TODO TITLE

A report submitted to the University of Manchester for the degree of Bachelor of Science in the Faculty of Science and Engineering

Author: Sai Putravu
Student id: 10829976
Supervisor: TODO

2025

School of Computer Science

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

Alphabetically sort this

AE	Auto-encoding.
AR	Auto-regressive.
BERT	Bidirectional Encoder Representation from Transformers.
CBM	Condition-based maintenance policy.
CLI	Command Line Interface.
CNN	Convolutional Neural Network.
CUDA	Compute Unified Device Architecture.
DBSCAN	Density-Based Spatial Clustering of Application with Noise.
DL	Deep Learning.
ELMo	Embeddings from Language Models.
EPB1	Extracted proton beam line 1.
EPB2	Extracted proton beam line 2.
FAP	Fault Analysis Pathway.
FLD	First-line diagnosis system.
GPT	Generative Pre-trained Transformer.
GPU	Graphics Processing Unit.
CPU	Central Processing Unit.
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Application with Noise.
HDD	Hard-Disk Drive.
HEDS	High energy drift space.
IoT	Internet of Things.
LEBT	Low energy beam transport line.
LINAC	Linear Accelerator.
LSA	Latent Semantic Analysis.
MCR	Main Control Room.
ML	Machine Learning.
NLP	Natural Language Processing.
Operalog	The ISIS Operational Log.
PCA	Principle Component Analysis.
PdM	Predictive maintenance policy.
PLM	Pre-trained Language Model.
PvM	Preventative maintenance policy.
R2F	Run-to-failure maintenance policy.
MTEB	Massive Text Embedding Benchmark.
RFQ	Radio-frequency quadrupole.
RF	Random Forest.
SAFE	Supervised Aggregative Feature Extraction.
SMART	Self-monitoring and reporting technology.
STFC	Science and Technology Facilities Council.
SVD	Singular Value Decomposition.
SVM	Support Vector Machine.
TS-1	Target Station 1.
TS-2	Target Station 2.
t-SNE	t-distributed Stochastic Neighbour Embedding.
UKRI	United Kingdom Research and Innovation.
UMAP	Uniform Manifold Approximation and Projection.

Abstract

TODO

Chapter 1

Introduction

This project concerns the use of statistical and machine learning models to augment the predictive maintenance process at the STFC Rutherford Appleton Laboratory's ISIS Neutron and Muon research facility. The ISIS Neutron and Muon facility is a research centre for physical and life sciences, owned and operated by the STFC, a council that forms the UK Research and Innovation. In order to produce beams of neutrons and muons, allowing scientists to study materials at the atomic level, large and complex structures and machinery are required. The facility has a wealth of instrumentation taking multitudes of measurements to ensure that proper maintenance is completed in a timely manner.

Finish this off, also mention this is one of two parts. And this aims to focus on the Operalog, a rich textual record of real-world failures of the of the ISIS facility.
Also, mention that this is a very open-ended research project and what we should be judged on? i.e. progressing the space of auto-categorisation of industrial operational logs.

: Introduce the sections of the paper.

- Refine this after coming back
- Section 2,
- Section 3,
- ...

1.1 Motivation

Motivate the research project.

: The things in this section will include

- Introduce the problem: Auto-categorisation and label inference
- Highlight the research aims. This should highlight the vastly open nature of the project and then hone in on the particular issue I am tackling. (i.e. to progress the state of PdM ...)
- Identify the data input, expected output, data shape and explain why this motivates the project

1.2 Overview of the ISIS Research Facility

The STFC Rutherford Appleton Laboratory's ISIS Neutron and Muon research facility is a research centre for physical and life sciences. It is owned and operated by the STFC, a council within the United Kingdom Research and Innovation (UKRI). The UKRI is a public body of the United Kingdom's Government that directs funding for research and innovation through the science budget of the Department for Science, Innovation and Technology. ISIS was designed in the 1970s and early 1980s, with the core of the design being a strong-focusing machine with six radio frequency accelerating cavities to provide an average beam current of $200\mu A$ [62]. According to (Thomason [62]), with the introduction of ISIS and other neutron sources, rapid development of neutron instrumentation was stimulated. Over the years, the facility has been augmented with numerous components and instruments, such as the second target station - the current schematic representation of the facility can be seen in Figure 1.1.

1.2.1 The end-to-end production of neutrons and muons

The ISIS facility, described in A Practical Guide to the ISIS Neutron and Muon Source (ISIS Neutron and Muon Source [23]), is comprised of four major stages: (1) the ion source, (2) the radio-frequency quadrupole (RFQ), (3) linear accelerator (LINAC) and (4) the synchrotron. These components combine to produce a 800 MeV proton beam that is directed to either target station 1 (TS-1) or target station 2 (TS-2). Starting with H^- ions (protons with two orbiting electrons), the path taken through the four stages is shown in Figure 1.2. Throughout the entire H^- or proton acceleration process, the beam must be kept under tight vacuum and, to do so, many tens of vacuum pumps are maintaining the vacuum between 10^{-8} and 10^{-9} of atmospheric pressure. Maintaining this vacuum integrity is a constant operational challenge as leaks may disrupt the beam and potentially damage sensitive components, necessitating numerous pumps and continuous monitoring system [23]. The rest of this section explains the various stages in a more detail with the purpose of highlighting the complex infrastructure in place and critical control required for the reliable operation of the facility.

The first stage of the ISIS machine is the ion source, which generates the negative hydrogen (H^-) ions, which are then accelerated through the RFQ and LINAC. This ion source is a pulsed source that ionises hydrogen gas via an electric discharge between an internal anode and cathode [23]. As detailed in the guide, around 20ml of hydrogen gas per minute is continuously delivered to the ion source from a hydrogen gas bottle. Once the H^- ions emerge from the ion source, they have an energy of 35 keV and are presented to the RFQ via the Low Energy

Figure 1.1: The schematic representation of the physical layout of ISIS. The light grey areas are footprints of the buildings. Source: (Thomason [62]).

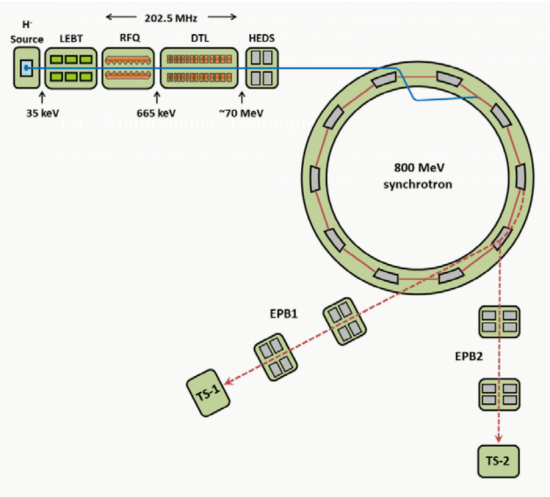

Figure 1.2: ISIS ion source and chain of accelerators, with H^- ions in blue and protons in red. Not to scale. Source: (ISIS Neutron and Muon Source [23]).

Beam Transport line (LEBT). The LEBT prevents the low-energy H^- beam from increasing in size (due to the mutual repulsion of the ions) and also incorporates a beam stop. This is essentially a remotely removable sheet of metal that physically blocks the beam. Precise control and monitoring of the beam intensity are required here since inconsistencies originating from the ion source affect the overall performance of the machine. This is because these issues can propagate and amplify through the subsequent stages. Additionally, the failure of the source itself, for example due to component wear, requires immediate intervention and is a common maintenance task [23].

Following the ion source and LEBT, the H^- beam enters the RFQ. This section of the machine utilises high-intensity radio frequency electric fields that contain (focus), group (bunch) and begin accelerating the particles in this beam up to an energy of 665 keV. A key function of the RFQ is to isolate downstream accelerators from variations originating in the ion source, thereby improving beam stability in the LINAC, synchrotron and target stations [23]. However, the stability of the RFQ's own high-intensity beam is prerogative as, again, fluctuations can lead to poorly formed bunches.

