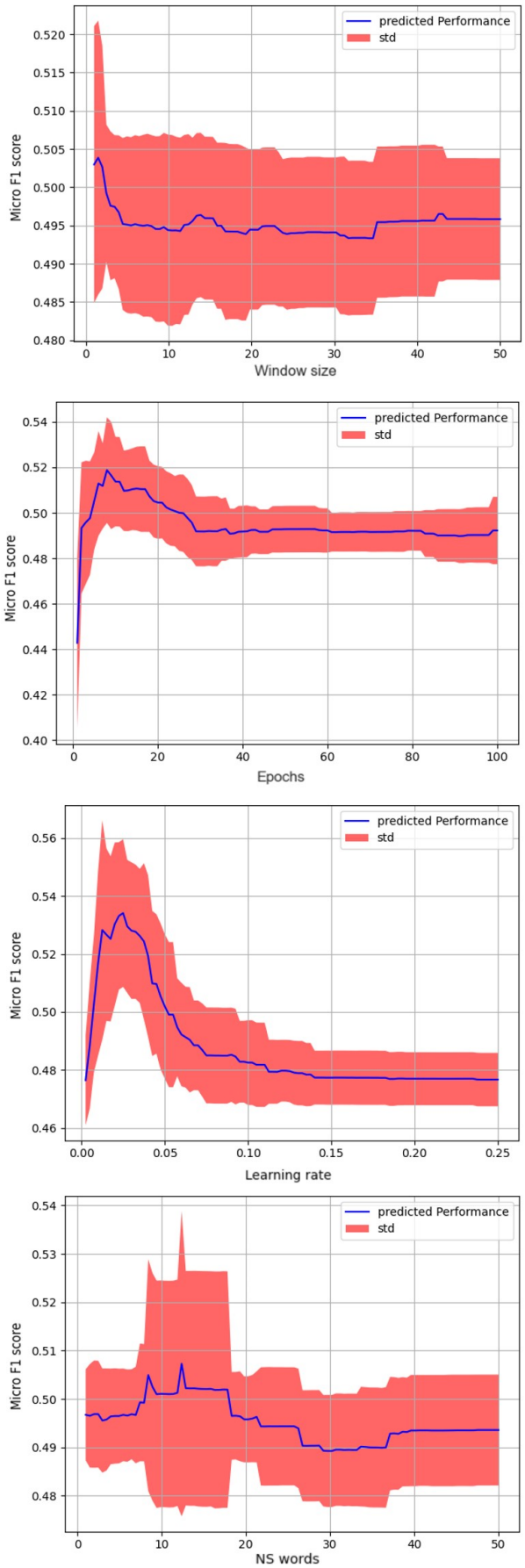

Figure 4.8: Individual marginal contribution plots, predicted by the functional ANOVA framework. Part 2.

4.4.2 Bayesian Optimization

In machine learning, hyper-parameter optimization problems usually encompass complex black box objective functions whose expression or derivatives are not known, restricting the evaluation to sampling singular points and extracting a possibly noisy response. With the aim of improving comprehension and steadily test the potentialities of Bayesian optimization for this setting, we will sequentially increment the number of free variables, prioritizing over the biggest marginal contributors (Figure 4.6).

Starting with Gamma and leaving its range to the original random search interval (Table 4.2), Figure 4.9 illustrates how the Bayesian optimization algorithm performs over a total of 100 iterations (shown on the left), and how it explores the relaxed state space (shown on the right). Note that in empirical trials, it was found that longer iterations often did not lead to better outcomes, even for higher free variable counts.

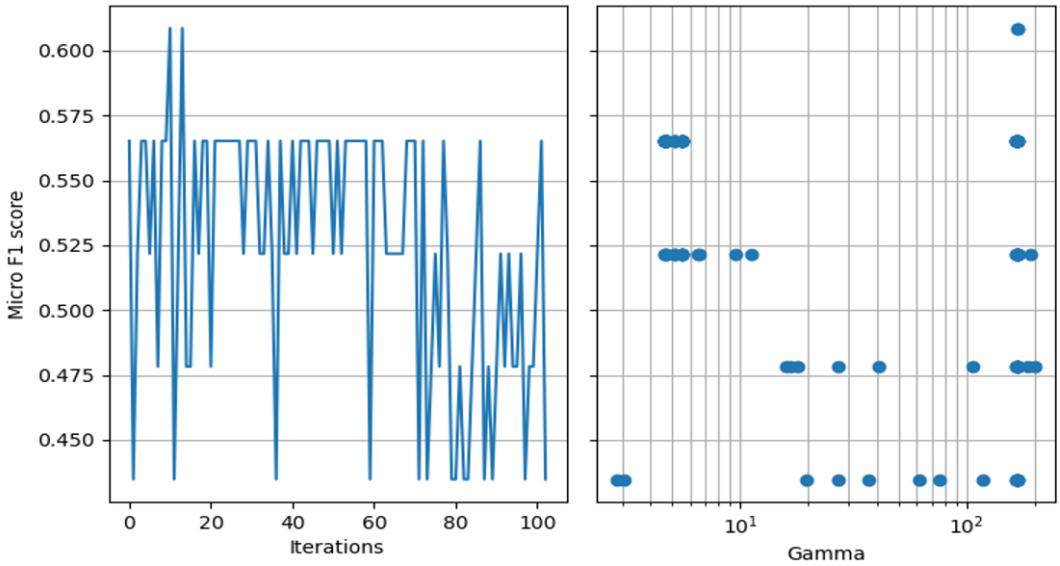

Figure 4.9: Bayesian optimization results with one free variable.

From Figure 4.9 it can be observed that the optimization algorithm explores both lower and higher ranges of the free spectrum, searching for the optimal regions. We may also note that the best result is achieved relatively quickly, at around iteration 15, and does not improve beyond this point. Unfortunately, the best achieved result of 0.61 is still worse than that of the baseline model, of 0.65, meaning that relaxing only one variable might not be sufficient to achieve significant improvements. To broaden the search scope, we ran the Bayesian optimization algorithm once again, but now with an additional free variable, Learning rate. Figure 4.10 shows the subsequent search distribution for both variables.

