DEPARTMENT OF ENGINEERING MATHEMATICS

Improving Text Classifier Performance through Human-in-the-Loop: Enhancing Learning from Explanations

Xinyang Song

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

Thursday 17th August, 2023

Supervisor: Dr. Edwin Simpson

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Xinyang Song, Thursday 17th August, 2023

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Text classification utilises Natural Language Processing (NLP) techniques to analyse pre-trained texts and assign them appropriate labels. Especially in crisis situations like floods, earthquakes, etc., text classifiers are crucial in identifying key information and effectively forwarding it to relevant agencies. However, the efficacy of text classification largely depends on abundant rich training data, which may be difficult to obtain in many scenarios [37]. Relying on a vast amount of non-representative annotated data can lead to delays in project commencement and may impact the model's accuracy. Specifically, in emergencies, identifying action-related information (such as casualties or missing persons) becomes particularly challenging.

To address these issues, this paper introduces a Human-in-the-Loop (HITL) system integrated into the classifier's training process. Coupled with the representational engineering technique of natural language explanations—ExpBERT, a BERT fine-tuned on the MultiNLI natural language inference dataset is utilised to learn from explanations [42]. This optimised embedding representation is used as input to the neural network classifier, further enhancing performance. The core task of the paper revolves around improving the performance of the text classifier based on ExpBERT by combining a limited but representative set of samples with labels and explanations provided by either humans or OpenAI [5].

This paper delves deeply into the application of active learning in text classification tasks based on ExpBERT. During the iterative process, acquisition functions based on uncertainty and diversity are used to select representative unlabeled instances for annotators to process. The original small sample and newly annotated data are used for model retraining. The results indicate that this method can achieve performance similar to models trained on vast datasets within a few iterations. To substantiate this, the model's performance using different active learning strategies is compared on the CrisisNLP dataset, and the effects of an active learning system combined with Monte Carlo Dropout (MCD) are also evaluated.

The main conclusions of this paper are:

- The Human-in-the-loop framework combined with a text classifier enables the model to achieve, or even surpass, the performance of models trained with the full dataset, using only a small amount of annotated data and undergoing a limited number of iterations.
- Compared to merely annotating labels, low-quality explanations, or a limited number of explanations, annotating a certain amount of high-quality explanations can significantly boost the model's performance.
- In terms of acquisition functions, semantic diversity-based sampling and Bayesian Active Learning by Disagreement (BALD) strategies combined with Monte Carlo Dropout (MCD) can allow the model to achieve higher average performance post-training.

Supporting Technologies

- I used Python as the development language.
- I used the *Transformers* library to introduce pre-trained models and *Pytorch* for implementing neural networks.
- I used OpenAI models as one of the annotation methods.
- I used LaTeX to format my thesis, via the online service *Overleaf*.

Notation and Acronyms

NLP	:	Natural Language Processing
BERT	:	Bidirectional Encoder Representations from Transformers
ExpBERT	:	Representation Engineering with Natural Language Explanations
NLI	:	Natural Language Inference
HITL	:	Human-in-the-loop
AL	:	Active Learning
BALD	:	Bayesian Active Learning by Disagreement
MCD	:	Monte Carlo Dropout
GPT	:	Generative Pre-trained Transformer
NN	:	Neural networks

Acknowledgements

Xinyang Song would like to extend her profound gratitude to her supervisor, Dr. Edwin Simpson, for sharing literature and implementation examples related to ExpBERT and active learning, offering countless invaluable assistance. His professional guidance and enthusiastic support helped me overcome numerous challenges, complete my dissertation, and achieve meaningful insights.

Additionally, I'd like to thank the University of Bristol and my peers for their resources and emotional support during this time. My gratitude goes out to all of you.

Chapter 1

Introduction

This chapter begins by introducing the research background of the text classifier based on ExpBERT integrated with Human-in-the-loop (Section 1.1). Based on this background, the motivation for the experiment was formed in Section 1.2, emphasising the significance of this research. Furthermore, a general overview of the work will be provided, contrasting with the limitations of traditional methods. Finally, the primary objectives and challenges are briefly summarised (Section 1.3).

1.1 Background

Emergency response systems for social platforms mainly focus on developing better text classification algorithms to learn from limited data. However, obtaining valuable annotated datasets can prove to be challenging [47]. The text classifier based on ExpBERT receives classification labels and corresponding explanations. These explanations detail which keywords led to a particular classification. ExpBERT incorporates this information into the model's training, allowing it to learn from deeper semantic insights to enhance its generalisation capabilities. Therefore, annotating with representative data and understanding the key phrases is vital for guiding the model towards accurate classification.

However, the data throughput of typical social platforms does not permit the annotation of tens of thousands of entries. Consequently, various forms of active learning that can sample high-information-value data at low costs have found extensive application in classification projects [48, 61]. Active learning emerges as an effective solution to these challenges, selecting a limited number of highly informative unannotated samples for human experts in the loop to annotate [54, 15]. The newly data, once effectively annotated, dictates the performance of the model in the next iteration. Thus, employing different acquisition functions to query the most information-rich new instances is perhaps the most popular approach in active learning. Acquisition functions have naturally become the focal point of research in the domain of active learning.

Additionally, neural network-based text classifiers are often ill-suited for early uncertainty sampling [52, 50]. This is because the weight parameters of the neural network are fixed values, causing the model to be overly confident in its predictions, both correct and incorrect. However, by employing dropout, it's possible to introduce a degree of uncertainty into the model. During training, the dropout layer randomly "turns off" a subset of input units. This randomness ensures that network weights are no longer fixed values, thus simulating weight uncertainty.

1.2 Motivation

In light of the aforementioned background, this section emphasises the research motivation from three aspects: accuracy, representativeness, and robustness.

1.2.1 Accuracy

A unified feature of social networks, exemplified by Twitter, is the vast amount of data combined with internet-specific language syntax. The ultimate goal is to precisely classify urgent messages when capturing emergent needs and to dispatch relevant departments to address those needs. However, traditional text classification systems that achieve performance improvements are primarily done through extensive supervised learning, which is costly and slow to respond. Texts with varying amounts of information are randomly distributed, and the difference in information content can significantly impact model accuracy. Consequently, the results of capturing and identifying crucial emergency information are often subpar. Therefore, employing active learning models to utilise a small amount of data and enhance accuracy is urgently needed.

1.2.2 Representativeness

The ability to query rich feature representations from a large pool of unannotated data can mitigate much of the bias in multi-class active learning problems for emergency scenarios. By actively selecting samples for annotation, the model can choose the most valuable and representative samples in each iteration to enhance its performance. Therefore, selecting an effective acquisition function to seek out representative data has the most profound impact on performance and offers the most significant room for improvement.

1.2.3 Robustness

Original models often exhibit an overconfidence bias when handling text classification tasks. When encountering unfamiliar content, they make incorrect and unreliable predictions. In many practical applications, such as disaster response and medical diagnostics, erroneous predictions can have severe consequences. Thus, introducing a certain level of uncertainty to the model to curb its overconfidence has become an essential research topic. On the other hand, we can enhance the robustness of the model without introducing excessive computational overhead. This holds significant value for neural networks in scenarios that demand highly accurate and reliable predictions.

1.3 Experimental Methodology

Considering the background and motivations, this paper eventually designs a pool-based Human-in-the-loop active learning system for text classification. Unlike traditional text classification models, we train with a small amount of annotated data and then use the trained model in combination with various acquisition functions to select instances. Subsequently, preset annotations or human or OpenAI annotators observe and analyse the key features of the extracted instances in the loop, providing corresponding labels and adding a certain amount of explanations. These newly annotated instances are incorporated into the original data, and the above cycle is repeated until the performance reaches that of the full data (with the training set ratio close to 1, annotated with labels and default explanations).

The implementation will utilise uncertainty sampling based on Least Confidence, sentiment diversity sampling, and random sampling as a baseline to examine the performance under various active learning strategies. Detailed explanations of the acquisition functions will be provided in Chapter 2. Furthermore, in every iteration, the concatenated explanatory texts will be processed using a pre-trained model based on ExpBERT to generate deep semantic representations. This embedded vector will then be input into the neural network text classifier.