The subsequent stage is the LINAC, which increases the beam energy from 665 keV to 70 MeV. This acceleration is achieved via high-intensity radio-frequency fields arranged in structure to possess a very high Q factor [36] (around 50,000). This means their geometry must be exceptionally stable. Therefore the precise temperature regulation of the cooling water tank is critical in ensuring it is significantly under $1°C$ so that the effects of thermal expansion and contraction are countered [23]. Furthermore, maintaining a high vacuum is critical throughout the LINAC. Failure to do so can lead to premature stripping of loosely bound electrons in the H^- ions, consequently leading to an increase in radioactivity and particle loss (i.e. beam loss). To detect and warn about such cases, beam loss monitors are installed throughout the LINAC, which provide indicate whenever there is an excessive loss

in particles. [23].

Once the H^- particles leave the LINAC, they then pass through a beam transport line - the High Energy Drift Space (HEDS). The function of this is to mainly reduce the particles spreading and focus the beam to the synchrotron. To accumulate the high intensity beam in the synchrotron, ISIS employs charge-exchange injection. This technique allows incoming H^- beam from the LINAC to be 'layered' into the synchrotron's circumference over many turns. As the H^- ions enter the synchrotron ring, they pass through a thin foil which strips away their two electrons, converting them to protons. The charge-exchange injection process is specifically chosen as it is the most efficient way of taking and wrapping a long string of particles around the circumference of a synchrotron [23, 5]. The integrity of this thin foil is vital for efficient injection and represents a consumable component requiring periodic replacement, which is a scheduled maintenance activity. Damage to the foil will lead directly to beam loss and reduced performance.

The 70 MeV beam is then injected into the synchrotron, which is the primary accelerator ring at ISIS. This ring boosts the (now proton) beam energy to 800 MeV. For an idea of scale, the machine has a circumference of around 163m (corresponding to a radius of approximately 26m). It features a repeating structure of ten 'superperiods'. Each superperoid uses dipole magnets to bend the proton path by 36° and quadrupole magnets to maintain beam focus [23]. The protons gain energy over just under 8000 revolutions, as each turn gives a proton 0.1 MeV. Achieving this final energy requires extreme precision over each and every revolution. While the protons gain energy, the magnetic fields and accelerating radio-frequency voltages must, also very precisely, be increased in synchronism. Any deviations can cause the beam to oscillate and strike the vacuum vessel walls which results in particle loss and potential component damage. Therefore, continuous precise monitoring of beam position, magnet currents and more via a network of sensors is imperative for controlling the beam orbit and ensuring successful acceleration [23]. The initially continuous ring of injected protons is gathered into two distinct bunches, a little more than $0.1\mu s$ apart, using a low-level radio-frequency voltage before the main acceleration begins [23].

Once accelerated, in order to direct the proton bunches towards a pre-selected target station (TS-1 or TS-2) three kicker magnets are used which deflect the bunches upwards. These bunches are then bent into the respective extracted proton beam lines (EPB1 or EPB2). These lines use further magnets for steering. Safe transport relies on accurate magnet performance as mis-steering could direct the high-energy beam into the pipe walls. Additionally, as a safety precaution, both lines are heavily shielded in thick steel and concrete and beam loss monitors are installed along their length to detect any errant particles and provide crucial, real-time feedback to operators [23].

The primary purpose of the ISIS facility culminates at the target stations. The high-energy proton beam collides with a dense, high atomic number target (tantalum-clad tungsten at ISIS) designed to produce a large number of neutrons via spallation, within a compact region of space [23]. These energetic neutrons are then slowed down (moderated) by surrounding materials (liquid nitrogen, liquid hydrogen and water at ISIS) and reflected back towards instruments by a beryllium reflector [53, 23]. Additionally, ISIS produces muons for experiments by inserting a thin (roughly 1cm thick) graphite target into the proton beam around 20m away from the TS-1 neutron target. Collision in this target generates pions,

which subsequently decay into muons [23]. This entire system needs extremely careful control and monitoring capabilities, as failure due to overheating or material damage is a major operational concern which requires careful management and represents a major maintenance undertaking. Precise monitoring is required of metrics such as incoming beam profile, target temperature, cryogenic temperatures.

This detailed description sourced from the guide highlights some key takeaways. There are an incredible number of points of failure in the facility due to the highly precise and radioactive nature of the task. The ISIS team has taken many precautions such as fault detection mechanisms (i.e. the beam loss monitors) and breakdown counter-measures (i.e. the thick layer of steel and concrete in the EPB1 and EPB2 lines). However, the inherent complexity and harsh operating environment (radiation, high-power) necessitate both robust preventative maintenance schedules and sophisticated diagnostic capabilities that address the component degradation and failures that will inevitably arise.

1.2.2 Maintenance at ISIS

As detailed in the 33-year historical account of the ISIS facility (Thomason [62]), the ISIS operations occur in cycles, periods of roughly 30-50 days where the machine runs constantly without breaks. Gaps between cycles typically range from 1 week to 3 months. In addition, typically every four years, shutdowns scheduled for 6-9 months occur for major maintenance and upgrade work. On the ground, day-to-day operations are run from the Main Control Room (MCR) by the ISIS crew which consists of 6 shift teams of the following roles: (1) duty officer, (2) assistant duty officer and (3) duty technician. The expertise and rapid response capabilities of these trained operational crews are vital for both routine operation and initial fault response. Minimising this downtime is crucial for several reasons inherent to operating a large-scale user facility like ISIS [62]. Primarily, unscheduled downtime directly impacts the international researchers allocated beam time, potentially jeopardising experiments planned months or years in advance and wasting valuable research opportunities. Furthermore, interruptions disrupt the tightly packed operational schedule, reduce the overall scientific output, and incur significant operational costs without delivering the facility's core research product [62]. Therefore, understanding the factors that contribute to downtime and implementing strategies to mitigate it are paramount. Many factors affect downtime such as having a robust plan, the day-to-day operators' knowledge of the system and adequate inventories of spares [62]. Figure 1.3 shows the ratio of downtime to active machine operation since 2016, with over half the time of the machine since 2016 being down. In other words, over half the time, the machine is not active and is undergoing maintenance. This highlights the trade-off between maximising operational availability and minimizing failures through planned maintenance. The major maintenance strategies and their trade-offs are further explored in Section 1.3.

Add the graph, if you can ever figure out the operating cycles before 2016.

To help reduce downtime, over the last few years, a first-line diagnosis (FLD) system was introduced, presented in (Asim Yaqoob [6]). The FLD system helps reduce downtime by providing expert guidance on fault diagnosis and resolution, which has been shown to improve the dissemination of knowledge from experts to operations. The FLD utilises Fault Analysis Pathways (FAPs), which provide structural links between ISIS subsystems. This allows users

of the system to access granular subsystems' local documentation minimising file hunting and saving time and effort. An example FAP can be seen in Figure 1.4.

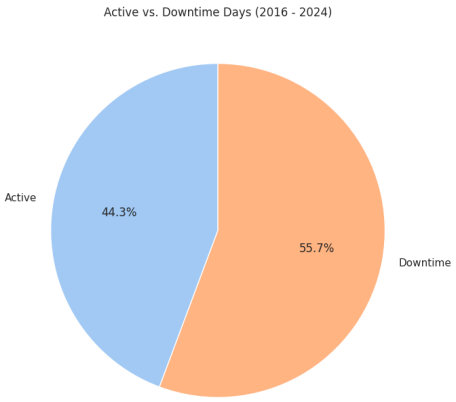

Figure 1.3: A visualisation of the machine downtime as opposed to time operating, according to the ISIS operational cycle. Data source: (ISIS Neutron and Muon Source [24])

: Introduce the research topic. The things in this section will include

- Talk about the ISIS research facility - DONE
- Talk about the Operational Cycle for ISIS - SORT OF DONE (graph too)
- Talk about the ISIS Crew and importance of having trained staff on premises. - SORT OF
- Talk about the Lost time and why it is important to minimise this for the ISIS research facility. - SORT OF
- Describe the first-line diagnosis system (FLD) and FAPs. - DONE
- Talk about the Datasets, operalog (MOVED)