A far better result can be observed in this new configuration. The classification system significantly surpasses the baseline model, reaching a maximum Micro F1 score of 0.821. We may also observe from the data distribution that Gamma has been explored in some of the same regions as in the previous iteration, but with a drastically different outcome. This is an evident reflection of the high association between Gamma and Learning rate. It is also relevant to point out that while there is a tendency for these variables to stay close to the performance peaks shown in Figures 4.7 and 4.8, these appear not

to be definitive guidelines, as high values may also be found in other areas of the configuration space.

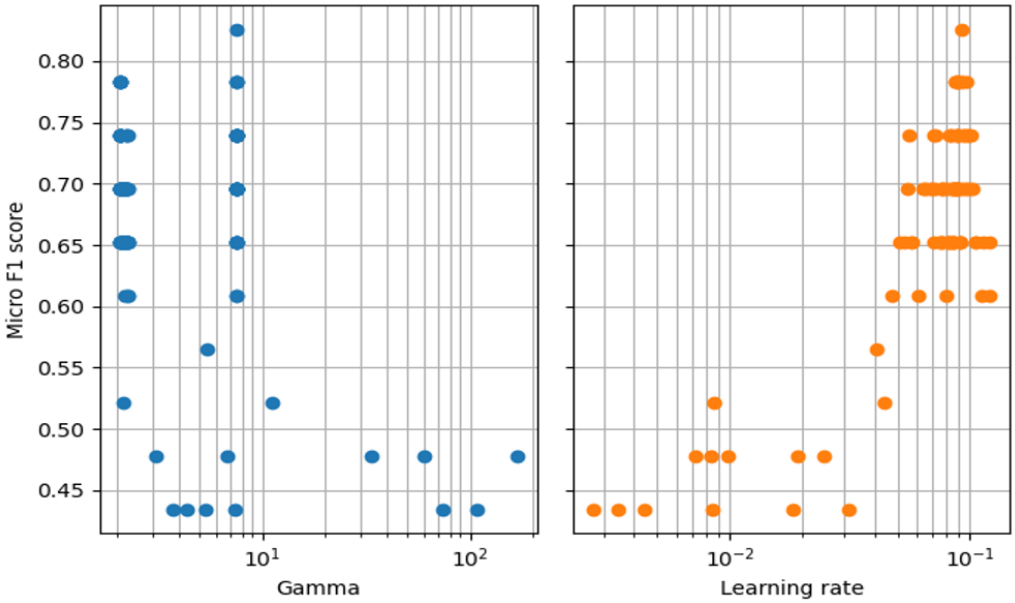

Figure 4.10: Bayesian optimization results with two free variables.

As the number of free variables increased, not necessarily did the final outcome. Similar values from the previous one were registered with three and four free variables, reaching a global best of 0.875 at four free variables. However, only lower values were obtained with five, six and seven free variables, reaching as low as 0.685 in these tests. This may suggest that, for the current maximization problem, there might be a limit to Bayesian optimization, situated around four free variables. This may be a consequence of the exponential expansion in configuration space, as well as in objective function complexity.

Moreover, empirical trials showed that, in general, over the three different HFACS-ML levels and four distinct D2V structures, global best results often appeared at around three free variables, widely surpassing the results from six or seven free variables. Due to this factor, in an effort to optimize computational efforts, we decided to limit our search of these models to five free variables. A summary of the best results extracted from the developed models is presented next, in Section 4.5.

4.5 Metric Results

To test the effectiveness of the developed text classification algorithm, after having incrementally applied Bayesian optimization with an increasing number of free variables on the four D2V embedding structures, for each level of the HFACS-ML framework, we took the best results from the optimized models and compared them against other baseline embedding and classification techniques. For this, we tried the previously tested TF-IDF as a D2V substitute, to represent the document vectors, and added a Support Vector Machine (SVM) as a potential substitute of LS, for the task of vector classification. To make it a fair comparison, we also included the results from our non-optimized baseline model, D2V DBoW NS + LS, after ”und” removal, and fixed Train size to 0.8 for the latter mentioned scenarios, as even most of the optimized models remained near this value. Additionally, we also took advantage of the random search

infrastructure, initially built to provide sufficient data for the marginal contribution analysis (Section 4.4.1), and retrained all the embeddings on this search mechanism, including another widely used optimization method in the analysis. The final comparison is summarized in Tables 4.3, 4.4 and 4.5, where each table retracts the results of a single HFACS-ML level.

From the results observed in Tables 4.3, 4.4 and 4.5, we distinctively attribute the best performance to the Bayesian optimization approach, as not only did it achieve most of the top scores but also presented more consistency throughout the different models. Comparatively, the random search approach provided acceptable results for a high enough number of iterations, but it did not prove to be as optimal or as stable. This may be justified by its stochastic nature. It is also worth mentioning that, while each singular random search took around 43 minutes to complete each 350 iteration cycle, the Bayesian algorithm generally took about 47 minutes to incrementally explore the search space up to five free variables. However, this value can be significantly reduced to 28 minutes if the search interval is narrowed between two and four free variables, where the best results were often found. This demonstrates another reason for properly tuning the Bayesian optimization algorithm.

As for the comparison between methods, various conclusions may be extracted. In a primary analysis, it can be observed that the DBoW architecture generally performed better than the DM for the current data set, while little distinction could be made regarding NS or HS superiority. In a second inspection, it can also be observed that the baseline semi-supervised LS algorithm outperformed the baseline supervised SVM in two levels (Unsafe Supervision and Unsafe Act), but underperformed in one (Precondition for Unsafe Act), coincidentally, the level with the highest label count. In addition, it should also be noted that the SVM dealt the worst with class imbalance, generally presenting the lowest Macro F1 scores. In contrast, a surprisingly good score resulted from the baseline TF-IDF + LS model, significantly surpassing the baseline D2V DBoW NS + LS on the Precondition for Unsafe Act and Unsafe Supervision levels. Due to this unexpected result, it seemed to be worth exploring up to what extent this configuration could perform if also optimized through Bayesian optimization. The best results for each level are shown in Table 4.6. Note that because the TF-IDF algorithm does not encompass hyper-parameters, the optimization of these trials consisted only in the tuning of the LS hyper-parameters, Gamma and Train size.