Lastly, structural modifications will be made to traditional neural network(NN) models. Using the dropout mechanism, the model becomes more robust to minor variations in input, thereby enhancing its generalisation performance. Additionally, dropout can address the model's overconfidence issue by introducing noise and randomness, ensuring that model predictions aren't overly deterministic. While dropout doesn't directly quantify uncertainty, the introduced randomness and noise can bolster the model's resilience and, to some extent, alleviate its overconfidence. Given the high computational cost of entropy calculations in the original Bayesian Active Learning by Disagreement (BALD) algorithm, we've refined

the BALD algorithm. Utilising Monte Carlo Dropout (MCD), we employ Least Confidence to select samples where the model's predictive probability is most uniform (i.e., without a particularly high predictive probability for any category).

1.3.1 Objectives

Based on the above experimental descriptions, the objectives of this paper can be summarised as follows:

- Develop a pool-based Human-in-the-loop framework applied to the ExpBERT-based text classifier and investigate its effectiveness.
- Construct an annotator simulation process and design stopping criteria (e.g., terminate when reaching the performance of the full data model).
- Establish a baseline (performance of the full-data model without active learning or random sampling) and study whether active learning can enhance the performance of the text classification model. Compare the effects of different acquisition functions.
- Explore the impact of the quality and quantity of explanations on this system. For instance, providing explanations with noise or increasing the number of explanations added in each iteration.

1.3.2 Challenges

The main challenges of this project lie in the significant differences that may exist between various active learning strategies and how to improve the original initialisation framework of the multi-classifier based on ExpBERT and the annotation process to reduce computational costs.

- It's worth noting that there are multiple acquisition functions in active learning text classification systems. The application of different algorithms in active learning strategies significantly affects the performance of the active learning framework. The choice of strategy needs to consider both time complexity and representativeness.
- Secondly, neural networks with dropout mechanisms might increase computational costs during training algorithm execution and when using MCD for active learning. Thus, how to reduce computational costs in each iteration to improve response efficiency poses a challenge.
- Finally, regarding the framework design, determining the appropriate sampling ratio, the number of iterations, and the number of explanations provided in each iteration requires further study to achieve satisfactory performance.

Chapter 2

Background

Chapter 2 introduces the relevant techniques centred around enhancing the performance of the text classifier based on ExpBERT through active learning. The ExpBERT mechanism used in the experiment is introduced (Section 2.1). ExpBERT will be applied during dataset pre-training. The active learning-based "Human-in-the-loop" (Section 2.2) and its main acquisition functions (Sections 2.3 and 2.4) are the focal implementation parts of this experiment, which can rapidly boost performance through this technique. Section 2.5 describes the multi-model integration method, while Section 2.6 discusses using OpenAI models to simulate the annotator role within the Human-in-the-loop. Lastly, the feasibility of applying Bayesian theory to active learning is concerned.

2.1 ExpBERT

Representation Engineering with Natural Language Explanations (ExpBERT) proposes a method to enhance the control over the inductive biases of a model and maintains the expressive power of the representation [42]. It combines fixed explanations provided by humans or, in this experiment, annotators with tweets to learn from these explanations and improve the model's performance. Figure 2.1 intuitively shows how to combine samples with explanations with BERT fine-tuned on MultiNLI [62]. Explanations play a key role here. The quality of the explanations has a significant impact on the performance of ExpBERT. By using high-quality explanations, the model's learning can be guided.

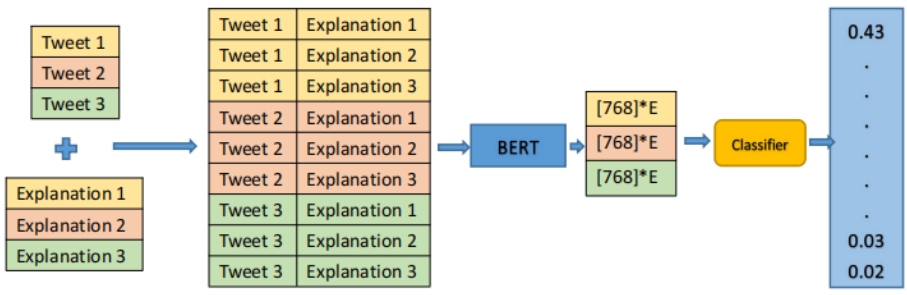

Figure 2.1: ExpBERT to produce representations that form the input to the classifier

First, each tweet is fully connected with a set of pre-written explanations to generate a 3×3 set. These explanations are unrelated to specific tweets; each tweet will be connected with the same number of explanations. The preprocessed text and explanations are input into BERT to generate features that "interpret" each explanation [13]. The classifier can be trained to classify the representations of the tweets and explanations.

After being fine-tuned on the MultiNLI dataset, the BERT model will generate a feature vector for each input sample, representing the entire input of a length of 768. The feature vectors of tweets and explanations are then connected to form an embedding with a size of 768∗E, where E is the number

of explanations. This will be used as the input data for training and predicting in the classifier model. This paper will use this model as the basic pre-trained model. At the same time, combined with Natural Language Inference (NLI) technique can be used to reduce the number of embeddings (the size will be 768 + (3∗E)) and integrate and initialise them into the HITL system [34]. Because this model can handle instance vectors with explanations, the optimised embedding representation as input to the classification model can improve the performance of the model.

2.2 HITL

Traditional natural language processing workflows are not designed to fully leverage human feedback. In contrast, Human-in-the-Loop (HITL) serves as an essential component of interactive systems, revealing potential model flaws through simulating human roles within the loop. These flaws might not be readily apparent before real-world testing [59]. Godbole et al. (2004) [19] expanded a text classifier employing Support Vector Machine (SVM) active learning. Ingeniously, they integrated human input on aspects such as feature engineering, terminology selection, and document labelling into the model, allowing statistically sound judgments to be made. This innovative mode of interaction between humans and machine learning algorithms is termed "Human-in-the-Loop (HITL) machine learning" [39]. In this approach, various HITL machine-learning schemes can be determined based on different needs and contexts, driven by different types of collaborations between humans and machines [40].

Active Learning (AL) [54]: The essence of active learning lies in the system's control over the model's learning process. Although humans serve as intermediaries to annotate unlabeled data, they cannot choose unlabeled data based on preference. AL will be applied as an optimisation framework in this paper, with a detailed exploration of its application in Section 2.3.

Interactive Machine Learning (IML) [2] is a method where humans maintain close interaction with the system. Within this context, there are notable distinctions between Active Learning (AL) and Interactive Machine Learning (IML). Dudley and Kristensson (2018) [14] highlighted that although both AL and IML prioritise the user's selection of new data points for labelling, they are driven by different motivations. Specifically, in AL, the selection is propelled by the model itself, while in IML, the choice is driven by the user. Given that IML has evolved from the foundations of AL, they share several common drawbacks. However, IML also faces unique challenges, especially those related to Human-Computer Interaction (HCI) complexities. These distinct challenges necessitate more specialised research to address, as indicated by Michael et al. (2020) [38].

Machine Teaching (MT) [49] is an approach where machine learning models are trained under the guidance of human educators. More precisely, human teachers define the knowledge they aim to convey to the model, underscoring their active involvement and direction throughout the learning process. Devidze et al. (2020) [12] pointed out that, compared to active learning, MT heavily leans on the teacher's expertise and offers less flexibility in sample selection and handling intricate tasks. Hence, selecting the right learning method becomes paramount based on the specific requirements.

