EXPLORING LANGUAGE MODELLING FOR NHS PATIENT SAFETY INCIDENT REPORTS

Niall Taylor
University of Oxford
Department of Psychiatry
`niall.taylor@st-hughs.ox.ac.uk`

Dan Schofield
NHS England
Transformation Directorate
`daniel.schofield1@nhs.net`

ABSTRACT

Transformer based language models are dominant in the current Natural Language Processing (NLP) literature, with pretrained language models (PLM) in particular becoming the standard backbone of many deep learning pipelines and applications. Whilst open source PLMs have shown amazing results in a number of domains, their ability on specialist text domains often decreases unless some form of domain adaptation or fine-tuning is undertaken. Once a PLM has been adapted to a new domain, investigation can refocus on how to assess its new embedding space compared to the base version, and how different pretraining objectives influence this structure.

The NHS collects free-text documents related to all types of patient safety incidents from National Reporting and Learning System (NRLS) and soon the Learn From Patient Safety Events (LFPSE) service. We present an exploration of a selection of different language model pretraining and fine-tuning objectives with these patient safety incident reports as the domain of interest, followed by discussion of a number of methods for probing and evaluating these new models, and their respective embedding spaces.

This work was completed as part of an NHS England Data Science PhD Internship project.

1 Introduction

Recent advances in deep learning architectures have led to a dominance in the use of transformer based architectures across a number of fields, including Natural Language Processing (NLP). Transformer based language models have become staples of many state of the art applications and the training and the open release of large Pretrained Language Models (PLMs) such as the Bidirectional Encoder Representation Transformer (BERT) and Generative Pretrained Transformers (GPT) families has changed the research landscape entirely [1–3]. Modern PLMs are trained on huge amounts of open free text data with the goal of *modelling* human language and producing a good, contextual representation that can capture grammatical and semantic information in a broad sense. Whilst the actual ability of these models to *understand* human language is questionable, their ability on a wide range of domains and tasks is certainly impressive.

A common problem that these open source models face is the ability to perform to the same level on new data, or data that is starkly different to that it was trained on. Even when the native language used is the same, i.e. English, the vocabulary and grammar used in different domains can differ dramatically and this can prove difficult for many open source pretrained models. This is especially true when the text is from a niche domain, and further not able to be openly shared due to its sensitivity, as is understandably the case with much of free text in the NHS.

NHS England maintains a national repository of incident reporting data (NRLS) for England and Wales. These data are used to support many patient safety initiatives, for instances, the detection of emerging patient safety risks, and also drive learning that improves patient safety. Although there are a number of categorical data fields, many of the ‘signals’ in these data are only present within the free-text fields [4]. The volume of these data provide a unique opportunity to apply the most recent advances in NLP and language modelling to produce a sizable and reusable PLM for this NHS domain.

A key objective in the collection of incident reports is to drive proactive change. Clinically experienced teams review the most serious events and identify emerging themes, but the scale of the dataset means that completing a full review of all incidents would be unfeasible. Natural Language Processing (NLP) techniques have the potential to help unlock learning from data that do not receive a full clinical review. The dataset is currently being used for topic modelling and other analyses, but a large scale language model is not yet in common use. The creation of an appropriate language model would allow for the use of recent clustering and topic modelling techniques to help identify novel targets for clinical review.

This work has an accompanying repository under the name **ELM4PSIR** (Exploring Language Modelling for (NHS) Patient Safety Incident Reports) which is available on GitHub at `https://github.com/nhsx/ELM4PSIR` and contains various pipelines to reproduce the results contained below.

2 Background

2.1 Language Modelling

Language models are a family of different architectures and algorithms with varying complexity and uses. Generally speaking, a language model is a probability distribution over a sequence of words, such that one can assign the probability a certain sequence of words would appear in a sentence or a corpus. The history of language modelling is decades old now, with the term becoming a little more abstract as model complexity increases and new language modelling objectives are introduced, especially with large non-linear neural networks coming into play. A journey through this long history and the many innovations of language modelling is beyond the scope of this report, and the focus will be on the most recent flurry of innovations in the space, namely the advent of transformer based language models and the NLP models that stem from them. A dominant trend in NLP has been the training and release of large PLMs for reuse by others trained on enormous amounts of text.

Pretrained Language Models

PLMs from the transformer based families are produced via training on huge amounts of free text data through language modelling objectives such as Masked Language Modeling (MLM) and next word prediction. Once pretrained, the model and its trained weights are supposed to have generalisable capabilities given written text. With these PLMs one can fine-tune on new domains and design downstream tasks with relative ease, often resulting in state of the art results on a number of popular benchmark datasets and tasks [3, 1, 5]. One common finding, especially in the smaller (relatively speaking) sized PLMs is a drop in performance when given text data from a domain different to that the model was originally trained on [6, 7].

For example, a particularly difficult text domain for many open sourced PLMs is the biomedical domain and any specialist medical text. This is often thought to be due to the complexity of medical language, and the large variance between countries in how clinical text is formulated, etc. There are several techniques to mitigate this problem which typically rely upon a form of transfer learning or domain adaption whereby the model is trained on the new domain. One typical approach is to continue training the PLM in this specialist domain using the same language modelling objective, to better prepare the model for its new domain, which has seen promising results [6–8]. Given enough data and computational resources, some may suggest training a Language Model (LM) from scratch, i.e. train a LM only on this specialist domain and produce a domain specific PLM [9]. Although a possible approach to consider, it is a timely and expensive endeavour and the model itself may lack generalisability to other text domains. Regardless of the approach, there still exists a benefit and often requirement of training a LM on the domain of interest.

There is good reason to believe that the written language used within the NHS may differ dramatically from that used in large open-source text data of which most, if not all, popular open-source PLMs are trained. Whilst numerous biomedical or clinically trained PLMs exist, none exist for UK based NHS language and most are United States based which can suffer due to differences in medical terminology, etc. We believe there is a great opportunity to develop and train a NHS specialist LM to enhance work with the NHS NRLS dataset.

2.2 Other Neural Language Models

There are a number of deep learning NLP architectures that can be used for the current project, most of which came prior to transformers. We will not be including an extensive report on these approaches, however they provide useful baselines to compare with the transformer based models on certain tasks. Therefore we will be including results where appropriate. We will be using two baseline models which can be seen as representing chronological progressions leading up-to transformer based architecture, Bag-of-Words and Word2Vec [10].

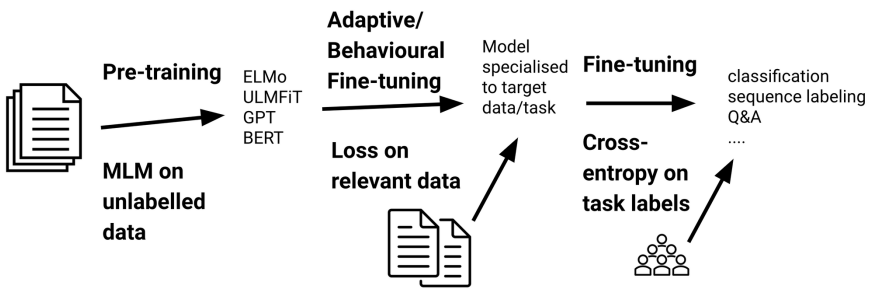


Figure 1: Overview of pretraining and domain adaptation of transformer based language models, taken from: Recent Advances in Language Model Fine-tuning.

2.3 Embeddings and Topic Modelling

The ultimate product of neural network based language models is word and potentially sentence level embeddings, which are essentially n-dimensional vector representations. These trained embeddings now hold the necessary information for representing human language and form the features for any subsequent downstream task.

PLMs such as the BERT and GPT families power rests in the fact they produce a mixture of embedding types: token, segment and positional embeddings combine with self-attention calculations to output *contextualised* embeddings for each word [1–3]. These *contextualised* embeddings are supposed to contain information about the position of the token and an attention weighting to other tokens in the sequence, meaning that the resultant embedding for a unique token in the vocabulary would be different in different sentences or *contexts*. On the other hand embeddings produced by earlier models such as Word2Vec and GloVe [10, 11] would be more static whereby each unique token in the vocabulary was assigned one embedding vector per token regardless of the sentence or context. Fig. 2 illustrates the passage of words (tokens) through the transformer architecture.