1.3 Maintenance Techniques

In industry, the uptime of production systems are strongly coupled with the equipment maintenance. So much so that what was once considered a 'necessary evil' is now seen as a 'profit contributor' to be able to maintain a competitive edge [65, 20]. For facilities aiming to provide systems for research, maintenance impacts the downtime and cost of running. As a result, both to minimise unexpected downtime and provide a competitive edge, many industrial applications collect vast quantities of data during the entire life cycle of the system. This large amount of data may include information about processes, events and alarms [15] which occur along the industrial production line, collected by different equipment. The equipment may

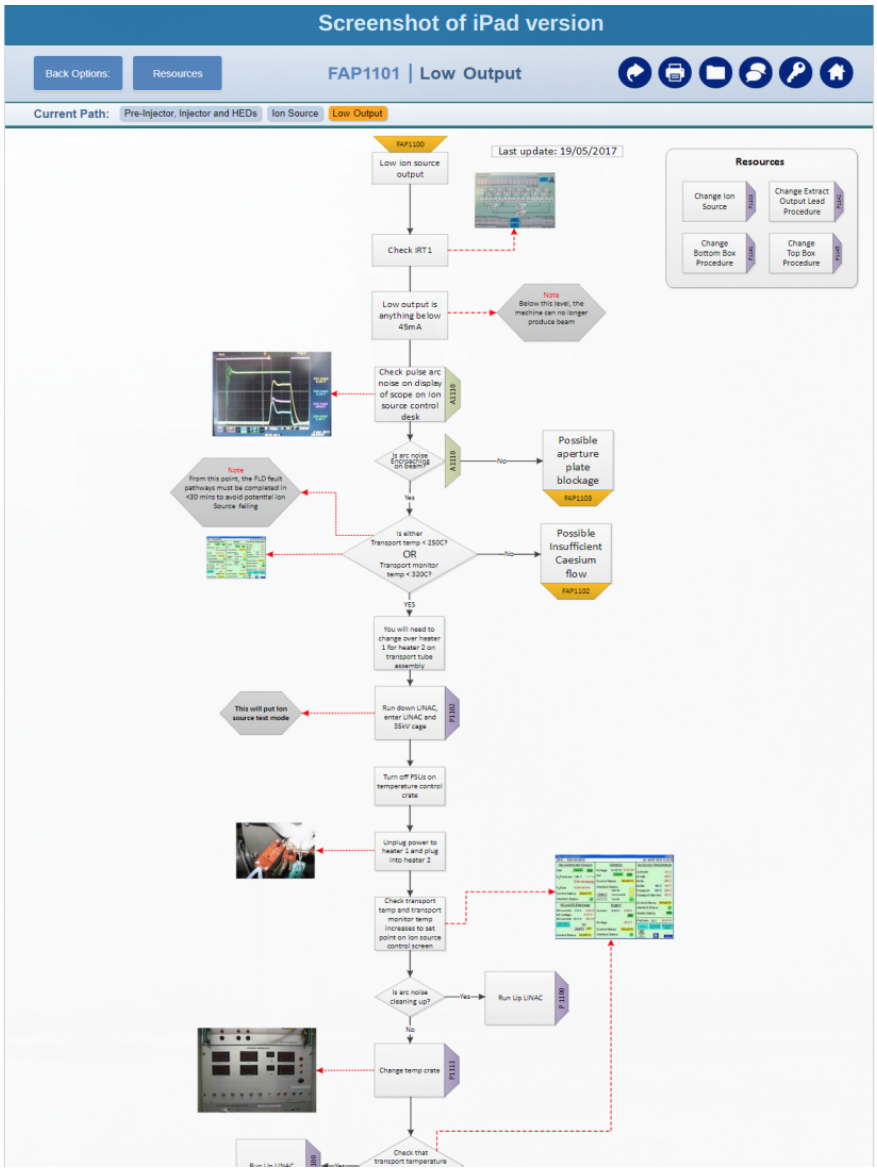

Figure 1.4: Example FAP1101, screenshotted from an iPAD version of FLD version 2.2, which shows the FAP for pre-injector, injector and High Energy Drift Space. Source: (Asim Yaqoob [6])

be located in different locations in the sub-components of the larger system or even different sub-components themselves.

talk about maintenance itself in a lot more depth

In the literature, various terms and categories of maintenance arise each with differing strategies [59, 39, 58]. Thus, while there exists some disagreement in nomenclature, we consider the four categories presented in (Susto et al. [59]). The four maintenance policy categories are as follows, noting that each policy has, uniquely, their own benefits and drawbacks:

1. Run-to-failure (R2F) maintenance: Continual usage of the system until failure. Restoration is performed at the point of noticing failure condition. This is the simplest approach and typically the most costly method as it reduces the facility's availability and requires a complete replacement of parts.

2. Preventative maintenance (PvM): Otherwise referred to as scheduled maintenance, performing maintenance at regular intervals to increase longevity of the component or in anticipation of the end of expected life of the component. While this typically prevents many errors, it wastes maintenance cycles when systems are perfectly healthy. Hence, causing unnecessary downtime and cost.

3. Condition-based maintenance (CBM): Taking the action to perform maintenance on equipment through monitoring various health characteristics and metrics of the components of the system. This approach requires continuous monitoring and, thus, allows for close to instant response on maintenance only when required. However, a drawback of this policy is that one cannot plan maintenances in advance.

4. Predictive maintenance (PdM): Otherwise referred to as statistical-based maintenance, only performs maintenance actions when determined necessary. Prediction tools are utilised to implement forward-planning and scheduling systems, using statistical inference methods. However, if these statistical inferences are not accurate, the whole system suffers which inevitably leads to additional downtime and costs.

It should be noted, that several sources conflate CBM and PdM [39]. As in (Susto et al. [59]), where they are given as separate categories, we follow suit.

The PdM strategy stands out in the four categories presented as, given a statistical inference model that is able to detect faults accurately, this policy optimises the trade-off between improving equipment condition, reduce failure rates for equipment and minimising maintenance costs [15]. This technique enables one to apply foresight for pre-emptive scheduling of large-scale maintenance. As pointed out in Section 1.2, the ISIS facility aims to strike a balance between PvM, CBM and PdM through periods of large-scaled scheduled maintenance and collection of high quantities of metrics. This balance is achieved through the careful coordination between cycle scheduling, day-to-day crew-based monitoring and the FLD [62].

In industry, many maintenance strategies prefer using PdM whilst experimenting with a variety of statistical inference and artificial intelligence modelling approaches [39, 26]. Some examples from [15] are listed in Table 1.1 which highlights the trend in the industry towards more accurate, ML-based approaches.

Table 1.1: Examples of applications of PdM for industrial maintenance strategies.

Type	Description	Reference
Statistical	Application of SAFE to deal with PdM problems characterised by time-series data. The approach is tested on a real-life dataset of the semiconductor ion implantation process.	(Susto and Beghi [58])
ML	Application of SVM classification for fault prediction of rail networks, with discussion on using the model in optimising trade-offs related to maintenance schedule and costs.	(Li et al. [32])
ML	Audio analysis on IoT devices, enabling acoustic event recognition for machine diagnosis. This paper describes designing an end-to-end system, utilising CNN-based classification.	(Pan et al. [45])
ML	Utilisation of RF decision trees trained on SMART data to predict reliability of HDD in real-time.	(Su and Huang [57])

Chapter 2

Natural Language Processing Background

This chapter delves into the technical background required to the methodology proposed in Chapter 3. Firstly, we discuss various measures of text similarity in Section 2.1, which motivates text embeddings and categorises similarity measures into different generations. Section 2.3 introduces both third-generation (BERT [17], XLNet [68] and MPNet [55]) and fourth-generation models (Nomic [42]). We specifically discuss the technical details of the BERT model and its family of encoder-only transformers. We then detail two direct improvements over BERT (XLNet and MPNet) and explore the state-of-the-art, fourth-generation Nomic architecture. In actuality, only MPNet and Nomic models are used in Chapter 3 however, as MPNet builds on top of XLNet and BERT and Nomic builds on top of BERT, it is important to understand the key features of all four architectures. Afterwards, we cover two methods of dimensionality reduction (PCA [46, 22] and UMAP [35]) motivated by the need to visualise samples from the high-dimensional embedding spaces of the aforementioned models, in Section 2.4. Finally, we present three clustering algorithms - with one partitioning-based (k-Medoids []) and two density-based (DBSCAN [19] and HDBSCAN [13]) in addition to four clustering evaluation metrics. The clustering metrics we look at are: (1) Inertia [], (2) Silhouette [50], (3) Davies-Bouldin Index [16], (4) Calinski-Harabasz Index [12].