The results from Table 4.6 show that although this alternative combination did not perform exceptionally well for the Unsafe Supervision classification task, it did achieve near-best results on the Unsafe Act levels and exactly the same results as the top D2V performer in the Precondition for Unsafe Act level. This observation suggests that this simpler algorithm is still able to compete against newer state of the art embedding solutions. Moreover, it also confirms the volatility of these methods regarding different taxonomies, reflecting the difficulty of selecting between feature extraction algorithms in advance, for untested databases.

Table 4.3: Best predictions from the Unsafe Supervision level.

		Best Results		
Model Type	**Model Name**	**Micro F1**	**Precision**	**Macro F1**
Random Search	D2V DBoW NS' + LS	0.816	0.894	0.548
	D2V DBoW HS' + LS	0.850	0.928	0.531
	D2V DM NS' + LS	0.833	0.880	0.492
	D2V DM HS' + LS	0.800	0.900	0.471
Bayesian Optimization	D2V DBoW NS" + LS	0.900	0.933	0.578
	D2V DBoW HS" + LS	0.900	0.933	0.578
	D2V DM NS" + LS	0.850	0.928	0.510
	D2V DM HS" + LS	0.850	0.866	0.518
Baseline Models	D2V DBoW NS + LS	0.800	0.750	0.488
	D2V DBoW NS + SVM	0.500	0.500	0.133
	TD-IDF + LS	0.650	0.625	0.378
	TF-IDF + SVM	0.600	0.555	0.276

Table 4.4: Best predictions from the Precondition for Unsafe Act level.

		Best Results		
Model Type	**Model Name**	**Micro F1**	**Precision**	**Macro F1**
Random Search	D2V DBoW NS' + LS	0.657	0.656	0.659
	D2V DBoW HS' + LS	0.729	0.732	0.782
	D2V DM NS' + LS	0.610	0.618	0.517
	D2V DM HS' + LS	0.573	0.573	0.375
Bayesian Optimization	D2V DBoW NS' + LS	0.729	0.724	0.693
	D2V DBoW HS" + LS	0.779	0.789	0.735
	D2V DM NS" + LS	0.644	0.644	0.536
	D2V DM HS" + LS	0.627	0.632	0.494
Baseline Models	D2V DBoW NS + LS	0.407	0.407	0.197
	D2V DBoW NS + SVM	0.441	0.441	0.185
	TD-IDF + LS	0.763	0.759	0.745
	TF-IDF + SVM	0.661	0.661	0.560

Table 4.5: Best predictions from the Unsafe Act level.

		Best Results		
Model Type	**Model Name**	**Micro F1**	**Precision**	**Macro F1**
Random Search	D2V DBoW NS' + LS	0.800	0.800	0.818
	D2V DBoW HS' + LS	0.731	0.710	0.735
	D2V DM NS' + LS	0.704	0.704	0.495
	D2V DM HS' + LS	0.741	0.739	0.761
Bayesian Optimization	D2V DBoW NS' + LS	0.875	0.923	0.859
	D2V DBoW HS" + LS	0.826	0.842	0.830
	D2V DM NS" + LS	0.782	0.800	0.755
	D2V DM HS" + LS	0.869	0.850	0.898
Baseline Models	D2V DBoW NS + LS	0.565	0.526	0.560
	D2V DBoW NS + SVM	0.522	0.522	0.288
	TD-IDF + LS	0.739	0.737	0.762
	TF-IDF + SVM	0.652	0.652	0.447

Table 4.6: Best results of the TF-IDF + LS model, after Bayesian optimization.

	Best Results		
HFACS-ML Level	**Micro F1**	**Precision**	**Macro F1**
Unsafe Supervision	0.760	0.750	0.454
Precondition	0.779	0.789	0.735
Unsafe Act	0.869	0.857	0.512

Finally, as a complement to the analysis, and in order to provide a deeper understanding of the results, Figures 4.11, 4.12 and 4.13 exhibit the confusion matrices of the best models, for each level of the framework.

A significant improvement can be observed in these matrices compared to those shown in Section 4.3, from the initial baseline model. Not only may we observe more correct predictions, as indicated by the higher values in the main diagonals, but also a smaller influence of the most frequent categories over the minorities, especially in the Precondition for Unsafe Act level, as indicated by the lower values out of the main diagonal. This may represent the ability of these optimized models to better recognize some of the criterions, subjectively defined by the interpreters, to distinguish between document categories. A deeper discussion of the main takeaways from the presented results follows next, in Chapter 5.