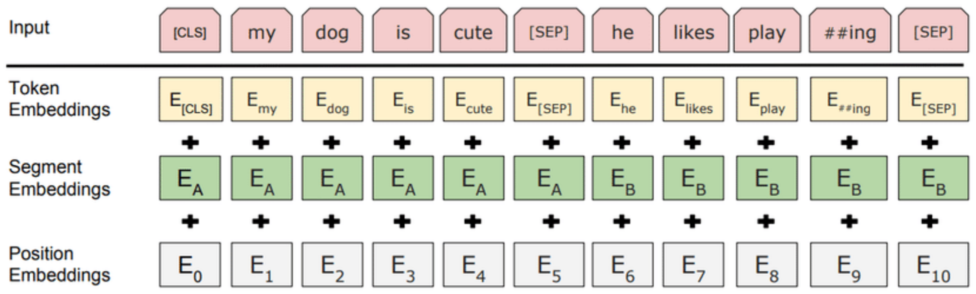

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2: Information flow for transformer based embeddings. Taken from: The Illustrated Transformer

Vanilla language modelling objectives work on a word or token level, with the aim of representing words based on the context they appear in. For example the MLM objective is operating on a token level loss objective. Essentially an embedding is produced per token of the input sequence. The resultant token level embeddings appear quite good for the task at hand, but many downstream tasks do not operate on the token level, but rather the sentence or entire document level e.g. document sentiment analysis, of which PLMs are consistently state of the art. From a PLM such as BERT a document level representation can be achieved in a number of ways based on the token level embeddings produced, with a common method being mean pooling i.e. the average of all token embeddings [1].

A potential issue with this relatively crude approximation of a document level embedding is that the LM pretraining objective did not encourage these representations to be useful or have any relationship to the loss function directly. A number of approaches have been taken to better train these LMs to produce sentence level embeddings through what we call alignment tasks [12–14]. Alignment tasks, such as contrastive learning, aim to cluster embeddings from sentences or documents that are of the same type or even from the same global document closer together than those that are not. The loss calculated (in theory) encourages the model to produce token embeddings that when averaged together produce a more meaningful embedding space. This project will explore the influence of different LM pretraining on subsequent downstream tasks that require document level embeddings.

Of particular interest to this project was the ability to cluster sentences or whole documents using these embeddings with the aim of producing meaningful topics or themed clusters of documents for later inspection. A major use of patient safety incident reports is to aid the discovery of emerging patient safety concerns and address them accordingly. This can involve manually reading a guided sample of these incident reports, which considering their volume, is a very time consuming and highly non-trivial task.

3 Datasets

3.1 NHS Patient Safety Incident Reports

Our corpus of interest is taken from nationally collected free-text documents relating to all types of patient safety incidents in the NHS from the National Reporting and Learning System (NRLS) and soon the Learn From Patient Safety Events (LFPSE) service, see the official website - National Patient Safety Reports. We obtained a sub-sample of approximately 2.3 million de-identified reports produced in the financial year 2019/2020.

Data splits

Of the $\sim$2.3 million documents provided in the corpus, 10% of this was held out from any of the work at the start to provide an untouched set for any final testing required. Of the remaining 90%, a 90:10 train:test split was performed for the language modelling and downstream tasks.

Further, it is important to note that whilst we had access to these $\sim$2.3 million documents, we developed and trained pipelines at various scales. We then chose to train and test models on a maximum sub-sample of 250,000 documents as this we felt this should be large enough to show any trends in the data, whilst keeping resource utilisation to a low level. Future work could seek to train with a larger amount of data. This could be further extended by changing the corpus sample such that it ranges multiple years, to both increase coverage of the training corpus and explore different longitudinal aspects of the wider corpus.

Length of documents

An important feature of a text dataset is the number of words, or tokens, contained in each individual sample or document. Transformer based language models such as RoBERTa can only handle a maximum of 512 tokens on even the most modern GPUs, due to the complexity of the attention calculations which increases exponentially with the number of tokens, see [15]. The nature of the patient safety reports means they are in general of a short length, with very few documents in our samples consisting of greater than 512 tokens. This ultimately allows us to use any transformer based model and be confident we are not missing any information contained within a document.

3.2 Pseudo Classification Tasks

As we were keen to explore models without focusing on a specific class of downstream task, we opted to derive *pseudo* classification tasks given the categorical variables provided alongside our cut of the NRLS data. The range of categorical variables pertaining to these data is large, and we opted to focus on two distinct categories: incident severity and incident type.

We make no claims to the clinical validity or utility of these pseudo tasks, and in fact want to overstate that these tasks were merely a means to compare and contrast different models on a defined task. The motivation was to highlight how different NLP models, trained with varying objectives and data produce embeddings that can be seen as holding semantic and structural information of a given text dataset. The reason we refer to these tasks as *pseudo* is because in each case we created a sub-sample of the training data with a balanced set of class labels, not representing the true distribution of the original data. The data used to create these classification datasets was sampled from the whole training data subset described above, and not the 250k sub-sample alone. This means that the training and testing for the downstream tasks is not tied to the same data as the language modelling.

This balancing approach is taken as often large imbalance in classification tasks can be problematic for large neural networks as they can often over-fit to the majority class and the loss function can fall into a pseudo-minima where predicting the majority class only gives good enough results as far as an evaluation metric like accuracy is concerned.

IN05 Level 1 - Incident Category

The labels of incident category at the first level are a standard field provided alongside each incident report which is used to detail the type of the incident. There are 13 classes present in the dataset, and we have not included level 2 which is included in the wider NRLS dataset.

PD09 - Incident Degree of Harm

The degree of incident severity categories are an ordinal scale ranging from no harm (1) to death (5) collected for all incidents. We simplify the pseudo task related to incidence severity prediction in the following way: the 1-5 incident severity labels given with the free text reports are skewed with the vast majority being attributed a 1 (or no harm), we create a much smaller and balanced binary classification dataset which bins incidence labels into low (severity categories 1-3) and high (severity categories 4-5) severity.

The main purpose of this *pseudo* task is to provide an intermediate evaluation metric for the various PLMs we trained and not to show performance in a production setting.

Table 1: Pseudo classification task descriptions. Sample size refers to a balanced class dataset

Task	Number of labels	Distribution note	Sample size for training
IN05: Category	13	∼50% data made up by top 3 classes, with remaining distributed amongst the rest	26,000
PD09: Severity	6 mapped to 2	∼90% originally in low severity class	14,000

3.3 Baseline Benchmark Dataset

To provide a route to compare domain adapted and fine-tuned models to a known baseline performance, we utilised the commonly used the General Language Understanding Evaluation (GLUE) dataset [16]. The motivation for testing all models against this benchmark was to determine how training models on the incident report data to varying degrees affects the end models performance on a much more general domain text task. Recall that most of the models reported in this paper were initialized from an open source, general domain PLM such as RoBERTa [17]. This can be considered as a way of probing for problems like *catastrophic forgetting* which has been seen when training or fine-tuning more general models on a specific domain, and over-fitting causes a significant performance drop on tasks where the original model was high performing.

4 Methodology

4.1 Language modelling

The choice of language model architecture was driven by current interest in downstream tasks, namely topic modelling, which will be more reliant on document level embeddings after the language model has been trained. For this reason we opted to use the RoBERTa [17] PLM as our starting model. RoBERTa is variation on a BERT model where the pretraining setup has been tweaked and it has undergone optimisation to improve on the original choice of training hyperparameters. It has been shown to outperform BERT in some settings, and is faster to train.

4.1.1 Tokenization

Before text can be passed to any model it needs to be converted into numerical representations, which in this setting are embedding vectors of fixed size H. But even before creating these vectors for each word, tokenization is the process of creating a vocabulary based on the training corpus and assigning a distinct identifier to each unique word or token. The conversion of unique words into tokens is called tokenization and different algorithms exist with varying levels of granularity. One important point is that tokenization is typically done once on the training data and hence why very large datasets are used to try and capture as many words as possible. After a tokenizer is *trained*, the vocabulary is ultimately fixed and any words that have not been encountered by the tokenizer before will be assigned a token ID representing *unknown* (and be aware that all unknown words will be given the same token ID). Therefore, this tokenization process is non-trivial and the tokenization algorithm is an important decision.