Fill this

find the reference for this

maybe talk about optuna, if we use it.

double check all citations here are not empty

: Describe the various technical factors required before attempting to understand the methodology. The things in this section will include

- Discuss sentence embedding, similarity measures: BERT, RoBERTA, MPNet, XLNet, NOMIC.
- Dimensional reduction techniques and need for them (UMAP, PCA, t-SNE).
- Clustering methods: kmedoids, DBSCAN, DBSCAN*/HDBCAN
- Clustering evaluation methods: todo I don't remember these off the top of my head
- Maybe briefly touch on Optuna?

2.1 Sentence Similarity

Sentence similarity, otherwise referred to as document similarity, is the (NLP) task of computing the quantification of the similarities between two sentences, documents or texts. This task is motivated by the increasingly large amount of digitisation of human languages (and data, in general), calling for the need to understand similarity between various texts [49]. Examples of the use-cases of sentence similarity include: detection of academic malpractice via plagiarism [34, 7] and text summarisation [4, 28, 27]. According to [49], there are two main types of sentence similarities: (1) lexical similarity and (2) semantic similarity. The former is a computation of the equality between the lexicon of two sentences (i.e. a purely syntactical view), as opposed to the latter being a comparison between the semantics. Further, the type we focus on, semantic similarity can be split into three types:

- String-based similarity: Measures similarity directly between two strings, accounting for string sequences and character composition. These can be fine-grained, i.e. character-based; coarse-grained, i.e. term-based; or a hybrid mixture of both [69].
- Knowledge-based similarity: Measures the degree to which two sentences are related, utilising semantic networks (i.e. knowledge graphs). Examples of Knowledge-based similarity approaches include WordNet [11], the most popular type of approach.
- Corpus-based similarity: Premised on a provided corpus, a large database of text to derive inferences from. Methods of this type require the development statistical or DL models that train on the provided corpus and estimate the similarity between two sentence-pair inputs. Popular examples include traditional statistical models, such as LSA [30] and SVD [56] as well as word embedding models (utilising ML), such as Word2Vec [10], GloVe [47] and fastText [37].

Most of the models mentioned above require some numerical representation of the text to be able to apply mathematical procedures for similarity calculation. Computing this representation involves converting unstructured textual data into one or more vectors. Typically, this process includes (1) general natural language pre-processing steps such as stop-word removal, case normalisation, parts-of-speech tagging, lemmatisation, and tokenisation [60]; and (2) applying an embedding model, either to a single token (word embedding) or to a sequence of tokens (sentence embedding). Step (1) can be seen as a feature extraction step applied on the unstructured textual data, where feature extraction is the process of extracting the most useful components of the data [51]. For example, part-of-speech tagging can be seen as introducing non-trivial features to some token through extracting the surrounding context.

This representation is known as an embedding, with the span of the possible vectors referred to as the embedding space. The dimension of the span is termed the embedding dimension. This is an important concept because the characteristics of the embedding space influence the model's ability to capture syntactic and semantic meaning in text, as the embedding space itself encodes this information. This can be seen in the Word2Vec model, described in (Bojanowski et al. [10]), which shows different embedding dimensions produce different results. Another conclusion that can be drawn from this paper is that, if embedding space is

not constructed to maximise the meaning of texts, the accuracy of model predictions tends to deteriorate.

Therefore, the problem of sentence similarity can be directly mapped from the problem of sentence embedding (otherwise referred to as text embedding), where text embedding is the (NLP) task of learning a high-dimensional embedding space representation. Various aspects of text embedding are more thoroughly covered in Section 2.3. However, with the advent of the transformer architecture [64] and rise of the large language models, text embedding has been increasingly solved using DL models with high parameter counts [14] and considering extremely large token sequences. Nowadays, word embedding models are considered obsolete with (Cao [14]) only considering these models second-generation. Further, the paper states newer generations fall into the following categories:

- Third-generation: contextualised embeddings. These models dynamically account for contexts, encoding them into the embedding space. Examples of models include ELMo [52], GPT [48] and BERT [17]. As these models are trained to both understand some embedding space and generate natural language text, they are canonically referred to as language models.

- Fourth-generation: universal text embeddings. The generation which is currently state-of-the-art, with the aim of developing a unified model which is able to address multiple downstream tasks. Examples of models in this generation, making progress towards unification include Gecko [31], Multilingual e5 text embeddings [66], Nomic [42] and many more.

Second-, third- and fourth-generation text embedding models are used frequently in PdM for applications such as insight extraction [2, 63] and clustering intents from unstructured text data. Sources of natural language datasets, in industrial applications typically arise from operational or managerial log files which document aspects such as failures, resolutions and comments. Advanced text embedding models enable for semi- or fully automatic insight retrieval and auto-categorisation, enabling intuitive understanding of the textual datasets potentially highlighting patterns in failure [41].

2.2 (maybe) Clustering

Talk about clustering lit. rev.

Think whether it is useful to present literature review in this section.

Think of a good transition between Sentence Similarity and Clustering

: The things in this section will include

- (DONE) Looking at general predictive maintenance
- (Done) Looking at general predictive maintenance in industrial applications
- (Done) Similar pairwise sentence similarity literature
- Similar literature in text clustering
- (sort of DONE) Similar literature in specifically sentence clustering in industrial applications

2.3 Sentence Embedding

Briefly touched on in Section 2.1, sentence embedding (otherwise known as text embedding) is the NLP task of computing some high-dimensional embedding vector-space representation for unstructured text data. This vector-based representation should encode the semantic and syntactic meaning of the text and establish meaningful relationships. For example, the sentence 'I like dogs' should have the opposite representation to 'I hate dogs'. However these sentences should be more related than to the sentence 'My house was destroyed in an earthquake'. For a naive illustration of this, see Figure 2.1, which shows how similar sentences should be grouped together. As Nomic requires the task explicitly in input text (more on this later), 'clustering: ' is appended to all sentences. Deep learning sentence embedding models are now seen to be state-of-the-art as they are able to extract features automatically and more effectively than manual efforts, when supported with large quantities of data [33].

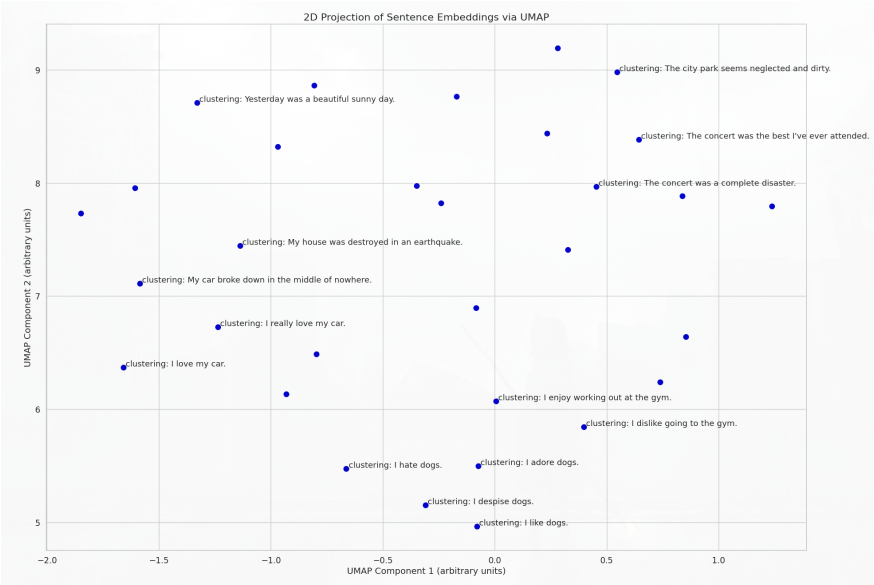

Figure 2.1: Example text embedding with Nomic Text Embedding v1.5 [42] of random sentences generated using OpenAI's ChatGPT o3-mini-high and projected onto 2 dimensions using UMAP [35] with a random seed of 42 and minimum neighbours of 15. Only some of the labels are highlighted for visual clarity.