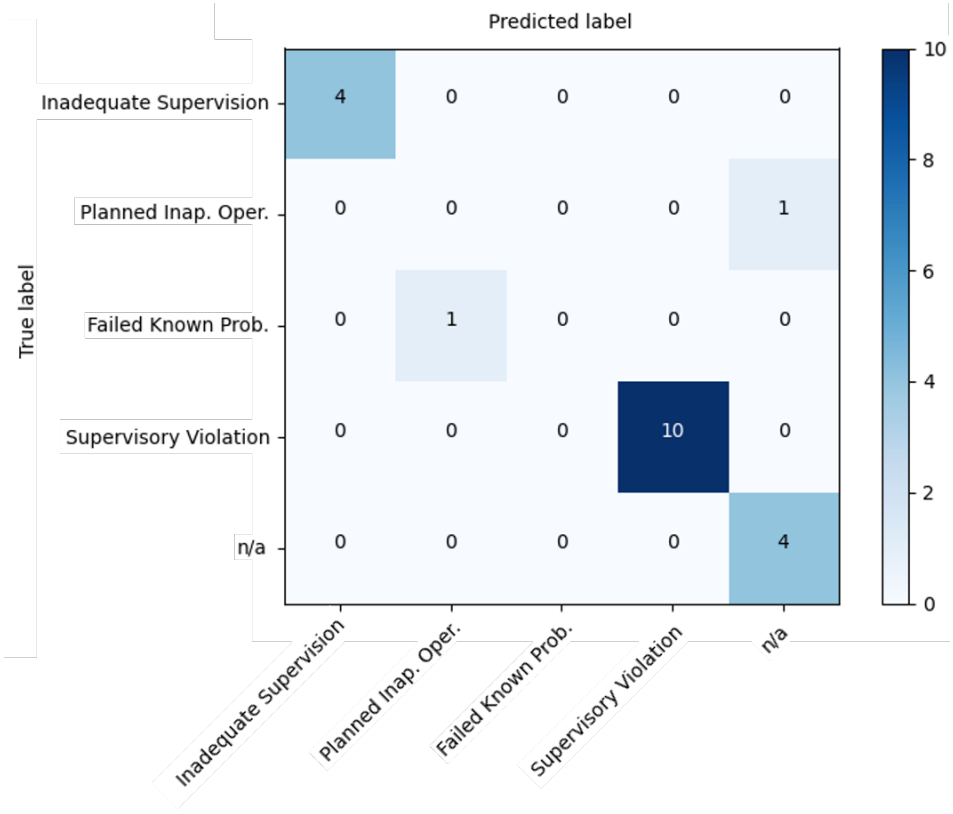

Figure 4.11: Best Unsafe Supervision confusion matrix.

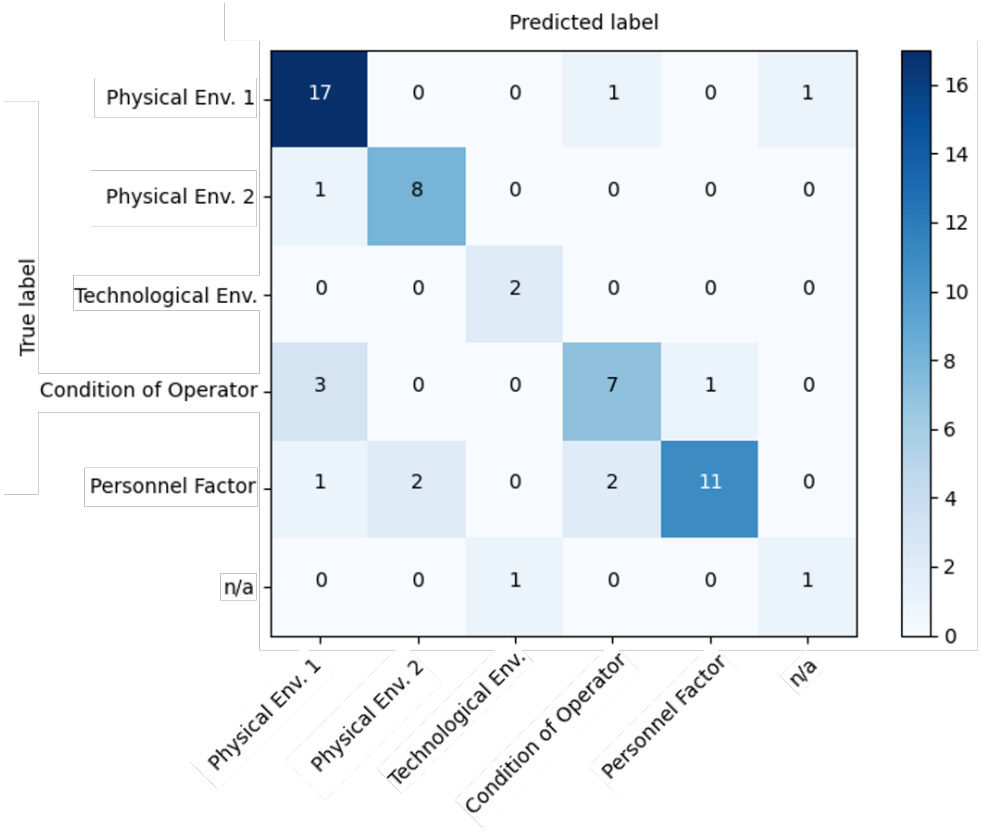

Figure 4.12: Best Precondition for Unsafe Act confusion matrix.

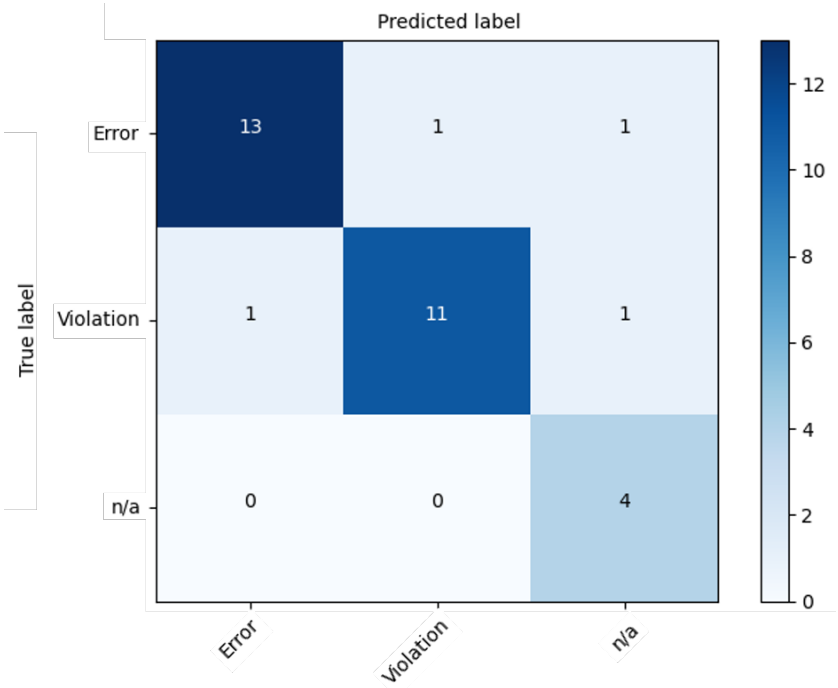

Figure 4.13: Best Unsafe Act confusion matrix.