The simplest form of tokenization is perhaps character level where each unique character in the alphabet of the language is used. But generally for language modelling objectives it is better to have tokens representing words or word pieces.

A whole word tokenizer will find all unique words split by white-space and assign each a unique label and is perhaps the most intuitive, but has certain drawbacks. In particular this can lead to a poor representation of common sub-word meanings and a greater number of unknown tokens. An in-depth review of tokenization algorithms is beyond the scope of this report, but for clarity RoBERTa utilises a Byte-Pair Encoding (BPE) which is a hybrid between character and word level representations which all but removes the possibility of unknown tokens, as unknown words are split iteratively until known sub-word pieces are found[17].

4.1.2 Pre-training objectives

Masked Language Modelling

The language modelling objective of RoBERTa is known as Masked Language Modelling (MLM) whereby a random sample of the input is *masked* or replaced by a special $[MASK]$ token. This essentially corrupts the original input and the labels as the untouched input ids, with cross entropy loss calculated on these masked token positions. In simpler terms, the loss is based on the models prediction of the token for the *masked* positions, a form of gap filling. A crucial change with RoBERTa was the introduction of dynamic masking whereby each training sequence provided a percentage of the sequence which is randomly masked during the batching process, as opposed to the static masking in the original BERT paper which saw the masked datasets created prior to training commencing.

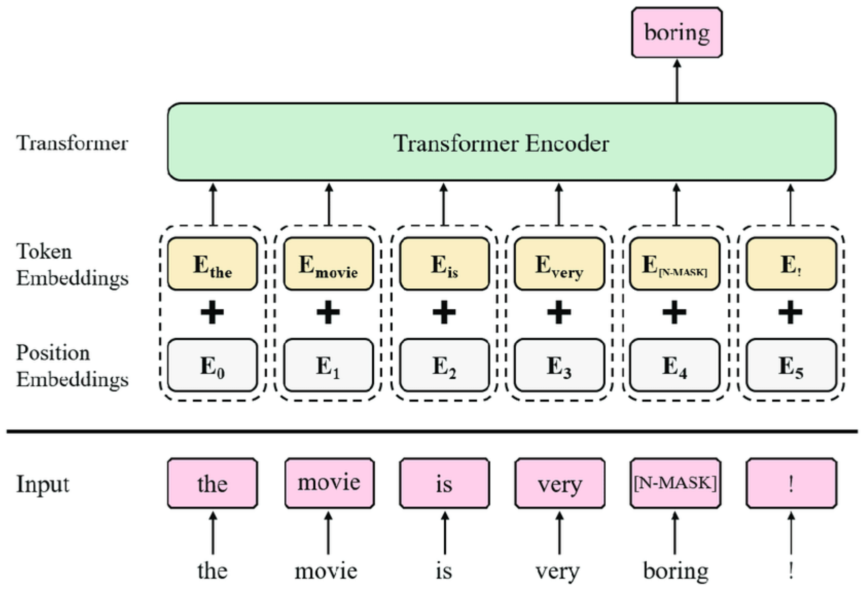


Figure 3: Overview of Masked Language Modelling objective used by RoBERTa and other BERT based models. The $N - MASK$ token replaced the actual world in the input and the objective is to correctly predict the word at that masked position. Image taken from [18].

Contrastive Loss

As we want to generate a model that can produce good document level embeddings, we explored an self-supervised cluster aligment technique used to produce the DeCLUTR model [13]. DeCLUTR stands for Deep Contrastive Learning for Unsupervised Textual Representations, and uses a contrastive loss function to encourage sentence or document level embeddings that are taken from the same document type or class to be closer together in the learned embedding space. DeCLUTR utilises a self-supervised contrastive loss function called InfoNCE (Noise-Contrastive Estimation) which aims to identify positive pairs in a set of samples also containing numerous negatives.

During training, a batch of N anchor-positive span pairs is taken. Each of the spans are separately encoded. The anchor and positive embeddings, z_i^T and z_i^I, are then compared for their cosine similarity by taking their dot product. The objective of the training procedure is to maximise the cosine similarity of the N matching anchor-positive pairs and minimise that of the remaining $N^2 - N$ non-matching pairs. For a given batch, the cosine similarities are then used to calculate the probability that a given pair is a match, which can be defined as:

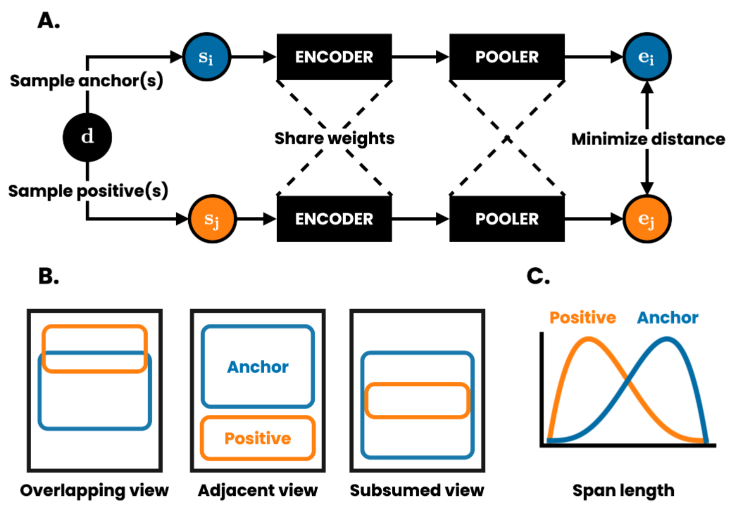


Figure 4: Taken from [13]. Overview of DeCLUTR sampling methodology and of the self-supervised contrastive objective. **(A)** For each document d in a minibatch of size N, we sample A_N anchor spans per document and P positive spans per anchor. For simplicity, we illustrate the case where $A_N = P = 1$ and denote the anchor-positive span pair as s_i, s_j . Both spans are fed through the same encoder $f(\cdot)$ and pooler $g(\cdot)$ to produce the corresponding embeddings $e_i = g(f(s_i)), e_j = g(f(s_j))$. The encoder and pooler are trained to minimize the distance between embeddings via a contrastive prediction task, where the other embeddings in a minibatch are treated as negatives (omitted here for simplicity). **(B)** Positive spans can overlap with, be adjacent to or be subsumed by the sampled anchor span. **(C)** The length of anchors and positives are randomly sampled from beta distributions, skewed toward longer and shorter spans, respectively.

$$P(z_i^I, z_i^T; \tau) = \frac{exp(z_i^I \cdot z_j^T/\tau)}{\sum_{i=0}^{N} exp(z_i^I \cdot z_k^T/\tau)} \tag{1}$$

where τ is a trainable temperature parameter. This results in the InfoNCE loss which symmetrically measures the success in maximising the similarity of matches and minimising the similarity of non-matches, defined as:

$$L_{\text{InfoNCE}} = -\frac{1}{2}\left[\frac{1}{N}\sum_{i=0}^{N} logP(z_i^I, z_i^T; \tau) + \frac{1}{N}\sum_{i=0}^{N} logP(z_i^T, z_i^I; \tau)\right] \tag{2}$$

For a more comprehensive description of the loss, please see [13].

4.2 Classification Tasks

In order to use the various trained PLMs for downstream classification tasks, we used the traditional fine-tuning approach. Conventional fine-tuning can be achieved by adding task-specific layer(s) or an entire multi-layer perceptron (MLPs). The exact approach to processing the PLM output is dependent on the task.

In the case of document classification, the downstream task head is an MLP $f_{\text{MLP}}(\cdot)$ which takes the pooled sentence embedding output by the PLM as input and generates an n-dimensional vector, where n is the number of classes. That is, given an input text $\boldsymbol{x}$, we first process the raw input with the PLM to get the m-dimensional embedding of each token. Then a pooling operation, such as the mean, as applied to all token embeddings to produce a singular sentence embedding $h(\boldsymbol{x})$ of the same dimension m. Then $h(\boldsymbol{x})$ is fed to the MLP block in a standard feed-forward manner to get the probability across n classes with a softmax operator:

$$P(y \mid \boldsymbol{x}) = \frac{\exp\left((f_{\text{MLP}}\left(h(\boldsymbol{x})\right)_y\right)}{\exp\left(\sum_{i=1}^{n} f_{\text{MLP}}(h(\boldsymbol{x}))_i\right)}.$$

The MLP block can have any depth of layers $m \in \mathbb{N}$ and in our experiments we opted for $d = 2$. Since the additional MLP block and PLMs are modular, their respective parameters are stored separately and we can opt to freeze the parameters of one or the other. An example of processing a short input text sequence using this method is shown in Fig. 5.