2.3.1 BERT-family Transformers

Pre-trained language models (PLMs), such as BERT [17], have been widely successful in a wide range of NLP tasks through fine-tuning [18, 38]. Fine-tuning is the process of re-training a PLM on specialised tasks, leveraging the model's base knowledge, by applying perturbations to the pre-trained model parameters through gradient descent learning algorithms. These language models, trained on finding an embedding space for natural language as well as

stochastically generating tokens to mimic natural language, often provide a great platform to perform task-specific model fine-tuning. Platforms such as HuggingFace [67], allow authors to upload these pre-trained model parameters which in-turn allows researchers to download them for them fine-tuning. Moreover, task-specific fine-tuned models parameters are uploaded, downloaded and shared on these platforms. These fine-tuned models are useful for researchers whose focus lies outside of optimising these model parameters. In this section, we talk about the BERT-family of language models.

fix wording

BERT

The Bidirectional Encoder Representations from Transformer (BERT) language model introduced in 2018 by Google AI Language team in (Devlin et al. [17]) was designed to learn deep bidirectional representations from unlabelled text. It achieved this by learning the left and right context in every layer of the model. The model implementation can be split into two phases: (1) the pre-training phase; and (2) the fine-tuning phase. This allows the model architecture to remain common, with many down-stream NLP tasks benefiting from a single PLM. An example illustration can be seen in Figure 2.2, where a single model can be fine-tuned on many down-stream tasks by simply replacing the output layer. The model architecture is a multi-layer Transformer encoder based on the original paper (Vaswani et al. [64]) which is bidirectional by nature. As the transformer architecture is a very well researched architecture, well represented in the literature and slightly out-of-scope for this paper, we refer the reader to the original paper.

sprinkle some images in here.

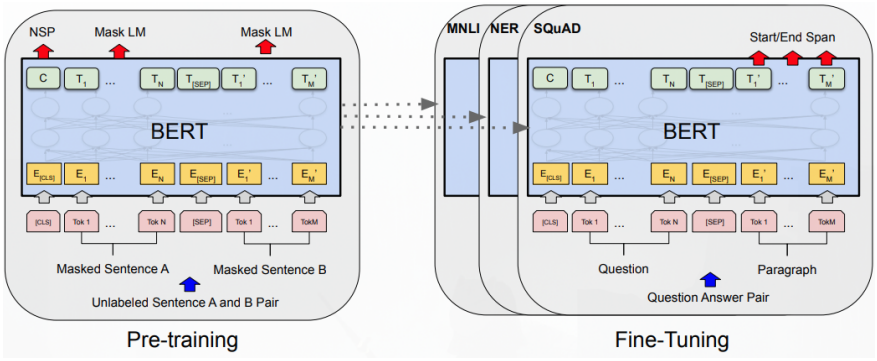

Figure 2.2: The overall pre-training and fine-tuning phases for BERT. For different down-stream tasks, notice the same architecture, except output layer, is used. Source: (Devlin et al. [17]).

The pre-training phase can be split into two further training tasks:

- Masked Language Model - During training, a portion of the input tokens are 'masked' and the model predicts these masked tokens based on the remaining context, using cross-entropy loss[70]. In the literature, this is referred to as a Cloze task [61]. In the experiments conducted by (Devlin et al. [17]) 15% of tokens in each sequence

are randomly selected for masking. However, to mitigate the mismatch between pre-training and fine-tuning phases, a training data generator replaces each chosen masked tokens with: (1) the special token `[MASK]` 80% of the time; (2) a random token 10% of the time; and (3) the original token itself 10% of the time.

- Next Sentence Prediction - In this task, the model is trained on a binary classification task prediction the relationship between sentence pairs. Given two sentences `A` and `B` either (1) sentence `B` is the actual sentence that follows `A` 50% of the time or (2) sentence `B` is randomly selected from the corpus 50% of the time.

The Google AI Language team further fine-tune BERT across 11 NLP tasks, showing that BERT out-performed the state-of-the-art models at its point in time.

As previously mentioned, a benefit of using PLMs such as BERT for NLP tasks (such as text embedding) is the ability to off-load the computationally intensive pre-training and any initial fine-tuning stages to another party. Although, this means that there now exists an implicit trust on that third-party's model parameters. For example, the third-party could be motivated factors to censor certain phrases, artificially injecting testing data into training to increase performance metrics or maliciously change the embedding space. These are all factors which influence our decision on using a PLM and, thus, PLMs with transparently documented data sources, pre-training and fine-tuning should be preferred.

XLNet and MPNet

Since the release of the Google AI Language team's BERT, there have been some developments on further improving the performance of PLMs based on BERT's architecture. Namely, we touch on two: XLNet [68] and MPNet [55].

BERT can be categorised as an auto-encoding (AE) language model. This is an

One downside of BERT, briefly touched on earlier, is the discrepancy between the pre-training and fine-tuning phases. Although BERT utilises bidirectional context for reconstruction of the special `[MASK]` token during pre-training, this special token does not exist at the fine-tuning phase. Additionally, the predicted tokens are masked in the *input* thus BERT assumes the predicted tokens are independent to the surrounding context [68]. XLNet, therefore, proposes

:

- Mathematics of BERT (detailed in XLNet paper)
- Mathematics of XLNet (detailed in XLNet paper)
- Autoregressive vs autoencoder
- Architectural changes in XLNet

Nomic

:

- Mention nomic is pitched as the first fully reproducable open-source, open-weights, open-data text embedding model
- Detail BERT archietcture adaptations
- Higher masking rate
- AdamW optimiser, learning rate with linear warmup.
- (weakly supervised) Pre-training: consistency filtering, curated long context text pairs, gradcache, mixed precision training.
- Prefixes on tasks: Symmetric category; asymmetric category.
- Supervised contrastive fine-tuning: datasets; learning rate. Randomly sampled mined negatives.
- talk about results, especially performance on long context

2.4 Dimensionality Reduction

:

- Talk why dimensionality reduction is needed: visualisation, curse of dimensionality, clustering? Figure out how to structure this well for the story.
- Talk about PCA and UMAP.

2.4.1 PCA

:

- Talk about PCA mathematics and why its used for dim reduction
- Talk about elbowing and finding the most optimal (?)

2.4.2 UMAP

:

- Talk about UMAP, the algorithm, the assumptions, the constraints
- Use the visualisation of the elephant here.

2.5 Clustering

:

- Talk about what clustering does, i.e. looking at trying to categorise or something
- Talk about the curse of dimensionality and why clustering does not perform well. Then talk about what people usually do (dimensionality reduction via feature selection or algorithms)
- Talk about k-Medoids (supervised), DBSCAN and HDBSCAN (unsupervised).

2.5.1 k-Medoids

:

- Talk about algorithm
- Talk about the parameters any effects when tuning these parameters.
- Talk about drawbacks and benefits of this approach, citations needed.

2.5.2 DBSCAN

:

- Talk about algorithm
- Talk about the parameters any effects when tuning these parameters.
- Talk about drawbacks and benefits of this approach, citations needed.

2.5.3 HDBSCAN

:

- Talk about algorithm
- Talk about the parameters any effects when tuning these parameters.
- Talk about drawbacks and benefits of this approach, citations needed.

2.6 Clustering Evaluation

:
Maybe this should be later on in the methodology?

- Motivate the need for clustering evaluation.
- Just highlight mathematics of each algorithm and mention the need for them.

2.6.1 Inertia

2.6.2 Silhouette

2.6.3 Davies-Bouldin Index

2.6.4 Calinski-Harabasz Index

2.7 Maybe Optuna

Chapter 3

Automatic Categorisation and Label Generation

3.1 Data

Data relevant to the operational state and maintenance of the ISIS facility originates from several sources. The facility publishes open-access datasets, primarily containing instrument calibrations and experimental metrics, via its data portal [25]. Alongside this, the ISIS team maintains proprietary internal datasets detailing specific component performance, which are available upon request. For this research, access was granted to historical logs for the ion source component, with entries dating back to March 27^{th}, 2003. Furthermore, the operations crew maintains a daily operational log, the 'Operalog', documenting facility faults and the corresponding remedial actions taken.