Chapter 5

Conclusions and Future Work

In this closing chapter, an overall summary of the results of our study and final reflections on the methodology proposed to solve the classification problem are presented. In Section 5.1 we discuss the main takeaways and challenges identified throughout this research, as well as comment on the scientific questions initially proposed. Section 5.2 finalises with some prospects for future work.

5.1 Conclusions

5.1.1 Dissertation Overview

In this dissertation, a new document classification methodology for identifying human factors in aviation incident reports based on limited labelled data was developed.

For this purpose, we set out to design a pre-processing and natural language feature extraction pipeline that could be availed by a semi-supervised learning algorithm, to classify vector representations of ”Probable Cause” reports based on their state space distribution. The results, summarised in the evaluation scores presented in Chapter 4, illustrate the importance of selecting appropriate techniques, as well as using hyper-parameter optimization tools for maximizing the final outcome.

Moreover, the importance of a previously performed tailored data analysis applied to the initially raw ASN database, described in Chapter 2, should also be noted. With this analysis, we were able to take full advantage of the proposed methodology, by carefully pre-processing the corpora with reliable data mining techniques and deep data comprehension. Specifically, it allowed the representation of document vectors to be built on top of more representative texts, rather than general-purpose information. From this chapter, we also highlight the relevance of introducing the novel HFACS-ML taxonomic framework adapted for machine learning research, as well as the construction of a labelled set. Both these implementations allowed the models to be developed on clearly distinguished guidelines, adapted for the specific taxonomic domain.

In Chapter 3, we demonstrated, in a local setting, how embedding techniques can be used to associate the semantic meaning between some words through their spatial orientation. We also proved that the same logic can be applied to longer pieces of text, through the comparison of human factor cate-

gories on differently distanced documents. In this experiment, we observed that documents from similar categories could indeed be placed close to each other and that the D2V architecture could prove especially appropriate for this purpose, as it performed identically to the transformed (averaged) W2V, while being more computationally efficient. It was also speculated, in this chapter, that document embeddings should surpass the classic TF-IDF, due to their higher potential for capturing word relations.

The work developed in Chapters 2 and 3 was availed in Chapter 4, where we associated the labelled samples and document vectors with classification algorithms, to infer the category of unknown documents. In a preliminary analysis, using a D2V and LS combination, we gained some understanding on certain limitations that could corrupt our models, and iterated on this information to improve over the different levels.

Final results from Chapter 4 confirmed that the semi-supervised LS algorithm was an appropriate classifier for the current setting, particularly in the levels with fewer labels. We do not discard the potential of the supervised SVM for the same purpose, but note that it might prove more reliable for larger and more even labelled environments. Surprisingly, the TF-IDF model was also observed to be an acceptable alternative to the D2V for some levels of the framework, despite showing less consistency across levels and proving to be more computationally expensive because of its high dimensionality. In this sense, this algorithm may provide an interesting solution for studies which encompass smaller data sets and may benefit from hyper-parameter simplicity.

Another relevant conclusion to be taken from this study is the impact of Bayesian optimization, when properly tuned, for finding near-optimal hyper-parameter combinations, over non-convex objective functions. The fANOVA marginal contribution analysis was also crucial for this purpose, providing valuable insight into the most influential hyper-parameters. Further to this, we note that although random search optimization did not present the best results, and showed less consistency between trials, it still performed relatively well. Moreover, due to its stochastic properties, we note that it could possibly even surpass Bayesian optimization, although at a higher efficiency cost.

5.1.2 Challenges

Personally, one of the real values of this research lies on the motivated approach to find and validate appropriate solutions to a previously unexplored domain. However, the series of accomplishments and contributions described in this research did not come without their set of challenges. This section outlines the most meaningful obstacles encountered throughout the development process.

A primary limitation was found to be related to the quality of the acquired data. Document classification is based on the assumption that documents which belong to the same categories possess similar core vocabularies, and documents which belong to different categories contain different textual features. However, in the ASN database, core vocabularies would sometimes overlap, forcing the interpreter to make a subjective decision based on interpretation, which could diverge in an unpredictable manner to the mathematical representations described by the algorithms. This phenomenon was found in greater occurrence on the Precondition and Unsafe Act level.

Another significant limitation, whose impact is noted in Chapter 4, was the lack of a prior labelled set. This affected, on the one hand, the robustness of the final conclusions, as a larger labelled set would have been able to provide a more complete set of takeaways. On the other hand, it also affected the accuracy of predictions, since class imbalance generally influenced a higher tendency towards bias. We found this issue to be specially troubling to the analysis of the Unsafe Supervision level, which possessed the shortest and most imbalanced labelled set.

5.1.3 Research Questions

This section comments on the research questions proposed at the beginning of this dissertation.

- Can machine learning techniques be used to classify aviation incident reports, based on their latent human factors?

  Yes. As observed in Chapters 3 and 4, natural language processing and label propagation techniques can be successfully applied to predict human factors of unknown incident reports, up to a certain degree of precision.

  - If yes, can this classification be used for pseudo-label generation?

    Yes. Although the results presented in this research are based on a small labelled set, and thus may not infer a definitive representation of the entire state space, they still suggest that pseudo-labels may be extracted with some reliability, at least at local level. Therefore, for the purpose of pseudo-label generation, a threshold could be established, based on a LS information indicator, to define a confidence interval in which documents are accepted to be tagged with new labels.

  - If yes, can this classification be used for statistical analysis?

    Inconclusive. No definite conclusions could be taken regarding this question, due to the biased classification distribution originated from the unbalanced labelled set. However, this does not invalidate that the proposed methodology, based on an even labelled set, could be used for the same purpose. Further research regarding this topic should be developed.

- Can semi-supervised classification approaches outperform supervised techniques for small labelled data sets?

  Yes. However, there is an important prerequisite: that the distribution of unlabelled data be descriptive of the taxonomy in cause. As only in this case, the semi-supervised approach may take advantage of the additional information that passes undetected to the supervised method.

5.2 Future Work

In the present dissertation, a novel HFACS-ML categorization framework is proposed. It would be interesting to perform a study comparing how it would stack against the original HFACS on the same task.