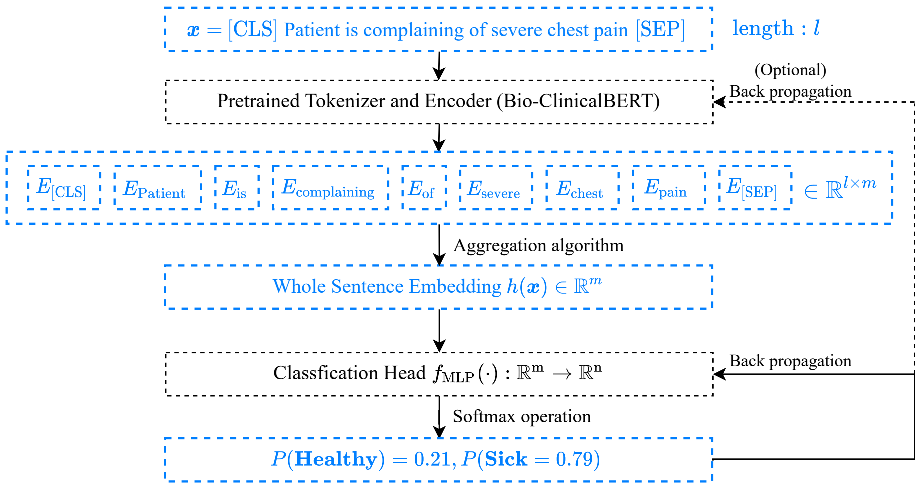

$\boldsymbol{x}$ = [CLS] Patient is complaining of severe chest pain [SEP]
length : l
Pretrained Tokenizer and Encoder (Bio-ClinicalBERT)
(Optional) Back propagation
$E_{[CLS]}$ $E_{Patient}$ E_{is} $E_{complaining}$ E_{of} E_{severe} E_{chest} E_{pain} $E_{[SEP]}$ $\in \mathbb{R}^{l\times m}$
Aggregation algorithm
Whole Sentence Embedding $h(\boldsymbol{x}) \in \mathbb{R}^m$
Classfication Head $f_{MLP}(\cdot) : \mathbb{R}^m \to \mathbb{R}^n$
Back propagation
Softmax operation
$P(\textbf{Healthy}) = 0.21, P(\textbf{Sick} = 0.79)$

Figure 5: Illustration of conventional fine-tuning method, with an option to freeze the PLM, shown in dotted line. Here $[CLS]$ and $[SEP]$ tokens are special tokens for BERT-based models that are added to the beginning and end of sequences. Taken from [19]

5 Implementation details

5.1 Pre-processing

Within NLP there is a wide range of typical pre-processing steps to apply to the text prior to passing to a model for training and evaluation. The level and depth of the pre-processing required can differ massively between models and desired task.

For language modelling with transformer based models minimal data cleaning is required as the tokenization of inputs paired with the *contextualised* representations of words actually means we want to keep as much of the original input as possible. Typical processing steps would include removal of carriage returns, tabs, and white space and any poorly encoded characters.

5.2 Hardware overview

Both training and inference were performed on a single machine hosted by Microsoft Azure with the following main specifications: 1 x NVIDIA Tesla T4 GPU, 4 x vCPUs from an AMD EPYC 7V12 processor, running Windows Server 2019.

5.3 Codebase

At the start of this project, there were a number of external codebases considered and we have adapted and applied a wide range of ideas from several. We list key repositories that play a part or influenced the work in Table 2. For all experiments we utilise PyTorch [20] in conjunction with the huggingface transformers library [2]. For DeCLUTR models we utilise AllenNLP [21].

5.4 Training setups

We performed quite a large and varied number of experiments, with different training regimes and requirements. An overview of these is presented in Table 3, although this is a curated selection intended to provide broad coverage of the models and training objective used.

Table 2: Related codebases.

Name	URL
BERTopic	`https://github.com/MaartenGr/BERTopic`
DeCLUTR	`https://github.com/JohnGiorgi/DeCLUTR`
CheckList	`https://github.com/marcotcr/checklist`
Ferret	`https://github.com/g8a9/ferret`
Bulk	`https://github.com/koaning/bulk`

Table 3: Overview of training setups for different PLMs. The *trained on* parameter refers to the amount of incident report data the model has been trained on. In-domain refers to whether this model has been explicitly trained using the patient safety incident reports data

Encoder	Pre-training		Model size	Trained on		
Text	Contrastive	In-domain	Parameters (m)	10k	100k	250k
RoBERTa-base	No	No	124.6			
RoBERTa-incident-base	No	Yes	124.6	✓	✓	✓
DeCLUTR-base	Yes	No	124.6			
DeCLUTR-base-incident	Yes	Yes	124.6	✓	✓	
RoBERTa-incident-DeCLUTR	Yes	Yes	124.6	✓	✓	✓

DeCLUTR Data Sampling

For the DeCLUTR models we initially only manipulated the sampling strategy, through changing the maximum length of documents spans to be used for the creation of anchor-positive pairs, in our experiments. We kept the actual sampling method used the same.

The sampling method used by DeCLUTR in its current form means that documents that do not meet a minimum span length are discarded and not used for either the MLM objective or contrastive learning objective used in the DeCLUTR training regime. As per the original paper, the minimum number of tokens for a document to be sampled is: $2 \times A_N \times S_{max}$ where A_N is the number of anchors to be sampled, and S_{max} is the maximum span length. For example the minimum span length for a document with 2 anchors and a maximum span length of 512 would be 2048.

The span length is thus a key parameter when training a DeCLUTR model and we have not been able to exhaustively explore all possible combinations. We instead opted for a span length that aligned with our average document length, and we recognise that future work could seek to explore this space further. The sample distributions based on the minimum document length we did explore are presented in Table 4.

Table 4: Sample distributions for different DeCLUTR sampling minimum document lengths.

Minimum document length	Max span length (S_{max})	Num anchors (A_N)	Samples obtained	Proportion of 250k dataset
16	4	2	206k	0.82
32	8	2	160k	0.64
64	16	2	95k	0.38
256	64	2	8.2k	0.03

Beyond this point in the report the models: DeCLUTR-base-incident and Incident-RoBERTa-DeCLUTR were trained using documents with a minimum length of 64 tokens, sampled from the same corpus used to train the Incident-RoBERTa-base model.

5.5 Hyperparameter choices

Table 5 reports the main hyperparameter choices for our experiments. Notably, the batch size has direct bearing on the difficulty of the contrastive objective the models are trained with (Equation 2). Intuitively, for any given anchor, the model needs to make the correct match out of the N reports in the batch. The probability of successfully finding the correct match by chance decreases with the batch size. We set the batch size to 32 on the single-GPU machine as this was the maximum possible across all span length options we tried. The original authors of DeCLUTR found 2 anchors was optimal for training, whereas the number of positives had little effect.

Each of the different experiments and objectives can require different hyperparameters and we opted to follow those used by original implementations where possible.

Table 5: Overview of hyperparameters used in experiments.

Parameter	LM	DeCLUTR	Pseudo-tasks	GLUE benchmark
Batch size	16	32	16	32
Gradient accumulation steps	4	1	1	1
Embedding dimension	768	768	768	768
Learning rate	1×10^{-5}	1×10^{-5}	1×10^{-4}	2×10^{-5}
Optimiser	AdamW	Adam W	Adam W	Adam W
Scheduler	linear with warmup	linear with warmup	linear with warmup	linear with warmup

6 Results

6.1 Language Modelling Losses

The loss for the LMs depends on the objective of the pretraining approach. For more traditional autoregressive language model objectives, like that used by GPT based models, perplexity is used: given a test set of sequences, perplexity is the normalised inverse probability over the set. In a way it is supposed to measure how well the underlying probability model predicts a given sequence. For MLM models like BERT/RoBERTa, perplexity is not well defined and during training the loss is calculated based on the predictions of the tokens at the masked positions. See perplexity in language models for more details on this. We present the loss curves for a selection of models in Fig.6.