Given the breadth of available data and the project's scope constraints, a decision was made to focus the analysis presented in this paper on the **Operalog**. This dataset provides a rich textual record of real-world failures across the facility, offering valuable insights despite its partially unstructured nature. Data from the specialised ion source logs is the focus of the partner project.

3.1.1 The Operalog

The operational log, otherwise referred to as the 'Operalog', is an Excel spreadsheet with entries documenting moments of machinery failure within the ISIS facility covering the period 1996 - 2023. Table 3.1 documents the important features. Notably, the day-to-day operational crew have developed custom abbreviations, acronyms and terminology which may not immediately be clear. Additionally, different crew members have varying writing styles and levels of depth of information. Thus, this results in an unstructured text dataset which has a huge variance in quality and quantity of information. Furthermore, as noted in Table 3.1, `FaultRepair` was seemingly made redundant post-2017. Figure 3.1 highlights the distribution in the text lengths of both unstructured text fields. Furthermore, around 0.02% of `FaultDescription` fields are empty whereas around 66% of `FaultRepair` fields are empty. The generally shorter

entries in `FaultRepair`, with large majority having fewer than or equal to 5 characters, suggest lower informational content compared to `FaultDescription`. This is further illustrated in the Wordcloud illustrations [44] (Figure 3.2), where `FaultRepair` only has one extremely large word (`reset`). As the size of the word indicates the frequency, this means `FaultRepair` has the word 'reset' in almost all non-empty entries. With 'reset' being exactly 5 characters and most non-empty entries in `FaultRepair` being at most 5 characters (of which, those that contain reset are over 70% of entries), it is easy to see that the `FaultDescription` field contains richer, more informative context.

Feature Name	Data Type	Description
FaultDate	Date-time	Date the fault occurred, with a precision up to the nearest second.
UserRun	String	Operational cycle that this fault has occurred within. Cycle information up to 2016 can be found at (ISIS Neutron and Muon Source [24]).
Downtime	Integer	The amount of time, in hours the downtime has occurred for.
Group	String, Fixed Category	The group that the faulty equipment is part of. There are 13 unique equipment groups.
Equipment	String, Fixed Category	The equipment type that failed. There are around 200 unique equipment types that have been logged to fail.
FaultDescription	String, Free-form	This is a free-form, unstructured text field that allows the on-shift operational crew to note details about the problem diagnosis and remediation that has occurred. There is no constraint to the size of this text field.
FaultRepair	String, Free-form	Another free-form, unstructured text field that allows the on-shift operational crew to note only the remediation steps. The crew seems to have stopped using this field after the end of 2017, preferring to put remediation steps in FaultDescription.
ManagersComments	String, Free-form	A very rarely used free-form text field, where the manager in charge of the on-shift crew will input comments.

Table 3.1: Description of important features (columns) in the Operalog.

3.1.2 Preliminary Data Analysis

:

- Talk about amount of issues per year
- Talk about the distribution of text lengths across whole df
- Talk about the distribution of text lengths across iondf

3.2 Overview

Make a diagram that showcases the process (and all the hyperparameter choices)

The process of transforming an unlabelled, uninformative Operalog into an informed, induced-label tagged dataset can be broadly characterised into five major steps. These steps are: (1) initial pre-processing and normalisation of the `FaultDescription` natural language

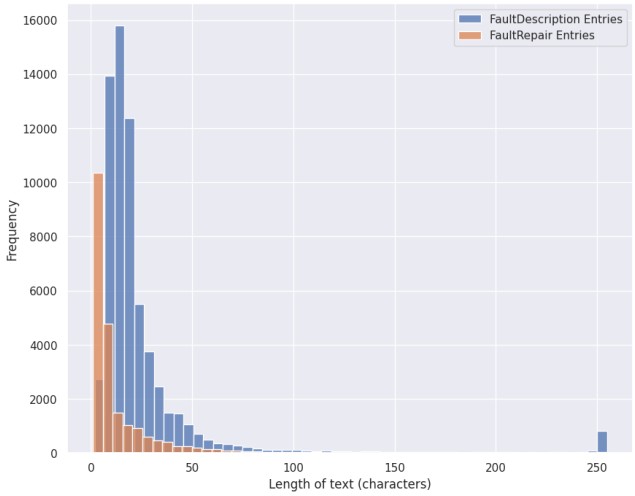

Figure 3.1: Frequency distribution of the text lengths for non-blank `FaultDescription` and `FaultRepair` fields. Each bar represents a range of 5 characters.

(a) FaultRepair **(b)** FaultDescription

Figure 3.2: Comparison of word clouds for the top 300 most common words for the FaultRepair field versus the FaultDescription field. Larger words are more common and collocations are enabled so both words and phrases are shown, leading to some repeating.

text; (2) embedding the cleaned text into a high-dimensional embedding space using a BERT-family transformer encoder model; (3) low-dimensional projection of the embedding space while respecting the higher-dimensional topological manifold; (4) performing clustering on the low-dimensional data, optimising clustering using clustering metrics as heuristics for the 'goodness' of a cluster; and (5) using natural language processing to induce label names for each cluster and tagging each entry with this label. Here, we choose to solely consider the `FaultDescription` field as the vast majority of `FaultRepair` fields are either empty or less than or equal to 5 characters long.

Several hyperparameters affect the performance of the label generating process. Examples include model or algorithm specific parameters (i.e. the dimensionality reduction or clustering algorithms) and clustering 'goodness' heuristics used. These hyperparameters are tuned using Optuna hyperparameter optimisation framework, an automatic optimisation framework specifically designed for machine learning applications [3] and also allows for multi-objective optimisation. The hyperparameter optimisation of the process then produces the 'best' induced categories and thus labels for each issue type.

Additionally, as this project is an early stage exploration of auto-categorisation for the ISIS Operalog, the major focus is on ion source `Equipment` type unless explicitly specified. This enables the project to focus on progressing the tooling in a scoped manner with a structured goal in mind - to progress the auto-categorisation effort on the ISIS Operalog.

The hardware available for research is detailed in Table 3.2. Some models, such as PLMs with a high parameter count, are unable to be loaded on the hardware available. Therefore, consideration of the limited computation resources played a major factor in the decisions regarding the pipeline.

Hardware	**Description**
CPU	11th Gen Intel i7-11800H (16 core) @ 4.600GHz
Main Memory	64Gb
GPU	NVIDIA GeForce RTX 3060 Mobile / Max-Q
GPU Memory	6Gb

Table 3.2: Research hardware description.

Clean up wording in the above

3.3 Natural Language Pre-processing

The aim of pre-processing the unstructured, natural language text data from `FaultDescription` is to normalise and prepare it from text embedding. Normalisation allows many syntactic permutations or augmentations of a text to be collapsed into one standardised form. This allows for consistent results as it ensures syntactically different but same-meaning texts will be mapped by embedding model to the same value. For pre-processing, and the rest of the pipeline, each log entry's `FaultDescription` is considered one text.

Clean up wording in the above

Tokenisation is one of the earliest stages of natural language processing. To perform tokenisation, we use Tensorflow's `UnicodeScriptTokenizer` [1]. Tensorflow is 'an end-to-end platform for machine learning', which provides efficient machine learning algorithm implementations. A tokeniser, such as this, splits the text into sub-units called 'tokens' [21]. These

tokens are useful to pass to subsequent natural language processing stages which is, in our case, text embedding. UnicodeScriptTokenizer, a specialised version of Tensorflow's `Tokenizer`, tokenises 'UTF-8 strings by splitting where there is a change in Unicode script' ([1], Tensorflow Text).

Once tokenised, stop-word removal is performed. Stop-words are non-informative words such as articles, prepositions and pronouns. Therefore removal of these words tends to help models have access to informative contexts and reduce noise [54]. We consider the standard English stop-words from NLTK [9] and some context specific stop-words. This includes 'ion source' which does not provide any informative contexts as we have already scoped our dataset to ion source `Equipment` and 'breakdown' which appears in just over 50% of entries.