2.3 Active Learning

Within the context of HITL, active learning systems, as one of the most popular learning approaches, aim to overcome the labelling bottleneck by posing questions to unlabeled instances and having them labelled by experts, such as human annotators [1, 54]. In essence, active learning seeks to identify the most informative unlabeled raw data and present it to annotators for labelling. This process closely mirrors real-world scenarios of extracting information from source data. Annotators label this source data, and the labelled instances are incorporated into the model's training process. In this way, active learning manages to achieve performance comparable to using the full dataset but with fewer labelled data, thereby addressing the labelling bottleneck. The active learning approach proves especially pertinent given this project's vast number of instances, coupled with significant noise, cumbersome full-scale training tasks, and suboptimal accuracy. Thus, active learning is apt for this project, focusing on instances that might enhance accuracy, minimise irrelevant instances, and boost the learning efficiency and precision of the

emergency response system.

2.3.1 The AL Process and Scenarios

The active learning process is illustrated in Figure 2.2, which depicts the process of pool-based active learning. The machine learning model is initially trained using the labelled instance set L. Afterwards, the model gives the score to the unlabeled dataset and selects representative unlabeled samples based on the acquisition function, presenting them to human annotators. These annotators label the chosen samples, which are then removed from the unlabeled sample set U and added to the labelled set L. Through iterative execution of the steps above, the quantity of labelled samples grows incrementally, enhancing the model's performance. The active learning process will conclude once the stopping criterion is met, such as achieving performance metrics on the entirety of the data set.

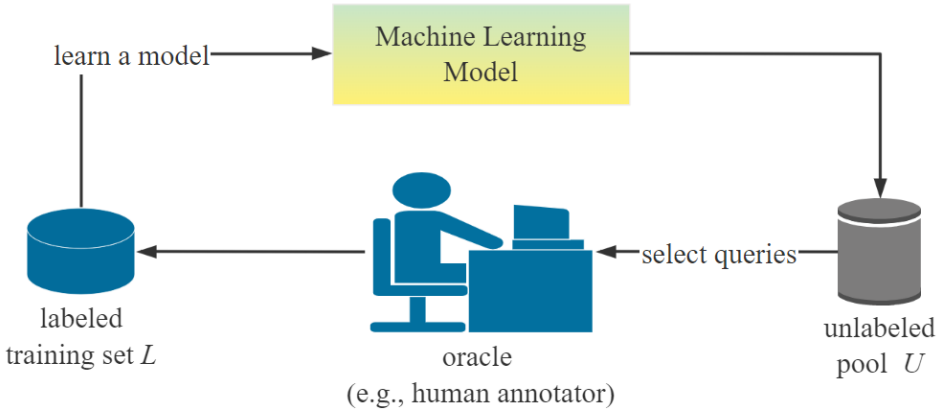

Figure 2.2: Pool-based active learning process

Based on the classification by Settles (2009) [54], active learning primarily encompasses three scenario settings: Membership Query Synthesis [3, 29], Stream-based Selective Sampling [9], and Pool-based active learning [32].

Membership Query Synthesis: In this active learning scenario, the strategy is generated by the model itself. The machine learning model can request labels for any unlabeled instance that hasn't been sampled from the underlying natural distribution [54]. This query approach is typically effective for more deterministic problem domains, such as tasks involving regression predictions on absolute coordinates. It can interpret straightforward data distributions and construct viable data for human annotation. However, as evidenced in the study by Baum and Lang (1992) [30], in complex tasks like natural language processing, the model might produce text strings that are hard to comprehend, rendering human judgment on these jumbled texts challenging. Within the context of deep active learning, one can address the scenarios of Membership Query Synthesis through Generative Adversarial Networks (GANs) for data augmentation, as GANs are capable of generating instances with a high degree of plausibility [20].

Stream-based Selective Sampling: The distinction of this setup in comparison to Membership Query Synthesis lies in its ability to operate irrespective of the input distribution. Stream-based Selective Sampling entails drawing a single unlabeled instance from the actual distribution at a time. The model then determines whether to request the label of the instance through "information measures" or "query strategies", essentially acting as a kind of biased random sampling [9].

Pool-based active learning: As a commonly employed setting in active learning, the difference between pool-based active learning and stream-based Selective Sampling is that the former employs a greedy mechanism before selecting the best query, comparing the entire dataset. In contrast, stream-based sampling assesses data as it comes in on an individual basis. However, dealing with vast datasets could result in lengthy computational times due to the need to evaluate all the data. Nonetheless, this method is

particularly suitable for scenarios similar to this project where manual annotation costs are significant. By choosing the most valuable data for annotation, it aims to maximise the return on the annotation investment.

2.3.2 Acquisition Functions

This section considers various acquisition functions in active learning. In the design of the paper (Chapter 3), we will utilise a text multi-classification model based on ExpBERT combined with different acquisition functions to enhance performance.

Random Sampling

Random sampling involves selecting instances independent of predictions, data, or model considerations. It's commonly used as a task benchmark since it might overlook potential informative instances, reducing learning efficiency. In this context, random sampling will serve as a baseline to contrast with the more sophisticated strategies discussed below, especially when the annotation pool becomes overly extensive [53].

Prediction Uncertainty Sampling

Uncertainty sampling is a specific active learning strategy that queries instances most difficult for the model to classify. By annotating these challenging instances, the model's accuracy is enhanced. In probabilistic models for binary classification, such instances have posterior positive probabilities close to 0.5 [32]. However, for more complex multi-label classification tasks, entropy-based methods are used. The more uniform the probability distribution, the greater the entropy. As entropy increases, the uncertainty or informational content of the random variable becomes higher. Conversely, when probabilities are concentrated on a few data points, the uncertainty is lower and the informational content is smaller.

$$x^*_{\mathrm{ENT}} = \arg\max_x -\sum_i P\left(y_i \mid x;\theta\right) \log P\left(y_i \mid x;\theta\right) \tag{2.1}$$

$P\left(y_i\right)$ denotes the probability distribution at classification i.

In the domain of text classification, an alternative method often used to measure uncertainty is the "Least Confidence(LC)" [25]:

$$x^*_{LC} = \arg\min_x P\left(y^* \mid x;\theta\right), y^* = \arg\max_y P(y \mid x;\theta) \tag{2.2}$$

y^* represents the label of the class with the highest likelihood. This method is equivalent to the effectiveness of entropy-based algorithms for binary text classification. It focuses on samples for which the model predicts the highest probability but with low confidence. Specifically, the method involves annotating the samples with the smallest maximum probability. This more comprehensive consideration will be adopted in this paper.

In addition to the two commonly used measurement methods mentioned above, Munro [39] introduced the margin of confidence and ratio of confidence. The former refers to the difference between the two most confident predictions, while the latter is the ratio between these two predictions.

$$\text{margin\_conf} = 1 - \left(P\left(y^*_1 \mid x\right) - P\left(y^*_2 \mid x\right)\right), \quad \text{ration\_conf} = \frac{P\left(y^*_2 \mid x\right)}{P\left(y^*_1 \mid x\right)} \tag{2.3}$$

y^*_1 is the most confident, y^*_2 is the second most confident.

Query-By-Committee

Compared to uncertainty sampling, which evaluates the judgement of a single model, the Query-By-Committee (QBC) approach employs multiple models to act as "members" of a committee [55]. These members are trained on the available labelled data using the Gibbs algorithm. The committee then randomly selects a hypothesis consistent with the current labelled data. The subsequent query is picked based on the instance where the committee members exhibit the most significant disagreement [8]. Hence,

while uncertainty sampling assesses the discernment of an individual member (model), QBC often incorporates strategies such as using vote entropy to select instances that are challenging for models to agree upon [9]. And choosing samples with a high average Kullback-Leibler (KL) divergence [36].

Vote entropy: For text classification tasks, each committee member (text classifiers) can vote on unlabeled samples.

$$H(V) = -\sum_{i=1}^{n} \frac{Vote(y_i)}{C} \log \frac{Vote(y_i)}{C} \tag{2.4}$$

$Vote(y_i)$ represents the votes received for class y_i, and the total number of votes is C, which is also the total number of classifiers. The larger the vote entropy, the more controversial the sample is.