The loss calculation for the DeCLUTR training combines the standard MLM cross entropy loss with a contrastive loss function, meaning the models are optimised for both tasks simultaneously. Again we present the training loss curves for the two main DeCLUTR models we focus on for later downstream tasks and behavioural testing in Fig.6. It is clear that the DeCLUTR models has this periodical spike in the loss, which we believe to be due to the random sampling of the anchor and positive pairs at the beginning of a new training epoch.

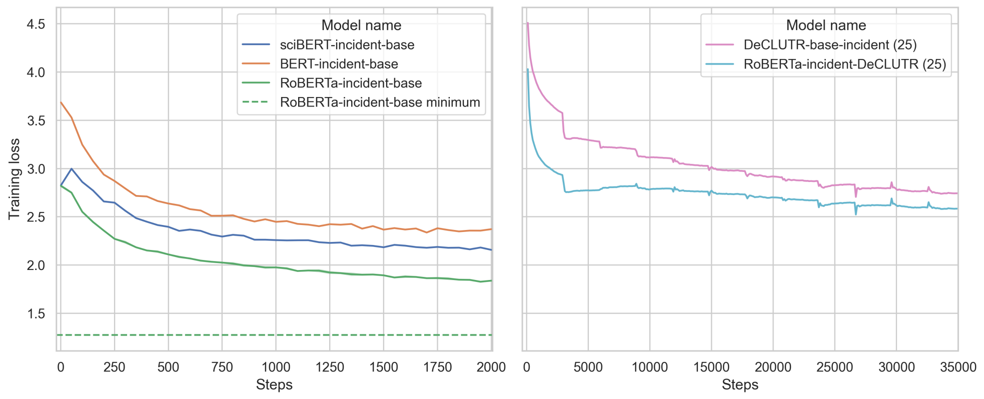


Figure 6: Training loss curves for masked language modelling in the left panel for a selection of open pretrained base models for 2000 steps, where the dotted line shows the minimum value reached after 50,000 steps for the RoBERTa-incident-base model. Similarly training loss curves for the DeCLUTR models appear in the right panel for approximately 35,000 steps. The reason for the large differences in steps between the two figures is due to the gradient accumulation steps used for the MLM-only models. The values in brackets after the model name indicates the number of epochs of further DeCLUTR pre-training performed.

6.2 Pseudo Classification Tasks

The results for the severity classification pseudo-task are presented in Table 6 and for the incident type classification pseudo-task in Table 7.[1]. It should be noted that we present results after training on the specific pseudo-task for only one epoch. The results for the baseline models can be found in Appendix A.1.

For both tasks we present results in both the frozen PLM and fine-tuned settings. The frozen PLM setting means the parameters of the PLM are set to *not require gradient*, which means gradients will not be computed during the backward pass and keeping these weights fixed and only the newly introduced parameters of the classification head are updated during training. Fine-tuning on the other hand will update all parameters, including the PLM, see Fig.5 for more details.

Table 6: Evaluation metrics for the PD09: severity classification pseudo-task for various models in both frozen and full fine-tuned settings. The values in brackets after the model name indicates the number of epochs of further DeCLUTR pre-training performed - see Appendix A.2 for more details on the results.

Model		Performance after 1 epoch					
Name	PLM	Trainable params.(m)	Accuracy	ROC AUC	F_1	Prec.	Recall
RoBERTa-base	Frozen	0.5	0.644	0.718	0.643	0.643	0.644
RoBERTa-incident-base	Frozen	0.5	0.704	0.787	0.687	0.679	0.704
DeCLUTR-base	Frozen	0.5	0.726	0.799	0.686	0.678	0.726
DeCLUTR-base-incident (10)	Frozen	0.5	0.771	0.855	0.752	0.740	0.771
RoBERTa-incident-DeCLUTR (3)	Frozen	0.5	**0.786**	**0.869**	**0.765**	**0.752**	**0.786**
RoBERTa-base	Fined-tuned	125	0.847	0.926	0.835	0.826	0.847
RoBERTa-incident-base	Fined-tuned	125	0.866	0.942	0.835	0.816	0.866
DeCLUTR-base	Fined-tuned	125	0.848	0.935	0.783	0.769	0.848
DeCLUTR-base-incident (3)	Fined-tuned	125	0.861	0.937	0.823	0.803	0.861
RoBERTa-incident-DeCLUTR (2)	Fined-tuned	125	**0.870**	**0.944**	**0.850**	**0.836**	**0.870**

Table 7: Evaluation metrics for the IN05: incident category classification pseudo-task for various models in both frozen and full fine-tuned settings. The values in brackets after the model name indicates the number of epochs of further DeCLUTR pre-training performed - see Appendix A.2 for more details on the results.

Model		Performance after 1 epoch					
Name	PLM	Trainable params.(m)	Accuracy	ROC AUC	F_1	Prec.	Recall
RoBERTa-base	Frozen	0.5	0.197	0.794	0.163	0.314	0.197
RoBERTa-incident-base	Frozen	0.5	0.498	0.851	0.474	0.494	0.499
DeCLUTR-base	Frozen	0.5	0.467	0.834	0.416	0.440	0.467
DeCLUTR-base-incident (25)	Frozen	0.5	0.577	0.893	0.544	0.557	0.577
RoBERTa-incident-DeCLUTR (5)	Frozen	0.5	**0.580**	**0.900**	**0.549**	**0.559**	**0.580**
RoBERTa-base	Fined-tuned	125	0.646	0.926	0.636	0.640	0.646
RoBERTa-incident-base	Fined-tuned	125	0.665	**0.935**	0.652	0.655	0.665
DeCLUTR-base	Fined-tuned	125	0.638	0.926	0.621	0.625	0.638
DeCLUTR-base-incident (3)	Fined-tuned	125	0.661	0.930	0.651	0.657	0.661
RoBERTa-incident-DeCLUTR (2)	Fined-tuned	125	**0.670**	**0.935**	**0.659**	**0.667**	**0.670**

6.3 Performance on GLUE benchmark

Results for the different PLMs performance on selected GLUE tasks [16] are presented in Table 8. Here we only report the results using fine-tuning and show the maximum score reached within 5 epochs. The selected tasks were: Standford Sentiment Treebank (**SST-2**) with the objective of determining sentiment of the input text, The Corpus of Linguistic Acceptability (**CoLA**) with the objective of acceptability of the input sentences grammar, and the Semantic Textual Similarity benchmark (**STS-B**) with the objective of determining the similarity of two sentences with a score from $1 - 5$.

[1]ROC AUC macro is a relatively poor metric for the multi-class problem of IN05 as good performance on one class may dominate and mask poor performance on other classes.

It can be seen from the results that the models which have been trained on a large portion of the incident reports data, and in particular with the contrastive loss function, have a degraded performance on the general domain tasks. To highlight this we have presented the results for the contrastively trained models in the best performing cases for the fine-tuned pseudo-tasks from Table 6 and Table 7, as well as the model after a longer period of contrastive pre-training.

Table 8: Evaluation metrics for the GLUE tasks: SST-2, STS-B, and CoLA.

Model		Max performance within 5 epochs		
Name	PLM	SST-2 - Accuracy	CoLA - Matts. corr.	STS-B - Spearmans r
RoBERTa-base	Fined-tuned	0.940	**0.591**	**0.905**
RoBERTa-incident-base	Fined-tuned	0.936	0.555	0.899
DeCLUTR-base	Fined-tuned	**0.944**	0.585	0.901
DeCLUTR-base-incident (3)	Fined-tuned	0.937	0.564	0.890
DeCLUTR-base-incident (25)	Fined-tuned	0.926	0.491	0.862
RoBERTa-incident-DeCLUTR (2)	Fined-tuned	0.942	0.552	0.887
RoBERTa-incident-DeCLUTR (25)	Fined-tuned	0.931	0.547	0.861

6.4 Topic modelling

Within the project we briefly investigated topic modelling with several of the trained models using a variety of topic modelling techniques such as LDA and BERTopic [22]. However we felt this research area warranted a much more involved and thorough investigation beyond the scope of the current timescales. We do present some preliminary results for topic modelling using BERTopic (with default parameters) for interested readers, but this is far from an exhaustive exploration.