Likewise, it would be relevant to analyse how different variations from these frameworks can better fit other machine learning applications and data sets.

Pointing out the high importance of conflicting vocabulary in the text classification task, which affects the subsequent hard distinction between categories, a possible solution to this challenge could be the introduction of fuzzy systems. The advantage of these techniques is that they enable to allocate each data point to more than one category, from the same level of the framework, with a certain degree of association. An alternative approach that could be applicable in a soft classification scenario is the Latent Dirichlet Allocation (LDA), a generative statistical classifier which is able to consider each document as a mixture of topics.

Another discussion concept that should also be considered is the application of the presented analysis on a larger labelled and unlabelled data set, in order to understand how the natural language processing and semi-supervised learning techniques perform in a scaled scenario. This fact would also motivate further research regarding other approaches for constructing labelled data sets. An interesting alternative to the methods used here is Active Learning. This method prioritizes the labelling of uncertain points, instead of randomly selected, so as to optimize convergence of label propagation algorithms.

Finally, feature selection studies, such as redundancy and noise analysis, should be carried out in greater depth. This topic is of great importance, particularly for the models described in this study, as they heavily rely on the quality and size of the vocabulary. Further studies could also be carried out on the investigation of other types of feature extraction and classification algorithms, as well as their respective combinations.

Bibliography

[1] ICAO. The world of air transport in 2018, . URL https://www.icao.int/annual-report-2018/Pages/the-world-of-air-transport-in-2018.aspx.

[2] EuroControl. Covid-19 impact on the european air traffic network. URL https://www.eurocontrol.int/covid19.

[3] E. Mazareanu. Global air traffic - scheduled passengers 2004-2021. URL https://www.statista.com/statistics/564717/airline-industry-passenger-traffic-globally/.

[4] L. Josephs. Delta is hiring 12,000 through 2020 as airline expands operations. URL https://www.cnbc.com/2019/10/10/delta-is-hiring-12000-through-2020-as-airline-expands-operations-ceo-says.html.

[5] S. Singh. 2019: The year united airlines expanded into new markets. URL https://simpleflying.com/2019-the-year-united-airlines-expanded-into-new-markets/.

[6] L. F. F. M. Santos and R. Melicio. Stress, pressure and fatigue on aircraft maintenance personal. 12(1):35. doi: 10.15866/irease.v12i1.14860. URL https://doi.org/10.15866/irease.v12i1.14860.

[7] K. A. Latorella and P. V. Prabhu. A review of human error in aviation maintenance and inspection. 26(2):133–161. doi: 10.1016/s0169-8141(99)00063-3. URL https://doi.org/10.1016/s0169-8141(99)00063-3.

[8] F. Schreiber. Human performance - error management. URL https://skybrary.aero/bookshelf/books/1640.pdf.

[9] N. R. Council. *Improving the Continued Airworthiness of civil aircraft: A strategy for the FAA's Aircraft Certification Service*. The National Academies Press. URL https://doi.org/10.17226/6265.

[10] ICAO. *Annex 19 to the Convention on International Civil Aviation - Safety Management*. ICAO, . URL https://www.pilot18.com/wp-content/uploads/2017/10/Pilot18.com-ICAO-Annex-19-Safety-Management.pdf.

[11] V. D. Pasquale, R. Iannone, S. Miranda, and S. Riemma. An overview of human reliability analysis techniques in manufacturing operations. In M. M. Schiraldi, editor, *Operations management*, pages 221–240. IntechOpen. URL https://doi.org/10.5772/55065.

[12] P. Trucco and M. C. Leva. A probabilistic cognitive simulator for HRA studies (PROCOS). 92(8): 1117–1130. doi: 10.1016/j.ress.2006.06.003. URL https://doi.org/10.1016/j.ress.2006.06.003.

[13] Y.-L. Hsiao, C. Drury, C. Wu, and V. Paquet. Predictive models of safety based on audit findings: Part 1: Model development and reliability. 44(2):261–273, . doi: 10.1016/j.apergo.2012.07.010. URL https://doi.org/10.1016/j.apergo.2012.07.010.

[14] Y.-L. Hsiao, C. Drury, C. Wu, and V. Paquet. Predictive models of safety based on audit findings: Part 2: Measurement of model validity. 44(4):659–666, . doi: 10.1016/j.apergo.2013.01.003. URL https://doi.org/10.1016/j.apergo.2013.01.003.

[15] D. A. Wiegmann and S. A. Shappell. *A Human Error Approach to Aviation accident analysis. The human factors analysis and classification system*. Ashgate Publishing Limited.

[16] T. Pham, T. Tran, D. Phung, and S. Venkatesh. Predicting healthcare trajectories from medical records: A deep learning approach. 69:218–229. doi: 10.1016/j.jbi.2017.04.001. URL https://doi.org/10.1016/j.jbi.2017.04.001.

[17] J. Liu, Z. Zhang, and N. Razavian. Deep EHR: Chronic disease prediction using medical notes. In *Proceedings of the 3rd Machine Learning for Healthcare Conference*, volume 85 of *Proceedings of Machine Learning Research*, pages 440–464. PMLR. URL http://proceedings.mlr.press/v85/liu18b.html.

[18] E. Masciari, V. Moscato, A. Picariello, and G. Sperli. A deep learning approach to fake news detection. In D. Helic, G. Leitner, M. Stettinger, A. Felfernig, and Z. W. Raś, editors, *Foundations of Intelligent Systems*, volume 12117 of *Lecture Notes in Computer Science*, pages 113–122. Springer. doi: 10.1007/978-3-030-59491-6_11. URL https://doi.org/10.1007/978-3-030-59491-6_11.

[19] L. Tanguy, N. Tulechki, A. Urieli, E. Hermann, and C. Raynal. Natural language processing for aviation safety reports: From classification to interactive analysis. *Computers in Industry*, 78:80 – 95, 2016. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2015.09.005. URL http://www.sciencedirect.com/science/article/pii/S0166361515300464. Natural Language Processing and Text Analytics in Industry.

[20] S. Ganesh. Text analytics on yelp reviews. URL https://sud3010ganesh.github.io/2018-05-26-yelpreviewtextanalytics/.