Average KL Divergence: The KL divergence increases as the difference between two distributions grows. The formula is expressed as:

$$x^*_{KL} = argmax_x \frac{1}{C} \sum_{c=1}^{C} D_{KL}(P_c||\overline{P}) \tag{2.5}$$

Where,

$$D_{KL}(P_c||\overline{P}) = \sum iP_c(i) \log \frac{P_c(i)}{\overline{P}(i)} \tag{2.6}$$

P_c is the prediction probability distribution of the ith committee member, and $\overline{P}$ is the average prediction probability distribution of all members.

It is noteworthy that Query-By-Committee (QBC) requires simultaneous operation of multiple models. Hence, one drawback of methods based on QBC is their slower speed, which is a factor of the committee's size [7]. Speed is paramount for active learning applications, but the trade-off with QBC is that annotators experience prolonged waiting times while the system identifies instances using QBC.

Semantic-based Diversity Sampling

Peng, Hao, et al. [46] introduced a semantic-based diversity sampling method that can be applied in text classification combined with active learning in this experiment. Distinct from the uncertainty sampling approach, the semantic-based diversity sampling method eliminates text sample redundancy semantically using the *Euclidean distance*. This distance measure is frequently used across various domains, including data clustering and nearest neighbour search in embedding spaces. In machine learning, calculating a Euclidean distance from an embedding to a cluster centre can be achieved by computing a distance matrix for every embedding to every cluster centre, where "$distances[i, j]$" represents the Euclidean distance from the ith embedding to the jth cluster centre [6].

To ensure that richer and less redundant samples are provided to the model (learner) in subsequent processes, this abstraction method employs the greedy k-centre clustering algorithm by Sener and Savaresse (2017) [53]. The dataset D contains $n \times m$ unlabelled texts and divides D into n batches, with each sample set containing m instances. It is the result of encoding the dataset. First, select α vectors from Y_i^* to initialise the clusters c_i^0. Here examples will be considered as cluster centres. Then the k-centred algorithm searches u_i^n from $\widetilde{c}_i^0$, which is a set that includes members that are not in Y_i^*. It is the furthest from the centre of all clusters. The algorithm chosen is formulated as follows:

$$\begin{aligned} u_i^0 &= \arg \max_{y_i^k \in \widetilde{c}_i^0} \min_{y_i^j \in c_i^0} \left\| y_i^k - y_i^j \right\|_2^2 \\ u_i^1 &= \arg \max_{y_i^k \in \widetilde{c}_i^1} \min_{y_i^j \in c_i^1} \left\| y_i^k - y_i^j \right\|_2^2 \end{aligned} \tag{2.7}$$

Where:

$$\begin{aligned} \widetilde{c}_i^0 &= Y_i^* \backslash c_i^0 \\ c_i^1 &= c_i^0 \bigcup u_0 \end{aligned} \tag{2.8}$$

This is followed by updating the existing clusters to c_i^1 after a loop execution and merging the output into P. All text instances in P converge into a core set that best represents and generalises the dataset D in the semantic space.

Bayesian Active Learning by Disagreement

Bayesian Active Learning by Disagreement (BALD) is an active learning strategy suitable for text classification problems, combining Bayesian inference with uncertainty measures to select information-rich samples. In BALD, uncertainty is measured by calculating the inconsistency in predictions for each sample's class probability distribution, as proposed by Houlsby, N. (2011) [24]. The learner (model) aims to maximise the uncertainty of the model parameters through the input x. $\mathrm{H}(y \mid x, D)$ represents the uncertainty of the target variable. The second term of the equation represents the expected uncertainty (entropy) of $\mathrm{H}[y \mid x, \theta]$, given that the parameter θ follows the posterior probability distribution $p(\theta \mid D)$ based on the training dataset D, measuring the average uncertainty.

$$x^* = \arg\max_x \mathrm{H}(y \mid x, D) - \mathrm{E}_{\theta \sim p(\theta|D)}[\mathrm{H}[y \mid x, \theta]] \mid \tag{2.9}$$

To better calculate the inconsistency between predictions, Gal et al. proposed a specific sampling function in 2017 that utilises the Monte Carlo Dropout (MCD) method [17]. MCD is a regularisation technique during training, which, by randomly deactivating certain neurons, simulates the posterior distribution of Bayesian networks [43]. This allows for selecting samples with the highest uncertainty for labelling by making multiple predictions for each sample and calculating the inconsistency between the model's predictions.

At its core, MCD is a regularisation method designed to prevent overfitting by probabilistically deactivating a subset of neurons. During the inference process, different results are produced during the prediction phase by inputting into the neural network a finite number of times (T times) and using dropout [60]. Consequently, after quantifying the uncertainty from these varying prediction results, they can be used to estimate the model's uncertainty about unlabeled samples. This is then applied in the Bayesian Active Learning by Disagreement (BALD) sampling strategy to select the most uncertain unlabeled data. The inference results of MCD are derived in the following manner:

$$\overline{y}(x) \approx \frac{1}{N_{mc}} \sum^{N_{mc}} y_{drop}(x) \tag{2.10}$$

Wherein, y_{drop} represents the varying outputs from the dropout-enabled network, x is the input to the network, and N_{mc} is the number of samples required to obtain the distribution. Ultimately, BALD selects the samples with the highest informational gain by comparing each sample's BALD value. These samples are then labelled and added to the annotated training set. Moreover, BALD can be paired with the previously mentioned *Least Confidence*, replacing the original entropy computation, to select samples for which the model prediction probabilities are most uniform (i.e., samples without a particularly high prediction probability for any category).

2.4 Small-Text Library

The text classifier in this project tends to focus on a single model, potentially overlooking the applicability of other feasible models. However, the time-consuming of switching models and active learning strategies, coupled with code redundancy, would significantly impact the progress of the experiment. The Small-Text Library integrates commonly used libraries like scikit-learn, transformers, and PyTorch that can be applied within the Python environment [51].

The pool-based active learning framework for text classification bridges the interfaces of active learning strategies, classifiers, and stopping criteria. It provides an advanced active learning framework for text classification tasks and offers a range of classifier and acquisition function components. These can be combined in various permutations for experimental and applied active learning tasks, facilitating the implementation of active learning in the Python ecosystem. Compared to the popular ModAL library

[10], Small-Text offers more flexible customisation services, with the former focusing on model ensembles and strategy choices. Nonetheless, this all-encompassing approach might not necessarily enhance the performance of the ExpBERT-based project, as performance discrepancies can vary based on circumstances.

2.5 GPT Annotator

GPT (Generative Pre-trained Transformer) is a pre-trained language model developed by the OpenAI team in 2018 [4]. Its key algorithm is the Transformer, a deep neural network structure based on the self-attention mechanism, boasting potent sequence modelling and representation learning capabilities [33]. This model can analyse and generate natural language text through pre-training and fine-tuning, proving beneficial in multiple scenarios such as automatic answering, intelligent customer support, and language translation.

As a result, for a time-saving and efficient text classification system, this project will employ this model as an active learning annotator based on ExpBERT. The primary task is to generate appropriate explanations for ExpBERT and provide them to the model to enhance classification performance. In this configuration, active learning generates human-readable explanations for input text by leveraging GPT's robust generative model [57].

A common approach involves using GPT to find keywords from the sampled text of a specific category, combine the keywords to form explanations, and then choose one or more to integrate into the original explanations. The combined text is then passed to the ExpBERT model. By analysing the sampled text in this manner, one can overcome deficiencies in human memory and the inadequacy of semantic similarity computation, ultimately identifying crucial feature words and phrases that best assist the model in understanding the intrinsic structure of the data.

2.6 Bayesian Neural Networks in Active Learning

In modern machine learning and deep learning applications, Bayesian Neural Networks (BNNs) have emerged as critical components due to their flexible inference mechanisms. BNNs offer a rich and adaptable modelling framework by introducing uncertainties in weights and predictions [35]. This not only bolsters the model's reliability but also enhances its ability to generalise, adapt to noise, and handle outliers. Unlike traditional neural networks where weights are fixed, for BNNs, they are variable, as shown in Figure 2.3. The idea behind the model is to combine the strengths of neural networks with probabilistic modelling, providing probabilistic assurance for predictions [41].