BERTopic was chosen as a starting point as it entirely based on contextualised PLMs such as BERT or RoBERTa and thus fits our needs well. From a high-level, BERTopic first retrieves the sentence or document level embeddings of provided text produced by PLMs, then applies a dimensionality reduction technique such as Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP), before using a clustering technique like Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) to present similar groups of documents to the user. The BERTopic package modularises these steps such that other approaches are easily swapped in.

We evaluate the topic quality in terms of both topic diversity and topic coherence: Topic Diversity (TU) [23] measures the uniqueness of the words across all topics and will be between 0 and 1, with higher values implying greater diversity of individual topics. Normalized Pointwise Mutual Information (NPMI) [24] measures topic coherence internally using a sliding window to count word co-occurrence patterns, and ranges from -1 to 1 with 0 being optimal. For results see Table 9 [2]. The results presented below look at fixing an arbitrary number of topics ($N = 50$) and also having the optimal number learnt automatically by BERTopic across the various different models.

It must be stated clearly that the evaluation of topic models is a difficult task and there are not currently standardised and universally robust metrics, often instead needing to rely upon domain experts to manually review the derived topics or clusters to determine if they are sensible. For example, having high or low diversity does not necessarily mean a positive or negative performance and the metric of interest is entirely dependent on the dataset and required use. Nevertheless, Topic Modelling is a common and sought after NLP task given large unstructured datasets.

6.5 Probing and Behavioural Testing of Language Models

A common problem in the evaluation of language models is the means of evaluating performance, in particular when a clear or readily available downstream task is missing. Traditionally open source benchmark datasets such as the General Language Understanding Evaluation (GLUE) [16] are used to demonstrate performance of general domain language models on the general language domain: typically this is testing on data that is going to be similar to the data the model was initially trained on. As discussed previously, there is a need to both train and test language models on non-general domains and whilst an increasing number of biomedical datasets and accompanying biomedical benchmark datasets are being released, none are closely aligned to the specific NHS domain we are interested in.

To test and probe these various NLP models we sought to use a regime similar to that of unit testing in software engineering: a form of testing for specific behaviours and expected outcomes for the trained NLP models. This idea has

[2]Due to a bug in the BERTopic code used we were unable to present results on all models at this time, specifically we have replace RoBERTa-base with distilbert-base-uncased for the below experiments

Table 9: NPMI and Topic Diversity metrics for BERTopic models using different PLMs as the encoder. We include both evaluation scores when fixing an arbitrary number of topics ($T_N = 50$) and when automatically discovering the number of topics for each case, over a testing dataset of 100,000 samples.

Model	$T_N = 50$		T_N discovered		
Name	NPMI	Diversity	T_N	NPMI	Diversity
distilbert-base-uncased	0.122	0.648	53	0.088	0.723
RoBERTa-incident-base	0.112	0.612	91	0.126	0.641
DeCLUTR-base	0.131	0.674	72	0.114	0.744
DeCLUTR-base-incident (3)	0.109	0.598	75	0.123	0.705
DeCLUTR-base-incident (25)	0.121	0.654	95	0.145	0.679
RoBERTa-incident-DeCLUTR (2)	0.110	0.644	89	0.139	0.725
RoBERTa-incident-DeCLUTR (25)	0.116	0.624	80	0.127	0.691

been formulated as a framework called CheckList [25]. CheckList allows the implementation of various behavioural tests for NLP at scale and provides well designed visualisation tools. However, there is no pre-defined or standardised set of tests to quantify or benchmark a model and this is still a very exploratory space. We include a description of the proposed default families of tests in CheckList below.

Minimal Functionality Test

Minimal Functionality Tests (MFTs) are designed to test a specific behaviour of a model with a known or desired outcome/label. The example given by the CheckList authors is negation with a sentiment analysis model. For example, the sentence *"The film was enjoyable"* can be given a negative sentiment by adding negation, *"The film was **not** enjoyable"* and the models prediction should reflect this sentiment change.

Invariance Tests

Invariance tests (INVs) are tests designed to test the models ability to handle perturbations to the input without drastically changing the models output. For example replacing the country name in the sentence *"The hotel in Spain was amazing!"* to *"The hotel in England was amazing!"* should not lead to a change in sentiment predicted by the model.

Directional Expectation Test

Directional Expectation tests (DIRs) use perturbations to the input with known or expected directional results, such as adding negative phrases to a once positive statement should not lead to an increase in positive sentiment probability. For example *"The house was really beautiful"* changed to *"The house was really beautiful, but was in a bad neighbourhood"*, should not increase the probability of a positive sentiment.

Cosine similarity of embeddings

A common approach to assessing the *similarity* of two pieces of text is calculating cosine distance between their embedding vectors, which is the normalized dot product of two vectors A and B given by the formula in equation 3. This simple metrics opens the door to provide the model with texts that are known to be similar or dissimilar in meaning, and couple with an expected outcome of the cosine similarities between the two associated embeddings. The higher the cosine similarity score, the closer the embeddings are in the space and the higher the *relatedness* of the pieces of text that generated the embeddings.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}} \tag{3}$$

As an example, if we take sentence A: *"The man was happy to see them"*, sentence B: *"The woman was pleased to be there"* and sentence C: *"A horse was eating by the river"*, sentence A should be closer i.e. more similar to sentence B, than to sentence C.

6.6 Saliency mapping and explainability

A popular approach to *understanding* how a model made a *decision* with respect to a downstream task, in particular classification, is to investigate the weights of the model. In the case of large and relatively complex architectures such as transformer based PLMs, this is not simple and some would argue not advisable. We produced a pipeline to explore various techniques for attributing feature importance given a classification task within our domain using the ferret library mentioned in Table.2. The library produces outputs of a number of techniques from the explainability field discussed in turn below.

LIME

Local Interpretable Model-Agnostic Explanations (LIME) [26] is quite a simple approach, but can be computationally expensive as it works on a single prediction at a time. LIME creates permutations of the input, and after obtaining the predictions from the model, it uses (by default) the cosine distance between the original and perturbed versions as a metric of similarity or *closeness*. Weights are given to each perturbed sample based on the *closeness* with closer samples given greater weight. A local and interpretable model such as a lasso regression is trained using these perturbed data and the weights to generate the generally important features.

SHAP

SHapley Additive exPlanations (SHAP) [27] is a method based on game theory, and forms a *game* of reproducing the outcome of the trained model, with the *players* as the features provided to the model. SHAP then quantifies the contribution of each feature to the output of the model. Similar to LIME, SHAP works on one prediction at a time and provides local interpretability, but is computationally expensive as needs to compute every combinations of features.

Gradient based

Gradient based methods, e.g [28], are relatively simple methods which utilise the gradients calculated via back-propagation from the output or loss of the model with respect to the inputs.

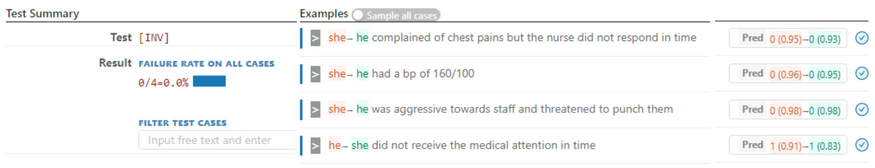
Integrated Gradients

Integrated gradients is akin to the reverse of the earlier gradient based methods and essentially uses a zero information baseline and integrates the gradient along the path from this baseline to the original input [29].

The methods described above are by no means exhaustive and it is an evolving field. Regardless, the general idea is to identify subsets or combinations of input features that maximise the prediction of a class. For a much greater depth of discussion, refer to the original papers cited.

6.7 Patient Safety Examples

We present a couple of examples of CheckList interface outputs in Fig.7 and Fig.8. These cover invariance tests on gender pronoun changes, and replacement with common domain specific synonyms.