[21] ASN. ASN aviation safety database. URL https://aviation-safety.net/database/.

[22] A. K. Uysal and S. Gunal. The impact of preprocessing on text classification. 50(1):104–112. doi: 10.1016/j.ipm.2013.08.006. URL https://doi.org/10.1016/j.ipm.2013.08.006.

[23] S. Vajjala, B. Majumder, A. Gupta, and H. Surana. *Practical Natural Language Processing: A comprehensive guide to building real-world NLP systems*. O'Reilly Media.

[24] S.-H. Han. Googletrans 3.0.0. URL https://pypi.org/project/googletrans/.

[25] S. Bird. Nltk 3.5. URL https://pypi.org/project/nltk/.

[26] T. Korenius, J. Laurikkala, K. Järvelin, and M. Juhola. Stemming and lemmatization in the clustering of finnish text documents. In D. A. Grossman, L. Gravano, C. Zhai, and D. A. Evans, editors, *Proceedings of the 13th ACM Int. Conf. Information and Knowledge Management*, pages 625–633. ACM. doi: 10.1145/1031171.1031285. URL https://doi.org/10.1145/1031171.1031285.

[27] V. Balakrishnan and E. Lloyd-Yemoh. Stemming and lemmatization: A comparison of retrieval performances. 2(3):262–267.

[28] P. Jackson and I. Moulinier. *Natural Language Processing for online applications: Text retrieval, extraction and categorization*. John Benjamins Publishing, 2nd edition. URL https://doi.org/10.1075/nlp.5.

[29] R. Priyadharshini. Simple overview of machine learning. URL https://www.slideshare.net/priyadharshiniR8/simple-overview-of-machine-learning.

[30] L. Havrlant and V. Kreinovich. A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation). 46(1):27–36. doi: 10.1080/03081079.2017.1291635. URL https://doi.org/10.1080/03081079.2017.1291635.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in {P}ython. 12:2825–2830.

[32] Q. V. Le and Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st Int. Conf. on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196. PMLR. URL http://proceedings.mlr.press/v32/le14.html.

[33] T. S. Project. Word embeddings. URL https://www.synthesisproject.org/resources.

[34] C. Manning and M. Lamm. CS224n: Natural language processing with deep learning. URL https://web.stanford.edu/class/cs224n/.

[35] Gensim. Gensim: topic modelling for humans. URL https://radimrehurek.com/gensim/.

[36] K. Spärk Jones. A statistical interpretation of term specificity and its application in retrieval. In P. Willett, editor, *Document retrieval systems*, pages 132–142. Taylor Graham Publishing.

[37] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd Int. Conf. on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 957–966. PMLR. URL http://proceedings.mlr.press/v37/kusnerb15.html.

[38] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In Y. Bengio and Y. LeCun, editors, *Workshop Proceedings of the 1st Int. Conf. on Learning Representations*, pages 1–12, . URL https://arxiv.org/abs/1301.3781.

[39] S. Haykin. *Neural Networks: A comprehensive foundation*. Macmillan.

[40] A. Bhalgat. An efficient scale-out deep learning cloud – your way. URL https://blog.mellanox.com/2018/12/an-efficient-scale-out-deep-learning-cloud-your-way/.

[41] L. Weng. Learning word embedding. URL https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html.

[42] W. Wolf. Deriving the softmax from first principles. URL http://willwolf.io/2017/04/19/deriving-the-softmax-from-first-principles/.

[43] X. Rong. word2vec parameter learning explained. URL https://arxiv.org/abs/1411.2738.

[44] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26, pages 3111–3119. Curran Associates, . URL https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[45] Google. word2vec. URL https://code.google.com/archive/p/word2vec/.

[46] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut. A brief survey of text mining: Classification, clustering and extraction techniques. URL https://arxiv.org/abs/1707.02919.

[47] T. Schnabel, I. Labutov, D. Mimno, and J. Joachims. Evaluation methods for unsupervised word embeddings. In L. Màrquez, C. Callison-Burch, and J. Su, editors, *Proceedings of the 2015 Conf. on Empirical Methods in Natural Language Processing*, pages 298–307. Association for Computational Linguistics. doi: 10.18653/v1/d15-1036.

[48] V. A. Dabeeru. User profile relationships using string similarity metrics in social networks. URL https://arxiv.org/abs/1408.3154.

[49] S. learn developers. Sklearn.manifold.TSNE, . URL https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.

[50] Semi-supervised learning. URL https://doi.org/10.7551/mitpress/9780262033589.001.0001.

[51] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. URL http://www.cs.cmu.edu/~zhuxj/pub/CMU-CALD-02-107.pdf.

[52] M. Speriosu, N. Sudan, S. Upadhyay, and J. Baldridge. Twitter polarity classification with label propagation over lexical links and the follower graph. In O. Abend, A. Korhonen, A. Rappoport, and

R. Reichart, editors, *Proceedings of the 1st Workshop on Unsupervised Learning in NLP*, pages 53–63. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W11-2207.

[53] Y. Murase, K. Yoshino, M. Mizukami, and S. Nakamura. Feature inference based on label propagation on wikidata graph for DST. In *International Workshop on Spoken Dialogue System Technology*, pages 1–12. URL https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.iwsds2017/papers/IWSDS2017_paper_12.pdf.

[54] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16, pages 321–328. The MIT Press. URL https://proceedings.neurips.cc/paper/2003/file/87682805257e619d49b8e0dfdc14affa-Paper.pdf.

[55] S. learn developers. Sklearn.semi_supervised.LabelSpreading, . URL https://scikit-learn.org/stable/modules/generated/sklearn.semi_supervised.LabelSpreading.html.

[56] S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In H. Dai, R. Srikant, and C. Zhang, editors, *Advances in Knowledge Discovery and Data Mining*, volume 3056 of *Lecture Notes in Computer Science*, pages 22–30. Springer. doi: http://doi-org-443.webvpn.fjmu.edu.cn/10.1007/978-3-540-24775-3_5.

[57] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. 13:281–305.