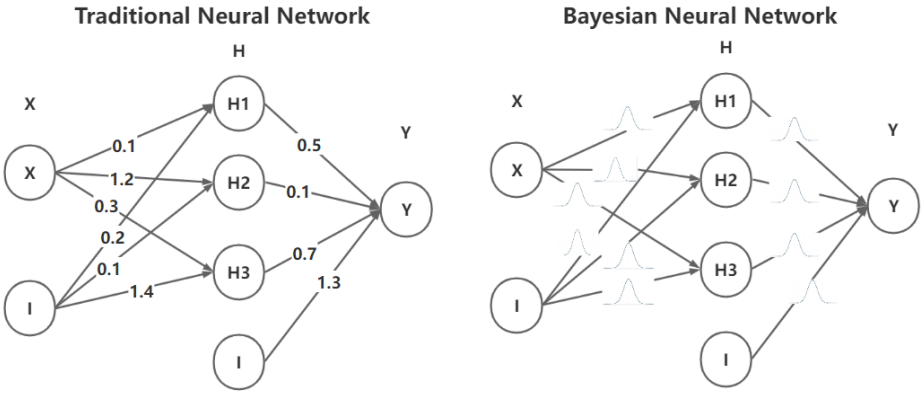

Figure 2.3: Traditional Neural Networks and Bayesian Neural Networks

The weights in this context are inferred based on the observations we have, presenting an inverse probability problem solved through Bayes' theorem [18]. As shown in the equation, the weight ω represents latent variables for which the true distribution is unseen. Bayesian inference allows us to obtain

the distribution of model parameters conditioned on the observed data, $p(\omega \mid D)$, known as the posterior distribution.

$$\pi(\omega \mid D) = \frac{p(\omega)p(D \mid \omega)}{\int p(\omega)p(D \mid \omega)d\omega} = \frac{p(\omega)p(D \mid \omega)}{p(D)} \tag{2.11}$$

Additionally, the likelihood function $p(D \mid \omega)$ in multi-class problems combines the neural network's predictions (after a Log Softmax transformation) with the actual observed class labels to quantify their consistency. The posterior distribution can be computed using Bayesian theory after determining the likelihood and the prior distribution $p(\omega)$.

After obtaining the posterior distribution, marginalisation methods will be used for prediction:

$$\mathbb{E}[f] = \int f(\omega)\pi(\omega \mid D)d\ \omega \tag{2.12}$$

Bayesian Neural Networks provide a way to quantify uncertainty, enabling active learning systems to identify instances on which the model is least confident, subsequently selecting these instances for annotation. However, integrating the model parameters often requires computationally expensive operations, such as Markov Chain Monte Carlo (MCMC) sampling [22]. This can become problematic when dealing with large-scale datasets, especially in the context of active learning, which typically requires multiple iterations and model updates. Therefore, the actual implementation of BNNs may prolong the initialisation time of emergency systems [44].

Chapter 3

Design

This chapter will provide an overview of the active learning framework in Section 3.1, followed by a detailed breakdown and design according to the sequence of tasks in active learning. Starting with pre-training the data in Section 3.2, we then discuss the design of classifier models that acquire embedded representations in Section 3.3. Section 3.4 presents the design of active learning strategies from multiple perspectives using the trained classifier model and analyses the implementation and effects of different acquisition functions. Finally, the design of the annotator's process functionality is covered in Section 3.5.

3.1 Structure Overview

The overall experimental framework is built upon the pool-based active learning technique discussed in Section 2.3.1. The system samples from a fixed pool of unannotated data using a trained model for sample assessment.

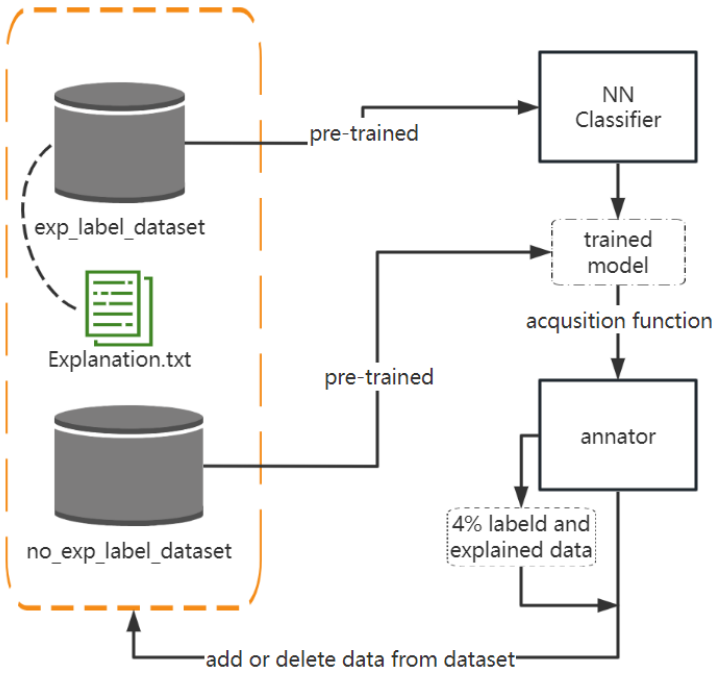

Figure 3.1: The overall experimental framework

As depicted in Figure 3.1, in contrast to traditional active learning methods, this study revises the dataset requirements, introducing two distinct datasets: **one annotated with labels and explana-**

tions (exp_label_dataset) and **another neither labelled nor explained** (no_exp_label_dataset). The 'exp_label_dataset' encompasses data for training, validation, and testing. However, during actual testing and validation, the model only accesses explanations and text embeddings without the corresponding labels. Both explanations and labels are essential during training. To harness the full capabilities of the pre-trained ExpBERT model, each annotation cycle not only provides labels but also appends new explanations to the default explanation set (Explanation.txt).

Throughout the process, three main sets continuously engage in iterations within three primary modules until the evaluation score based on the **validation set** approaches or exceed the evaluation score for the full data volume.These modules are training, annotation, and updating:

Training: During the initial training phase, **18%** of the entire data volume, which comes with labels and explanations, is used. Each sample in the 'exp_label_dataset' is linked to 9 default explanations (label descriptions) for the first round of training. After the first round of training, the **trained model** and the 'no_exp_label_dataset' are used as input parameters for the active learning strategy. The "information content" of each sample is then calculated from perspectives such as uncertainty or diversity. Samples with the highest "information content" are subsequently selected for annotation.

Annotation: In each iteration, the annotator receives **4%** of the samples. During the annotation process depicted in the figure, the annotator identifies key phrases in the text that **semantically resemble** the category description as explanations, and the system automatically provides the classification label for each text. Once samples are fully annotated, they are added to the 'exp_label_dataset', forming the dataset required for the next iteration.

Dataset Update: Subsequently, the annotated data, which constitutes **4%** of the total data volume, is removed from the 'no_exp_label_dataset'. With each iteration, the size of the 'exp_label_dataset' increases by **4%**, and one or more explanations are added. After updating the 'Explanation.txt', the total number of explanations will exceed the initial default set. All these datasets are pre-trained by the ExpBERT-based model before being fed into the neural network text classifier.

3.2 Pre-training

According to the framework design mentioned above, all datasets require a pre-training process before training the text classifier. As outlined in Section 2.1 about ExpBERT, pre-trained models are typically trained on a large volume of text data, capturing deep bidirectional language representations. As a result, they often achieve higher performance on specific tasks. In many text classification scenarios, using a pre-trained model as a foundation and fine-tuning it can significantly reduce the training time of the text classification model.