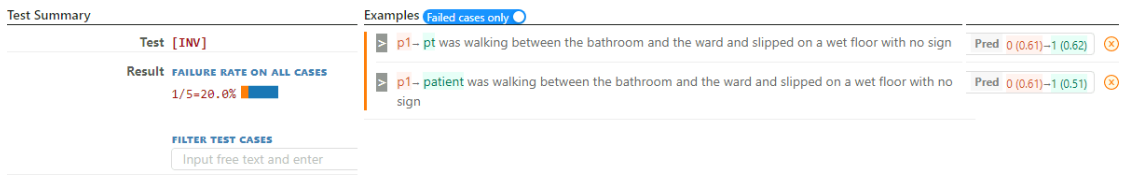


Test [INV]
Result FAILURE RATE ON ALL CASES
0/4=0.0%

Examples	Pred
she→ he complained of chest pains but the nurse did not respond in time	0 (0.95)→0 (0.93) ✓
she→ he had a bp of 160/100	0 (0.96)→0 (0.95) ✓
she→ he was aggressive towards staff and threatened to punch them	0 (0.98)→0 (0.98) ✓
he→ she did not receive the medical attention in time	1 (0.91)→1 (0.83) ✓

Figure 7: Example INV CheckList summary for gender pronoun changes with the model passes on all examples.

An example of *explained* classification decision using the *ferret* library is provided in Fig.9. This was produced using a model trained on the patient safety incident reports severity classification task.


Test Summary

Test [INV]

Result FAILURE RATE ON ALL CASES
1/5=20.0%

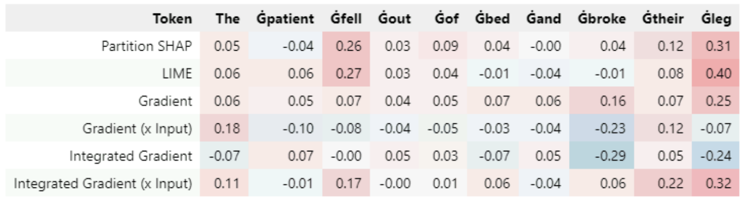
FILTER TEST CASES
Input free text and enter

Examples Failed cases only

p1→ pt was walking between the bathroom and the ward and slipped on a wet floor with no sign — Pred 0 (0.61)→1 (0.62)

p1→ patient was walking between the bathroom and the ward and slipped on a wet floor with no sign — Pred 0 (0.61)→1 (0.51)

Figure 8: Example INV checklist summary with failures. Here the patient pronouns were changed to common synonyms which are used to refer to a *patient*.



Token	The	Ġpatient	Ġfell	Ġout	Ġof	Ġbed	Ġand	Ġbroke	Ġtheir	Ġleg
Partition SHAP	0.05	-0.04	0.26	0.03	0.09	0.04	-0.00	0.04	0.12	0.31
LIME	0.06	0.06	0.27	0.03	0.04	-0.01	-0.04	-0.01	0.08	0.40
Gradient	0.06	0.05	0.07	0.04	0.05	0.07	0.06	0.16	0.07	0.25
Gradient (x Input)	0.18	-0.10	-0.08	-0.04	-0.05	-0.03	-0.04	-0.23	0.12	-0.07
Integrated Gradient	-0.07	0.07	-0.00	0.05	0.03	-0.07	0.05	-0.29	0.05	-0.24
Integrated Gradient (x Input)	0.11	-0.01	0.17	-0.00	0.01	0.06	-0.04	0.06	0.22	0.32

Figure 9: Example of a severity classification decision with feature importance derived by various techniques: LIME, SHAP, Gradients, and Integrated gradients

7 Discussion

7.1 General

Throughout this work a number of techniques related to language modelling have been developed and applied. Whilst these are quite data agnostic approaches, we sought to identify the most suitable models and workflows for a NHS text dataset, such as patient incident safety reports of interest.

A major goal of this project was to both train and evaluate a language model independent of a specific downstream task, instead relying on probing and exploratory analysis of the embedding space created by the model. It is clear, though not surprising, that differently trained language or NLP models produce distinct embedding spaces. These embedding spaces are the crucial component to any downstream task and this is highlighted using a number of probing methods and *pseudo* classification tasks. Whilst the pseudo-tasks presented may have limited realism or be desirable from a clinical utility standpoint, they do highlight the ability of a model to represent structure and decision boundaries of domain-specific data distributions.

Another finding of interest is the usefulness and re-usability of language model embeddings without the need to fine-tune entire models for different downstream tasks. We present results of fine-tuning entire PLMs with classification heads and frozen PLMs with fine-tuned classification heads only: the latter consisting of considerably fewer trainable parameters and thus much more resource efficient.

Results showed that the models trained on the patient safety incident reports using either the MLM objective, or the MLM plus contrastive loss objective, appeared to have a superior performance on the presented pseudo-tasks when compared to their general domain equivalent. Whilst the performance in the frozen setting did not match that of the full fine-tuned setting, we have not performed a thorough investigation, for instance we could look to utilising larger base models. Further there are other examples of promising approaches which can better utilise frozen language models at scale, such as prompt learning and parameter efficient fine-tuning [8].

7.2 Patient Safety Specific

We undertook the training of a RoBERTa-based PLM [17] on a large subset of the incident reports data, via continued training using the MLM objective for around 50,000 training steps. The resultant model then acted as our incident report PLM. Subsequently we introduced a contrastive loss objective to the pre-training stage using the DeCLUTR approach [13] to produce several further incident report PLMs.

In order to investigate the changes in the embedding space produced by these different models we derived *pseudo* classification tasks, CheckList-style test suites, and embedding similarity comparisons presenting aspects of the results in each case. Generally, we found a performance increase using the mixture of pre-training extensions, over the base general domain models.

8 Future work

8.1 Other Language Models

Whilst we opted to use RoBERTa-base [17] as the initial model the majority of our experiments, there is a wealth of alternative PLMs in the open which could be also explored, some of which are designed for biomedical texts such as BioBERT or clinicalBERT [30, 7, 6], knowledge enhanced transformers [31, 32], and many more variations. We similarly have the option to integrate a combination of ideas to enhance or change the pre-training of any model on the incident reports data.

8.2 CheckList Additions

The CheckList framework provided a valuable tool for proving and testing different aspects of a NLP model, and only a limited number were explored in this project. Future work could develop a much broader range of tests at a greater scale. Moreover, the integration of embedding cosine similarity expectation functions into the CheckList protocol would be desirable.

8.3 Gold Standard Downstream Tasks

With the help of easy to use frameworks such as Bulk one could attempt to label a sample of reports to provide a *gold standard* set of topic labels to allow more sophisticated approaches. The work done so far has sought to explore NLP models without the use of a well-defined downstream task, nevertheless one is surely needed to push this research towards a desired end goal of clinical utility.

8.4 Topic Modelling

The space of topic modelling is large and the ability to evaluate the derived *topics* not necessarily clear: however, given sufficient domain expertise a useful output is still achievable. Future work could seek to explore topic modelling with this data in much greater depth with a focus on how to best evaluate and integrate domain expertise.

8.5 Parameter efficient fine-tuning

A new wave of NLP research is investigating how to effectively and efficiently utilise PLMs for new datasets and downstream tasks without having to fully fine-tune all parameters each time. Given a suitable downstream task for patient safety incident reports, there is a sizable body of experiments to be carried out covering methods such as prompt learning, parameter efficient fine-tuning, adapter networks, and few-shot learning [8, 33, 34].

References

[1] 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, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi:10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[2] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi:10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[4] Christopher Mainey. Statistical methods for nhs incident reporting data. *PhD Thesis*, 2020. URL https://discovery.ucl.ac.uk/id/eprint/10094736.

[5] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi:10.18653/v1/2021.emnlp-main.243. URL `https://aclanthology.org/2021.emnlp-main.243`.

[6] Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission, 2019. URL `https://arxiv.org/abs/1904.05342`.

[7] Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi:10.18653/v1/W19-1909. URL `https://aclanthology.org/W19-1909`.

[8] Niall Taylor, Yi Zhang, Dan Joyce, Alejo Nevado-Holgado, and Andrey Kormilitzin. Clinical prompt learning with frozen language models. arXiv, 2022. doi:10.48550/arXiv.2205.05535. URL `https://arxiv.org/abs/2205.05535`.

[9] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. 2019. doi:10.48550/arXiv.1903.10676. URL `https://arxiv.org/abs/1903.10676`.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013. doi:10.48550/arXiv.1301.3781. URL `https://arxiv.org/abs/1301.3781`.

[11] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1162. URL `https://aclanthology.org/D14-1162`.

[12] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL `http://arxiv.org/abs/1908.10084`.