As mentioned previously, there are a few non-trivial abbreviations used by the operational crew when writing a log entry. These abbreviations are normalised into a standard English word or phrase (see Table 3.3).

Furthermore, punctuation tokens are converted into one of two categories: end-of-sentence (`EOS`) tokens or regular punctuation (`PUNC`) tokens, which are self-explanatory. The goal for this is to normalise punctuation to either signify sentence boundaries or collapse into one token. The majority of the `FaultDescription` text does not typically utilise punctuation to convey additional meaning. However, this is not trivial to see and thus this part of the process has been made into a parameter through the usage of a flag which decides whether punctuation mapping is enabled. Text casing normalisation follows the same procedure.

Just in terms of ion source `Equipment`, the pre-processing is applied across more than 1200 log entries, ranging from 2009 - 2023. Tokenisation and operating on tokens is computational intensive, thus the solution operates on tensors, leveraging Tensorflow capabilities to perform GPU-based compute, which speeds up machine learning based applications [1, 8].

As text pre-processing is intensive to compute, when performing hyperparameter optimisation (more on this in Section 3.7) attempting to use these flags as part of the optimisation process resulting in re-computation of this pre-processing is an inefficient use of computational resources. Additionally, as the flags are boolean, it is relatively easy to manually enumerate each configuration option. Therefore, these parameters are exposed through the CLI application (see Section 3.9), rather than the hyperparameter optimisation process.

Clean up wording in the above

Table 3.3: Operational crew abbreviation mapping.

Abbreviation Regular Expression	Mapped word or phrase
o/p	output
i/s	ion source
(b/down | break-down | b\down | b/d)	breakdown

: TODO Talk abuot

- Tokenizer - DONE
- Stopwords - DONE
- TOKEN mapping (PUNC, EOS) - DONE
- Casing - DONE
- Shortcut mapping - DONE
- a word on using tensorflow tensors (GPU mapping) - DONE

3.4 Establishing an embedding space

After pre-processing, the cleaned and normalised text is ready to be digested by a sentence embedding model (see Section 2.3). This is motivated by the need to understand syntactic and semantic relationships between each label before any categorisation can take place. The categorisation process itself can be viewed as a more sophisticated task of understanding and ranking the relationship between various `FaultDescriptions`, grouping together entries which appear the closest in meaning.

In interest of time, and motivated by the literature suggesting the benefits of the current state-of-the-art models, this project only considers the PLMs listed in Table 3.4 (see Section 2.3.1 for further detail on the models). The table describes the model architecture and specific model parameter configuration used (for example, different authors who fine-tune to different tasks will have varying model parameter configurations).

Choosing the 'right' embedding model for our task is extremely paramount. Similarly to choosing the pre-processing steps, choosing an embedding model affects the subsequent pipeline. If the embedding space is poorly constructed by during the pre-training or noise is introduced that affects clustering performance, the end result will change quite drastically. This is a key point to note going forward.

By using a pre-trained and pre-fine-tuned model we can have some certainty about performance on our specific task. However, due to the aforementioned computational resource bandwidth, choices in models is somewhat restricted - mainly bottlenecked by the amount of GPU memory (6 Gb, in our case). With this in mind, we specifically look at models that perform well for clustering on the Massive Text Embedding Benchmark (MTEB) leaderboard and have low computational usage requirements. The MTEB leaderboard compares more than 100 text and image embedding models across 132 tasks of 9 categories, 17 of which are clustering tasks [40]. Nomic has been shown to compete with models around 70 times its size (that rank in the top 10), ranking in the top 50 and achieving an average clustering score of around 43.9 in clustering (`nomic-ai/nomic-embed-text-v1.5`)[43]. However, with MPNet being an older model, it falls slightly behind, ranking at 137 yet achieves an average of around 43.7 in clustering (`sentence-transformers/all-mpnet-base-v2`).

Fix wording in the above

Model	PLM Model Path	Description
MPNet	`sentence-transformers/all-mpnet-base-v2`	This model is a fine-tuned version of the pre-trained MPNet base model. It was fine-tuned on a sentence pairs dataset that has around one billion entries using a contrastive learning objective (i.e. given a sentence from a pair, the model should predict which it was paired with out of randomly sample set of sentences).
Nomic	`nomic-ai/nomic-embed-text-v1.5`	This model is a slight upgrade from v1, which is the paper detailed in Section 2.3.1. Nomic Text v1.5 outperforms v1 by using Matryoshka representation learning which trains the model to learn nested representations at different embedding dimensions [29].

Table 3.4: PLMs considered for this project, the HuggingFace [67] model path and its feature description.)

:

- Motivate the need for sentence embedding - DONE
- Talk about the two types of sentence embedding (PLM, MPNet and Nomic) - DONE
- Talk about why we do not consider more embedding models - DONE
- Highlight that choosing this is critical as it affects the rest of the pipeline - DONE

3.5 Embedding dimensionality reduction

:

- motivate this with the curse of dimensionality and state the embedding spaces of the models
- Here, we put some effort into visualisation of the embedding dimension with MPNET and with Nomic. Show images
- Note that only unique sentences are considered for embedding (we have a one to one mapping backwards anyways).
- In literature, there exists PCA and t-SNE and UMAP. Explain why we did not choose t-SNE and stuck with comparing UMAP and PCA.
- Reason why UMAP is fine to use here, the topological manifold assumption (hard).
- Talk about selecting UMAP and why over PCA.
- Talk about UMAP parameters and how changing them affects things.
- Talk about UMAP librarys and just lightly motivate this as an optimal implementation.

3.6 Unsupervised categorisation through clustering

3.7 Hyperparameter optimisation

3.8 Label generation

3.9 CLI Application

:

- Okay so we have done hyper parameter tuning etc but the real benefit of this tool is that it gives you a qualitatively assessable result in an easy to run format.

: Describe the methods and procedures used. The things in this section will include.

- Explaining data format and data visualisation: wordcloud.
- Data cleaning steps, including removing key words such as Ion Source.
- Text preprocessing steps (cleaning) and computational challenges (tensorflow).
- Choosing the best sentence embedding transformer: MPNET, NOMIC.
- Data visualisation (before and after sentence embedding): similarity visualisation, explain unique sentences, token length distribution.
- Motivate why clustering in higher dimensions performs worse
- UMAP, PCA, t-SNE comparison. Motivate using UMAP.
- UMAP hyperparameter optimisation.
- Performing clustering with kmedoids, dbscan, hdbscan.
- Using optuna.
- Evaluation of results and choosing the best model (and arguing why hdbscan is the best by looking at the variance of dbscan and inflexibility of kmedoids)
- Touch on the production of a CLI application that allows you to mix and match various parts of the pipeline. Motivate the need for command line tool.

Chapter 4

Results and Discussion

: Describe the results and analyse the results

- Analyse the word cloud.
- Analyse the sentence embedding results.
- Analyse UMAP vs. PCA vs. t-SNE qualitatively and later quantitatively (compared to the clustering).
- Anaylse the UMAP hyperparameter optimisation qualitatively, mention that we use Optuna.
- Next steps: Fine tuning the PLM (Nomic or MPNet). Using MOE nomic (v2). Rent higher computation, then you can use top performing PLMS.

Chapter 5

Conclusion

:

- Definitely talk about clear and transparent PLMs and malicious stuff (see BERT section).

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[2] PY Abijith, Piyush Patidar, Gaurav Nair, and Rohan Pandya. Large language models trained on equipment maintenance text. In *Abu Dhabi International Petroleum Exhibition and Conference*, page D021S065R003. SPE, 2023.

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[4] Ramiz M Aliguliyev. A new sentence similarity measure and sentence based extractive technique for automatic text summarization. *Expert Systems with Applications*, 36(4): 7764–7772, 2009.

[5] C Ankenbrandt, C Curtis, C Hojvat, RP Johnson, C Owen, C Schmidt, L Teng, and RC Webber. H- charge exchange injection systems. In *11th International Conference on High-Energy Accelerators: Geneva, Switzerland, July 7–11, 1980*, pages 260–271. Springer, 1980.

[6] Asim Yaqoob. First line diagnosis at ISIS, Oct 2017. URL https://indico.cern.ch/event/558933/contributions/2724364. Last visited 2025-04-15.