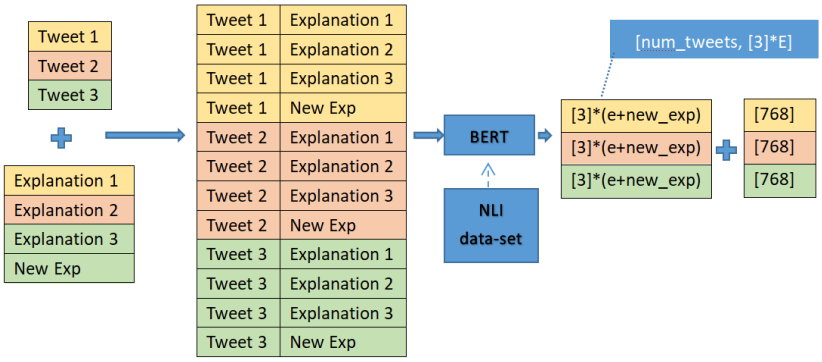

Figure 3.2: Overview of pre-training process

Text is typically input into the model to acquire its embedding representation. This embedding representation can then serve as the input for downstream text classifiers. However, the embedding size

obtained from the original model is often too large, leading to extended computation times. Therefore, as illustrated in Figure 3.2, Natural Language Inference (NLI) is employed to fine-tune BERT, resulting in an embedding size of 3∗E for each sample. To retain tweet information, after concatenating the text embedding representations, the size of a sample's embedding is only **768 plus 3∗E**.

In addition, it is assumed that the number of explanations corresponding to the annotated dataset will increase by 'new_exp' during the active learning iterations. Therefore, every time the iteration reaches the pre-training step, it is necessary to re-connect the updated training set, test set, and validation set texts with all the updated explanations as input to the pre-training model. As a result, each pre-training session will handle an additional **num_tweets∗new_exp** amount of data. This aspect also contributes to the extended duration of the experiment implementation.

For the unannotated dataset, to simulate real-world conditions, only the text data needs to be processed during pre-training. However, the trained neural network expects to receive input with the same dimensions as during training. Thus, for the embedding representation of unannotated data, the 'F.pad(embedding, (0, pad_amount))' method is used to ensure the tensor dimensions of the embedding match. Lastly, pre-training employs a pool-based approach to speed up the computation to enhance the computational efficiency of each iteration. The 'Pool' class from Python's 'multiprocessing' library is utilised to process data batches in parallel. Multiple batches can be processed simultaneously on multi-core machines, allowing for flexible process control and increased processing efficiency.

3.3 Classifier Model

This paper employs neural networks as the classifier model, taking the post-pre-training embeddings as input. The evaluation phase (Chapter 5) will compare the performance of conventional neural network classifiers and neural network classifiers with 'dropout' in conjunction with active learning techniques. Structural modifications will be made to the traditional neural networks in this experiment. The final model structure will be built using PyTorch [26].

3.3.1 Before Dropout

Regarding model architecture, the neural network in this study employs a single hidden layer and 100 neurons. The output feature count, or 'output size', is 9 (number of labels). The forward propagation design can be represented by the following formula, where W_i and b_i are the weights and biases of the linear layer, respectively. After passing through an activation function, the model output y is obtained. This study will use the $ReLU$ non-linear activation function, which induces network sparsity, alleviating overfitting issues [56].

$$h = W_i x + b_i, \quad a = ReLU(h), \quad y = W_{i+1} a + b_{i+1} \tag{3.1}$$

Regarding training methods, the conventional neural network uses the standard loss computation method (cross-entropy [11]). Additionally, the steps of forward propagation, loss computation, backward propagation, and optimisation are executed separately using PyTorch's standard methods.

3.3.2 After Dropout

This paper has improved the model structure by employing a dropout neural network model as a text classifier when applying the BALD method. Hinton mentioned that Dropout can prevent neurons from overly relying on other specific neurons in the network, thereby enhancing their independence and strengthening the model's generalisation capability [23]. Applying dropout to neural networks is equivalent to having a "sparse" network made up of units that have survived the dropout. A neural network with n units can have 2^n possible sub-network configurations. For each presented training instance, a new sparse network is sampled and trained [58].

Dropout effectively approximates different neural network structures by randomly discarding units. The dropout process is illustrated in Figure 3.3, achieved by removing some neurons from the network and breaking their connections. In this paper, the 'dropout_prob' parameter is set to **0.2** during the

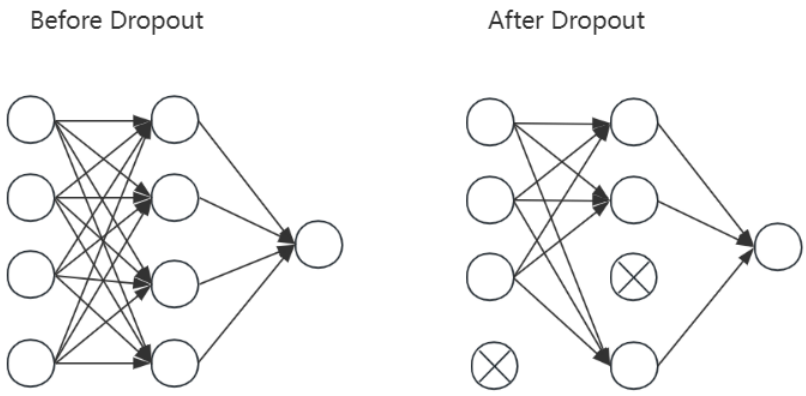

Figure 3.3: Standard Neural Network and Dropout Neural Network.

initialisation of the neural network, meaning that the probability of a neuron being temporarily removed from the network during each forward propagation is 20%. At this point, the forward propagation formula for the original neural network changes:

$$a = ReLU(h), \quad r = Bernoulli(p), \quad aDropout = r * a, \quad y = W_{i+1} * aDropout + b_{i+1} \tag{3.2}$$

$Bernoulli(p)$ represents a random variable following the **Bernoulli distribution**, used to generate a probability vector r, which randomly yields a vector of 0s and 1s. Consequently, some elements in vector a are set to 0 based on this distribution.

3.4 Active Learning Strategy

In Section 3.1, the acquisition function process was mentioned. In the design, **four different active learning strategies** were adopted to try to improve model performance: random sampling as a baseline; uncertainty sampling based on Least Confidence; diversity sampling based on sentiment; and Bayesian Active Learning by Disagreement (BALD) based on MCD and Least Confidence. Each of these strategies explores and analyses the characteristics of the data in different ways, serving as the core of the active learning framework to enhance the efficiency and accuracy of model learning. The best ways to improve model performance are explored by sampling from different perspectives.

Random Sampling

During those sampling processes, the acquisition function will evaluate nearly 70% of the unannotated dataset, and each evaluation provides 4% of the unannotated dataset to annotators for annotation. Random sampling ensures **unbiased** sample selection. The sampling is done directly on the unannotated dataset using Python's built-in random module without taking the trained model into account. The model performance under its influence will serve as a **baseline**.

Uncertainty Sampling Based on Least Confidence

This paper employs the Least Confidence approach for uncertainty estimation. As discussed in Section 2.3.2, the Least Confidence method is often used as an alternative to entropy computation for multi-label text classification problems. In this experiment, the model is set to evaluation mode. The **highest probability** value is sought within the probability distribution 'prob_dist' given by the model. The number of labels 'num_labels' is calculated, and the most confident prediction is normalised using the formula below. The resulting 'normalized_lc' provides a standardised score where a **higher value indicates less confidence** in the sample.

$$\text{normalized\_lc} = (1 - \max(\text{prob\_dist})) \times \left(\frac{\text{num\_labels}}{\text{num\_labels} - 1} \right) \tag{3.3}$$

Samples are ranked in ascending order based on this score, and the top 4% of samples with the **highest scores** are selected as the output for this uncertainty method.

Semantic-based Diversity Sampling

In this study, a semantic diversity-based sampling method is employed. This approach leverages a perspective distinct from uncertainty sampling in the active learning process to identify enriching and minimally redundant samples. For the dataset under consideration, there is a significant imbalance in the number of samples across categories. Some categories account for only 2% of the entire dataset. Such imbalances might lead to certain categories being under-trained, resulting in considerable uncertainty during evaluation. Relying solely on uncertainty for sampling may lead to the over-sampling of a specific category, thereby skewing the distribution of the sampled results, consequently affecting the learning outcome of the model. Hence, a semantic diversity-based sampling strategy is adopted to utilise samples from all categories during active learning effectively.