[58] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The bayesian optimization algorithm. In W. Banzhaf, J. M. Daida, A. E. Eiben, M. H. Garzon, and V. Honavar, editors, *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*, volume 1, pages 525–532. Morgan Kaufmann Publishers. URL https://dl.acm.org/doi/10.5555/2933923.2933973.

[59] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 2951–2959. Curran Associates. URL https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

[60] F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st Int. Conf. on Machine Learning*, volume 32(1) of *Proceedings of Machine Learning Research*, pages 754–762. PMLR. URL http://proceedings.mlr.press/v32/hutter14.html.

[61] G. Hooker. Generalized functional anova diagnostics for high-dimensional functions of dependent variables. 16(3):709–732. doi: 10.1198/106186007X237892. URL https://doi.org/10.1198/106186007X237892.

[62] F. Hutter and S. Falkner. fANOVA 2.0.5 documentation. URL https://automl.github.io/fanova/index.html.

Appendix A

List of Keywords

The complete list of keywords extracted from the ASN database, and considered in this study for the automated labelling process is presented from Tables A.1 to A.3, in alphabetical order.

Keyword
ATC & navigation
ATC & navigation - Altimeter setting
ATC & navigation - Language/communication problems (also flightcrew)
ATC & navigation - Premature descent
ATC & navigation - VFR flight in IMC
Airplane - Airframe
Airplane - Airframe - Cargo door
Airplane - Airframe - Fuselage
Airplane - Airframe - Passenger door
Airplane - Airframe - Tail
Airplane - Airframe - Wing
Airplane - Engines
Airplane - Engines - APU
Airplane - Engines - All engine powerloss
Airplane - Engines - Fire
Airplane - Engines - Fuel contamination
Airplane - Engines - Fuel exhaustion
Airplane - Engines - Fuel starvation
Airplane - Engines - Loss/opening of engine cowling
Airplane - Engines - Precipitation-induced flame-out
Airplane - Engines - Prop/turbine blade separation
Airplane - Engines - Reverse thrust/prop ground fine pitch
Airplane - Engines - Separation
Airplane - Engines - Shutdown of wrong engine
Airplane - Engines - Uncontained failure
Airplane - Flight control surfaces - Elevator
Airplane - Flight control surfaces - Flaps
Airplane - Flight control surfaces - Horizontal stabilizer
Airplane - Flight control surfaces - Rudder
Airplane - Instruments
Airplane - Instruments - ADI
Airplane - Instruments - Autopilot
Airplane - Instruments - Pitot heads
Airplane - Pressurization
Airplane - Systems - Electrical system
Airplane - Systems - Hydraulics
Airplane - Undercarriage
Airplane - Undercarriage - Brakes
Airplane - Undercarriage - Gear-up landing
Airplane - Undercarriage - Landing gear collapse
Airplane - Undercarriage - Premature retraction on take-off
Airplane - Undercarriage - Tire failure
Cargo
Cargo - CofG
Cargo - Fire/smoke
Cargo - Overloaded
Cargo - Shift
Collision - Aircraft - In flight
Collision - Aircraft - On ground (platform)

Table A.1: Complete list of keywords extracted from the ASN database, Part 1.

Keyword
Collision - Aircraft - On ground (runway incursion)
Collision - Object
Collision - Object - Airport equipment
Collision - Object - Approach, rwy lights
Collision - Object - Bird
Collision - Object - Houses, buildings
Collision - Object - Mast/pole/wires
Collision - Object - Person, animal
Collision - Object - Trees
Collision - Object - Vehicle (on runway)
Collision - Object - Wall, dyke
External factors - FOD
External factors - Wake vortex
Fire
Fire - Fire
Fire - Fire Resulting from tire failure
Fire - Fuel tank explosion
Fire - Hangar, ground fire
Fire - Inflight
Fire - Litium battery thermal event
Flightcrew
Flightcrew - Alcohol, drug usage
Flightcrew - Disorientation, situational awareness
Flightcrew - Distraction in cockpit
Flightcrew - Failure to monitor instruments
Flightcrew - Incapacitation
Flightcrew - Insufficient rest / fatigue
Flightcrew - Mental condition
Flightcrew - Navigational error
Flightcrew - Non adherence to procedures
Flightcrew - Un(der)qualified
Landing/takeoff - Landing - Bounced
Landing/takeoff - Landing - Fast
Landing/takeoff - Landing - Heavy
Landing/takeoff - Landing - Late, far down rwy
Landing/takeoff - Landing - Undershoot
Landing/takeoff - Landing - Unstabilized approach
Landing/takeoff - Landing - Wrong runway/taxiway
Landing/takeoff - Tailstrike
Landing/takeoff - Takeoff - Aborted
Landing/takeoff - Takeoff - Locked rudders/ailerons/gustlock
Landing/takeoff - Takeoff - Wrong runway (runway confusion)
Landing/takeoff - Takeoff - Wrong takeoff configuration (flaps/trim)
Maintenance
Maintenance - (repair of) previous damage
Maintenance - Engine (deficient, inspections etc)
Maintenance - Failure to follow AD and SB's
Maintenance - Substandard practices (general)
Maintenance - Wrong installation of parts

Table A.2: Complete list of keywords extracted from the ASN database, Part 2.

Keyword
Result - CFIT - Hill, mountain
Result - CFIT - Hill, mountain (presumed)
Result - CFIT - Level ground
Result - CFIT - Water
Result - Damaged on the ground
Result - Deliberate
Result - Emergency, forced landing - Ditching
Result - Emergency, forced landing - On runway
Result - Emergency, forced landing - Outside airport
Result - Emergency, forced landing - crashed during
Result - Loss of control
Result - Loss of control (presumed)
Result - Normal landing
Result - Runway excursion
Result - Runway mishap
Result - Undershoot / overshoot
Security - Suicide
Weather
Weather - Heavy rainfall
Weather - Icing
Weather - Lightningstrike
Weather - Thunderstorm
Weather - Turbulence
Weather - Visibility -low
Weather - Windshear/downdraft

Table A.3: Complete list of keywords extracted from the ASN database, Part 3.