[7] Kensuke Baba, Tetsuya Nakatoh, and Toshiro Minami. Plagiarism detection using document similarity based on distributed representation. *Procedia computer science*, 111: 382–387, 2017.

[8] Ioana Baldini, Stephen J Fink, and Erik Altman. Predicting gpu performance from cpu runs using machine learning. In *2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing*, pages 254–261. IEEE, 2014.

[9] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[10] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[11] Alexander Budanitsky and Graeme Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and other lexical resources*, volume 2, pages 2–2, 2001.

[12] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[13] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[14] Hongliu Cao. Recent advances in text embedding: A comprehensive review of top-performing methods on the mteb benchmark. *arXiv preprint arXiv:2406.01607*, 2024.

[15] Thyago P Carvalho, Fabrízzio AAMN Soares, Roberto Vita, Roberto da P Francisco, João P Basto, and Symone GS Alcalá. A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137: 106024, 2019.

[16] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[18] Sergey Edunov, Alexei Baevski, and Michael Auli. Pre-trained language model representations for language generation. *arXiv preprint arXiv:1903.09722*, 2019.

[19] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[20] Maurizio Faccio, Alessandro Persona, Fabio Sgarbossa, and Giorgia Zanin. Industrial maintenance policy development: A quantitative framework. *International Journal of Production Economics*, 147:85–93, 2014.

[21] Gregory Grefenstette. Tokenization. In *Syntactic wordclass tagging*, pages 117–133. Springer, 1999.

[22] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[23] ISIS Neutron and Muon Source. A Practical Guide to the ISIS neutron and Muon source, Apr 2021. URL https://www.isis.stfc.ac.uk/Pages/News21_PracticalGuide.aspx. Last visited 2025-04-15.

[24] ISIS Neutron and Muon Source. ISIS beam operations, Apr 2024. URL https://www.isis.stfc.ac.uk/Pages/beam-status.aspx. Last visited 2025-04-15.

[25] ISIS Neutron and Muon Source. ISIS data gateway, 2025. URL https://data.isis.stfc.ac.uk/datagateway. Last visited 2025-04-16.

[26] Ali Jezzini, Mohammad Ayache, Lina Elkhansa, Bassem Makki, and Maya Zein. Effects of predictive maintenance (pdm), proactive maintenace (pom) & preventive maintenance (pm) on minimizing the faults in medical instruments. In *2013 2nd International conference on advances in biomedical engineering*, pages 53–56. IEEE, 2013.

[27] Taeho Jo. K nearest neighbor for text summarization using feature similarity. In *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, pages 1–5. IEEE, 2017.

[28] Sushil Kumar and Komal Kumar Bhatia. Semantic similarity and text summarization based novelty detection. *SN Applied Sciences*, 2(3):332, 2020.

[29] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.

[30] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[31] Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, et al. Gecko: Versatile text embeddings distilled from large language models. *arXiv preprint arXiv:2403.20327*, 2024.

[32] Hongfei Li, Dhaivat Parikh, Qing He, Buyue Qian, Zhiguo Li, Dongping Fang, and Arun Hampapur. Improving rail network velocity: A machine learning approach to predictive maintenance. *Transportation Research Part C: Emerging Technologies*, 45:17–26, 2014.

[33] Hong Liang, Xiao Sun, Yunlei Sun, and Yuan Gao. Text feature extraction based on deep learning: a review. *EURASIP journal on wireless communications and networking*, 2017:1–12, 2017.

[34] Romans Lukashenko, Vita Graudina, and Janis Grundspenkis. Computer-based plagiarism detection methods and tools: an overview. In *Proceedings of the 2007 international conference on Computer systems and technologies*, pages 1–6, 2007.

[35] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[36] HT Michael. Electronic circuits: fundamentals and applications, 2006.

[37] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[38] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40, 2023.

[39] R Keith Mobley. *An introduction to predictive maintenance*. Elsevier, 2002.

[40] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

[41] Giancarlo Nota, Alberto Postiglione, and Rosario Carvello. Text mining techniques for the management of predictive maintenance. *Procedia Computer Science*, 200:778–792, 2022.

[42] Zach Nussbaum, John X Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*, 2024.

[43] Nussbaum, Zach and Cembalest, Max. Nomic Embed's Surprisingly Good MTEB Arena Elo Score, Aug 2024. URL https://www.nomic.ai/blog/posts/evaluating-embedding-models. Last visited 2025-04-18.

[44] Layla Oesper, Daniele Merico, Ruth Isserlin, and Gary D Bader. Wordcloud: a cytoscape plugin to create a visual semantic summary of networks. *Source code for biology and medicine*, 6(1):7, 2011.

[45] Zhaotai Pan, Yi Ge, Yu Chen Zhou, Jing Chang Huang, Yu Ling Zheng, Ning Zhang, Xiao Xing Liang, Peng Gao, Guan Qun Zhang, Qingyan Wang, et al. Cognitive acoustic analytics service for internet of things. In *2017 IEEE International Conference on Cognitive Computing (ICCC)*, pages 96–103. IEEE, 2017.

[46] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11): 559–572, 1901.

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

[48] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[49] T Nora Raju, PA Rahana, Raichel Moncy, Sreedarsana Ajay, and Sindhya K Nambiar. Sentence similarity-a state of art approaches. In *2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS)*, pages 1–6. IEEE, 2022.

[50] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[51] Mark Sammons, Christos Christodoulopoulos, Parisa Kordjamshidi, Daniel Khashabi, Vivek Srikumar, and Dan Roth. Edison: Feature extraction for nlp, simplified. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4085–4092, 2016.

[52] Justyna Sarzynska-Wawer, Aleksander Wawer, Aleksandra Pawlak, Julia Szymanowska, Izabela Stefaniak, Michal Jarkiewicz, and Lukasz Okruszek. Detecting formal thought disorder by deep contextualized word representations. *Psychiatry Research*, 304:114135, 2021.

[53] BP Sharma. Nuclear reactors: Moderator and reflector materials. *Encyclopedia of Materials: Science and Technology*, pages 6365–6369, 2001.

[54] Catarina Silva and Bernardete Ribeiro. The importance of stop word removal on recall values in text categorization. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1661–1666. IEEE, 2003.

[55] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867, 2020.

[56] Josef Steinberger and Karel Ježek. Text summarization and singular value decomposition. In *Advances in Information Systems: Third International Conference, ADVIS 2004, Izmir, Turkey, October 20-22, 2004. Proceedings 3*, pages 245–254. Springer, 2005.

[57] Chuan-Jun Su and Shi-Feng Huang. Real-time big data analytics for hard disk drive predictive maintenance. *Computers & Electrical Engineering*, 71:93–101, 2018.

[58] Gian Antonio Susto and Alessandro Beghi. Dealing with time-series data in predictive maintenance problems. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4. IEEE, 2016.

[59] Gian Antonio Susto, Alessandro Beghi, and Cristina De Luca. A predictive maintenance system for epitaxy processes based on filtering and prediction techniques. *IEEE Transactions on Semiconductor Manufacturing*, 25(4):638–649, 2012.

[60] Ayisha Tabassum and Rajendra R Patil. A survey on text pre-processing & feature extraction techniques in natural language processing. *International Research Journal of Engineering and Technology (IRJET)*, 7(06):4864–4867, 2020.

[61] Wilson L Taylor. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433, 1953.

[62] JWG Thomason. The isis spallation neutron and muon source—the first thirty-three years. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 917:61–67, 2019.

[63] Juan Pablo Usuga-Cadavid, Samir Lamouri, Bernard Grabot, and Arnaud Fortin. Using deep learning to value free-form text data for predictive maintenance. *International Journal of Production Research*, 60(14):4548–4575, 2022.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[65] Geert Waeyenbergh and Liliane Pintelon. A framework for maintenance concept development. *International journal of production economics*, 77(3):299–313, 2002.

[66] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*, 2024.

[67] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[68] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[69] Minghe Yu, Guoliang Li, Dong Deng, and Jianhua Feng. String similarity search and join: a survey. *Frontiers of Computer Science*, 10:399–417, 2016.

[70] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018.