This method requires loading an unannotated dataset and using the embeddings from a pre-trained model. By setting the trained neural network model to evaluation mode, probabilities corresponding to the input embeddings are computed. As introduced in Section 2.3.2, each text's embedding vector is denoted as Y_{i*}, where i represents the index of the text. Suppose the dataset is divided into n batches, with m instances each. Initially, a random subset of embedding vectors is chosen as the clustering centres, denoted as C_{i*}. In this study, C_{i*} is represented as follows, where the proportion of the initial clustering centres, or α, is set to 0.2:

$$C_{i*} = \text{random\_choice}(Y_{i*}, \alpha * n * m) \tag{3.4}$$

Subsequently, the greedy k-centre algorithm is utilised to search for embedding vectors in Yi that aren't in Ci*. The algorithm identifies embedding vectors that are the **furthest from the existing clustering centres** and updates the set of clustering centres. This process is illustrated as:

$$C_{i*} = \text{argmax}_{y \in Y_{i*} \setminus C} \text{min}_{c \in C_{i*}} \text{dist}(y, c) \tag{3.5}$$

The 'dist' function computes the **semantic distance** between embedding vectors and clustering centres. Finally, we return the indices in Y_{i*} for each cluster centre in C_{i*} denoted as O_i.

MCD Least Confidence BALD

This study adopts a sampling strategy based on Monte Carlo Dropout(MCD) to find samples with higher model uncertainty for more effective model training. The foundation of this strategy is a study by Gal and Ghahramani in 2016 [16], where they proposed using dropout during neural network training as an approximation to Bayesian inference. Similar to other active learning, the embedding representation of the unlabeled dataset is loaded.

Algorithm 3.1: BALD using MCD

```
Set model to TRAINING mode
for input_data in Embeddings do
    Initialise predictions as an empty list
    for j = 1 upto T_samples do
        logits ← model.forward(input_data)
        probabilities ← SOFTMAX(logits)
        APPEND probabilities to predictions
    end
    avg_probabilities ← AVERAGE(predictions)
    bald_score ← NEGATIVE MAXIMUM of avg_probabilities
    APPEND (input_data's index, bald_score) to selected_indices
end
SORT selected_indices by bald_score in DESCENDING order
return top k indices from selected_indices
```

However, the difference is that the text classification model is a neural network with dropout. Also, as shown in Algorithm 3.1, it needs to be **set in training mode**, not evaluation mode. This is because MCD requires the dropout to be active during training. Unlike the traditional BALD strategy, the experiment measures uncertainty by **maximising the negative maximum of the prediction probability** rather than the conventional BALD calculation method. Next, the indices of all samples and their corresponding

BALD scores are stored in a list and sorted in descending order based on the BALD scores. In this way, we can identify the samples for which the model is most uncertain about their classification, i.e., samples with the **highest** BALD scores. This method can be seen as a variant of the BALD algorithm that integrates MCD and the Least Confidence strategy. Compared to uncertainty sampling that only uses the Least Confidence, this method can capture **more stable uncertainty information** from the model.

3.5 Annotation Process

After the sampling guided by the active learning strategy, a set of unannotated texts with high information content is obtained. For these texts, annotators need to provide the corresponding classification based on their content and decide which explanatory notes to add in the current iteration.

3.5.1 Label Annotation

The focus of this experiment is to explore whether the addition of explanatory annotations during the active learning process can enhance the performance of the text classifier. To simulate an ideal situation, it is assumed that in each iteration, the annotators have a complete understanding of the text classification, so the provided labels will **accurately correspond** to the source data. Furthermore, in the evaluation stage of Chapter 5, to eliminate the potential impact of the added labels and text volume on model performance, we will compare two scenarios: one with active learning that only added labels and another with active learning that added both labels and explanations.

3.5.2 Explanation Annotation

This experiment designed three different explanation annotation strategies. The provided explanations will be stored in the 'Explanation.txt' set displayed in Figure 3.1, available for the text concatenation task prior to subsequent pre-training. System operators can choose among these three methods based on workforce allocation or cost control to better support the model's learning process:

The first strategy involves previewing the texts in each iteration in advance and simulating the possible added explanations, just as an annotator would. The **preset explanations** will be saved in a 'txt' file. When the annotation phase is executed, explanations are automatically extracted from this file. Although this method can reduce workforce consumption, it lacks adaptability and is only suitable for the dataset related to this experiment.

The second strategy allows **humans** to provide input as explanations directly. This approach requires users to analyse the features of the selected text and provide their explanations. In each iteration, the annotator will focus on ten prominent texts of a particular category to quickly analyse these texts. This method incorporates human judgment and applies to other emergency-related data, offering some flexibility.

The last strategy uses **OpenAI's model** to automatically provide explanations, reducing the need for human effort. This experiment chose the **'text-davinci-003'** engine for text analysis [63]. Through the developer platform's **Playground** on the OpenAI interface, the prompt to be input into the model can be simulated in the Playground. The prompt uses *"generate a string that only contains the words in the text that are only semantically equal to"* to connect the active learning strategy-selected prominent texts of a particular category. On the right navigation bar of the Playground, one can select the 'Model', 'Temperature', and maximum response length. The best-performing hyperparameters are chosen based on the reasonableness of the response. As shown in Table 3.1, for the classification "deaths and injured", tests found that the 'text-davinci-003" responses were far more reasonable and accurate than those of "text-ada-001". Other configurations tended to generate irrelevant information like geographical locations and numbers.

After debugging, it was found that when the **'Temperature'** (the lower the temperature, the less randomness) is set to 1 and **'Top_P'** (which controls diversity via nucleus sampling) is set to 1, the

Model	Temperature	Top P	Response
text-ada-001	0.5	1	"As many as *98%* of those killed .. *Pakistan Somalia*"
text-ada-001	1.0	1	"Hundreds dead in *Pakistan..* country *6.0* quake, dozens hurt"
text-ada-001	1.0	0.5	"As many as *98%* of those killed strikes are civilians."
text-ada-001	1.5	1	"patrol western shuts the exchange *peace 65 party*"
text-ada-003	0.5	1	"Hurt Death *Pakistan* Quake Dozens Killed Loved Ones"
text-ada-003	1.0	1	**"dozens hurt killed families loved ones RIP injured"**
text-ada-003	1.0	0.5	"Hurt, Dozens, Killed, *Pakistan*, Families, Loved, Injured"
text-ada-003	1.5	1	"death Mortality over *320 people* killed drone *98%* victims"

Table 3.1: Results of the OpenAI model evaluation.

model's output is most reasonable. The Playground provides configuration code for the Python environment, allowing the code to be directly applied to the system. To successfully connect to the OpenAI interface, it is necessary to apply for an **OpenAI key** and load this key before generating responses.

It can be observed that using this model can analyse texts, selecting keywords that are semantically closest to the category label description to concatenate into a string. Ultimately, it returns a string that best reflects the characteristics of the text and helps the model learn from it as an added explanation. However, its drawback is that it's not friendly for the demand of generating multiple explanations at once or for active learning of hundreds of tokens. Although it reduces workforce costs, the generation of each token consumes certain funds.

Chapter 4

Implementation

This chapter primarily discusses the setup required to conduct the experiment. Section 4.1 introduces the fundamental environment in which the experiment is conducted, and Section 4.2 details the preparation and splitting of the dataset used in the experiment. Additionally, the performance metrics (Section 4.3) used in the evaluation in Chapter 5 and the rationale behind their selection are explained. Lastly, the stopping criteria adopted for the experiment (Section 4.4) and the execution modules (Section 4.5) are specified.