[13] John Giorgi, Osvald Nitski, Bo Wang, and Gary Bader. Declutr: Deep contrastive learning for unsupervised textual representations. arXiv, 2020. doi:10.48550/arXiv.2006.03659. URL `https://arxiv.org/abs/2006.03659`.

[14] Fredrik Carlsson, Amaru Cuba Gyllensten, Evangelia Gogoulou, Erik Ylipää Hellqvist, and Magnus Sahlgren. Semantic re-tuning with contrastive tension. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=Ov_sMNau-PF`.

[15] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. 2020. URL `https://arxiv.org/abs/2004.05150`.

[16] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint 1804.07461, 2018.

[17] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. 2019. doi:10.48550/arXiv.1907.11692. URL `https://arxiv.org/abs/1907.11692`.

[18] Dongju Park and Chang Wook Ahn. Self-supervised contextual data augmentation for natural language processing. *Symmetry*, 11:1393, 11 2019. doi:10.3390/sym11111393.

[19] Niall Taylor, Lei Sha, Dan W Joyce, Thomas Lukasiewicz, Alejo Nevado-Holgado, and Andrey Kormilitzin. Rationale production to support clinical decision-making, 2021. URL `https://arxiv.org/abs/2111.07611`.

[20] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[21] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi:10.18653/v1/W18-2501. URL `https://aclanthology.org/W18-2501`.

[22] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure. arXiv, 2022. doi:10.48550/arXiv.2203.05794. URL `https://arxiv.org/abs/2203.05794`.

[23] Feng Nan, Ran Ding, Ramesh Nallapati, and Bing Xiang. Topic modeling with Wasserstein autoencoders. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6345–6381, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1640. URL `https://aclanthology.org/P19-1640`.

[24] David Newman, Jey Lau, Karl Grieser, and Timothy Baldwin. Automatic evaluation of topic coherence. pages 100–108, 01 2010.

[25] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Association for Computational Linguistics (ACL)*, 2020.

[26] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi:10.1145/2939672.2939778. URL `https://doi.org/10.1145/2939672.2939778`.

[27] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf`.

[28] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2013. URL `https://arxiv.org/abs/1312.6034`.

[29] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/sundararajan17a.html`.

[30] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746, 2019. URL `http://arxiv.org/abs/1901.08746`.

[31] Michihiro Yasunaga, Jure Leskovec, and Percy Liang. Linkbert: Pretraining language models with document links. In *Association for Computational Linguistics (ACL)*, 2022.

[32] Fangyu Liu, Ehsan Shareghi, Zaiqiao Meng, Marco Basaldella, and Nigel Collier. Self-alignment pretraining for biomedical entity representations. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4228–4238, June 2021.

[33] Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. Openprompt: An open-source framework for prompt-learning. arXiv, 2021. doi:10.48550/arXiv.2111.01998. URL `https://arxiv.org/abs/2111.01998`.

[34] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019. URL `https://arxiv.org/abs/1902.00751`.

[35] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany, August 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-2034. URL `https://aclanthology.org/P16-2034`.

A Appendix

A.1 Baseline model results

Classification results for the incident report severity pseudo-task using a Bi-LSTM model with attention inspired by [35], and a tf-idf (term frequency - inverse document frequency) vectorizer with a random forest classifier.

Table 10: Evaluation metrics for the PD09: Severity classification task using two baseline models

Model	Performance after 1 epoch				
Name	Accuracy	ROC AUC	F_1	Prec.	Recall
BiLSTM w/attention	0.833	0.908	0.813	0.801	0.833
Random Forest	0.844	0.878	0.791	0.681	0.680

A.2 DeCLUTR extended results

More granular classification results for the incident report severity and incident category pseudo-tasks, varying numbers of epochs of pre-training with the DeCLUTR approach (i.e. both the MLM and contrastive loss objective). The models were trained using documents with a minimum length of 64 tokens, see Table 4 for more details.

Table 11: Evaluation metrics for the PD09: severity classification pseudo-task for various models in both frozen and full fine-tuned settings of DeCLUTR models at various different total epochs of further pre-training for performance after one epoch of training on the task.

Model			Performance after 1 epoch					
Name	PLM	Pre. Epochs	Trainable params.(m)	Accuracy	ROC AUC	F_1	Prec.	Recall
DeCLUTR-base-incident	Frozen	2	0.5	0.758	0.842	0.725	0.713	0.758
		3	0.5	0.763	0.847	0.719	0.709	0.763
		5	0.5	0.763	0.848	0.75	0.741	0.763
		10	0.5	**0.771**	0.855	0.752	0.74	**0.771**
		25	0.5	0.769	**0.859**	**0.756**	**0.746**	0.769
		50	0.5	0.755	0.840	0.717	0.705	0.755
RoBERTa-incident-DeCLUTR	Frozen	2	0.5	0.775	0.858	0.750	0.736	0.775
		3	0.5	**0.786**	0.869	0.765	0.752	**0.786**
		5	0.5	0.780	**0.871**	**0.769**	**0.760**	0.780
		10	0.5	0.782	0.865	0.751	0.731	0.762
		25	0.5	0.776	0.866	0.752	0.740	0.776
		50	0.5	0.765	0.851	0.734	0.720	0.765
DeCLUTR-base-incident	Fine-tuned	2	125	0.859	0.936	0.833	0.816	0.859
		3	125	**0.861**	**0.937**	0.823	0.803	**0.861**
		5	125	0.858	0.936	0.841	0.828	0.858
		10	125	0.859	0.934	0.827	0.808	0.859
		25	125	0.854	0.934	**0.846**	**0.835**	0.854
		50	125	0.851	0.931	0.792	0.767	0.851
RoBERTa-incident-DeCLUTR	Fine-tuned	2	125	**0.870**	**0.944**	**0.850**	**0.836**	**0.870**
		3	125	0.868	0.942	0.838	0.819	0.868
		5	125	0.867	0.941	0.831	0.811	0.867
		10	125	0.858	0.938	0.825	0.807	0.858
		25	125	0.863	0.939	0.833	0.815	0.863
		50	125	0.857	0.934	0.831	0.815	0.857

Table 12: Evaluation metrics for the IN05: incident category classification pseudo-task for various models in both frozen and full fine-tuned settings of DeCLUTR models at various different total epochs of further pre-training for performance after one epoch of training on the task.

Model				Performance after 1 epoch				
Name	PLM	Pre. Epochs	Trainable params.(m)	Accuracy	ROC AUC	F_1	Prec.	Recall
DeCLUTR-base-incident	Frozen	2	0.5	0.548	0.878	0.511	0.525	0.548
		3	0.5	0.542	0.883	0.503	0.509	0.542
		5	0.5	0.559	0.886	0.527	0.531	0.559
		10	0.5	0.562	0.888	0.532	0.534	0.562
		25	0.5	**0.577**	**0.893**	**0.544**	**0.557**	**0.577**
		50	0.5	0.550	0.877	0.518	0.538	0.550
RoBERTa-incident-DeCLUTR	Frozen	2	0.5	0.555	0.883	0.521	0.527	0.555
		3	0.5	0.568	0.892	0.531	0.541	0.568
		5	0.5	**0.580**	**0.900**	**0.549**	**0.559**	**0.580**
		10	0.5	0.578	0.894	0.544	0.551	0.578
		25	0.5	0.574	0.892	0.545	0.545	0.574
		50	0.5	0.572	0.883	0.548	0.552	0.572
DeCLUTR-base-incident	Fine-tuned	2	125	0.648	0.930	0.641	0.651	0.648
		3	125	**0.661**	0.930	**0.651**	**0.657**	**0.661**
		5	125	0.649	**0.931**	0.632	0.641	0.649
		10	125	0.658	0.931	0.647	0.647	0.658
		25	125	0.651	0.931	0.640	0.643	0.651
		50	125	0.645	0.929	0.631	0.635	0.645
RoBERTa-incident-DeCLUTR	Fine-tuned	2	125	**0.670**	**0.935**	0.659	0.667	**0.670**
		3	125	0.668	0.933	**0.660**	**0.672**	0.668
		5	125	0.660	0.934	0.646	0.660	0.660
		10	125	0.658	0.932	0.644	0.653	0.658
		25	125	0.659	**0.935**	0.649	0.654	0.659
		50	125	0.653	0.934	0.644	0.654	0.653