4.1 Experimental Environment

This experiment employs **Python 3.9** as the primary development language for natural language processing tasks. Being highly suitable for natural language processing tasks, Python boasts a plethora of data science and machine learning libraries [28]. This study utilised the capabilities provided by libraries such as PyTorch, Pandas, and Numpy for data analysis, pre-processing, and machine learning [Reference]. Python's high compatibility facilitates the connection to remote High-Performance Computing Systems (HPC) within the context of this study using PyCharm, or running through Google Colab in conjunction with Google Drive. This research integrates **PyTorch** with Python to construct dynamic neural networks. By leveraging PyTorch's automatic gradient computation, the features of Dynamic Computational Graph, and the capability of CUDA for accelerated computation, a high level of flexibility and accuracy is maintained in the neural network model [45].

4.2 Dataset

The dataset used in this experiment simulates the text distribution of social media in emergencies. The dataset is prepared and split according to the framework described in section 3.1 to meet the training requirements with a small amount of data in active learning.

4.2.1 CrisisNLP

In this experiment, the CrisisNLP dataset is utilised, which comprises data IDs, textual information, and descriptions of their corresponding categories. There are nearly 16,000 entries in total. This dataset requires preprocessing, splitting, pre-training, and classification. CrisisNLP is an open-source resource designed explicitly for humanitarian emergency and crisis response related natural language processing (NLP) research and development [27]. The dataset contains interaction information from various large social media platforms, covering multiple crisis events, such as typhoons, earthquakes, and other events that require timely responses from relevant departments. The nature of the information can be divided into nine categories. The descriptions of these labels and their distribution are shown in the Table 4.1.

According to the distribution in Table 4.1, it is evident that the data proportion is **highly imbalanced**. Label 7 has the highest proportion, and the information it provides is also the most mixed,

Label	Description	Percentage
0	injured_or_dead_people	13%
1	missing_trapped_or_found_people	2%
2	displaced_people_and_evacuations	3%
3	infrastructure_and_utilities_damage	8%
4	donation_needs_or_offers_or_volunteering_services	14%
5	caution_and_advice	6%
6	sympathy_and_emotional_support	11%
7	other_useful_information	30%
8	not_related_or_irrelevant	13%

Table 4.1: Data distribution in CrisisNLP.

accurately reflecting the distribution of emergency information in real-life scenarios. This poses a challenge for text classification tasks. Therefore, this dataset is of great value to researchers, as it documents real-time reactions and behaviours of people in various crisis situations and their corresponding data distribution. Consequently, researchers can develop more effective systems tailored for genuine data imbalances, addressing these crises, such as automatically detecting crisis-related social media posts or predicting the development trend of a crisis through analysing social media data.

4.2.2 Data Preparation

Preprocessing: Before feeding into the pre-trained model, preprocessing the tweets in the dataset is a critical step. The goal is to minimise the noise in the text and enhance the model's learning efficiency. Firstly, internet slang is standardised, converting informal abbreviations or colloquialisms into their standard forms and splitting camel-case named words. Subsequently, emojis in tweets are replaced with their textual counterparts using the emoji library, enabling the model to better understand these symbols' meanings. Lastly, website links, usernames, and "#" tags in tweets are cleaned or replaced since these elements typically don't contribute much to the model's learning. Such preprocessing steps help in reducing noise in tweets, thereby enhancing the model's learning efficiency and performance.

Dataset Splitting: For active learning tasks, splitting the dataset into unannotated and annotated datasets is essential. At the same time, the annotated dataset will be further split into training, validation, and testing sets. Active learning requires adding or removing data in each iteration. However, to ensure that the model's performance on unseen data can be tested fairly in each active learning iteration, the number of test samples must **remain constant**. Thus, the test set, unannotated and annotated datasets must be split **outside of the active learning loop**. The **training and validation sets will expand** with each iteration as the dataset grows, but their relative proportions remain unchanged. Since the data corresponding to label 1 only account for about 2% of the total dataset, to ensure that all data subsets contain data corresponding to label 1, we use the 'StratifiedShuffleSplit' mechanism from the scikit-learn library to maintain the original data's class distribution. Compared to random splitting, this method provides a more accurate assessment of the classifier's performance.

Data	**Count**		**Percentage**
Annotated Data	Training dataset	2890	18%
	Validation dataset	331	2%
	Test dataset	1606	10%
Unannotated Data	11241		70%

Table 4.2: Distribution of datasets

To meet the criteria of active learning, which achieves high performance with a small amount of training data, the number of annotated data in the first iteration of active learning is not the total data. As shown in Table 4.2 above, the total is **only 30%** of the entire dataset, with a length of 4827. Therefore, the size of the training set is no longer the most significant component, with the training set representing only **18%** of the total data for the active learning strategy and the size is set to 2890. The size of the unannotated dataset is set to **11241**, which accounts for 70% of the total dataset. The unannotated

dataset becomes the most significant component compared with other datasets.

When training with the **full dataset** (with the training set having the highest proportion), the combined length of the training and validation datasets in the annotated data will be 12845. The number of the test set remains unchanged both in the full dataset and in the amount of data required for active learning, with a length of 1606. 4% of the data is sampled for annotation in each cycle. In the subsequent evaluation chapter, it was found that after 9 iterations, the model performance approached the model's performance corresponding to the validation set under the full data. At this time, the length of the unannotated dataset remains 6741.

4.3 Performance Metrics

The setting of performance metrics can significantly influence the effectiveness of active learning projects. Proper evaluation criteria can measure and compare the impacts of different active learning strategies and model frameworks on emergency systems in human learning. This, in turn, determines the choices of models, active learning strategies, and parameters. To assess the performance of active learning, the data is first partitioned according to the standards in Section 4.2.2. Then, the training begins on the training set. After each pool-based active learning iteration round, the trained model is evaluated on the test and validation sets using the predefined performance metrics. After each iteration, learning curves are plotted based on the performance metrics.

To comprehensively assess the model's performance, the experiment utilises two performance metrics: **accuracy and F1 score**. Accuracy measures the proportion of instances that the model correctly predicts relative to the total number of instances, and it can be a biased evaluation criterion. In contrast, the F1 score is the harmonic mean of precision and recall. Compared to accuracy, it eliminates bias and is more commonly used in multi-class problems [21].

$$F1\_score = \frac{2}{(1/Precision + 1/Recall)} \tag{4.1}$$

Precision is the proportion of positive instances correctly predicted by the model out of all instances predicted as positive, and Recall is the proportion of actual positive instances that the model correctly predicts as positive. When dealing with imbalanced datasets like CrisisNLP, the F1 score can provide a more comprehensive measure of performance.

In addition, the **learning curve** can also serve as a performance evaluation criterion. By observing the changes in the learning curve, we can better understand the model's learning capacity and stability. Tensorboard records the learning curve during the active learning iteration process. The learning curve depicts the relationship between the model's performance and the number of samples used for training (number of active learning iterations), evaluating whether increasing the number of annotated datasets can effectively enhance the model's performance. Moreover, the experiment also logs the performance of the neural network classifier for each epoch during each training process through Tensorboard, ensuring the model achieves the best performance without overfitting after each active learning iteration.

4.4 Stopping Strategy

Within the active learning framework, the Stopping strategy is used to determine when to halt the learning process or no longer request new data from annotators [31]. The design of the Stopping strategy directly affects the cost of annotation, time consumption, and the final performance of the model. Although the Stopping strategy is somewhat akin to a model's hyperparameters, it doesn't directly influence the model's training process. Instead, its primary objective is to reduce annotation costs and save time while maintaining optimised performance. This strategy aims to balance time efficiency and performance output to achieve a learning process that is both efficient and high-performing.

For the training of neural network models, the configuration of hyperparameters directly impacts the training process and model performance. In early research, there have been studies that meticulously

explored and determined various hyperparameters. Here, we referenced these studies and selected hyperparameters suitable for neural network training, namely using a batch size of 8 and a learning rate of 5e-5.

When devising the stopping strategy for the active learning experiment, monitoring whether the model performance has reached saturation is essential. For this experiment, the primary focus is on the average F1 score of the model on the **validation set**. Specifically, we aim to observe when the model, based on a random sampling strategy, can surpass the performance of the model trained on the entire dataset or when the rate of performance growth begins to decelerate.

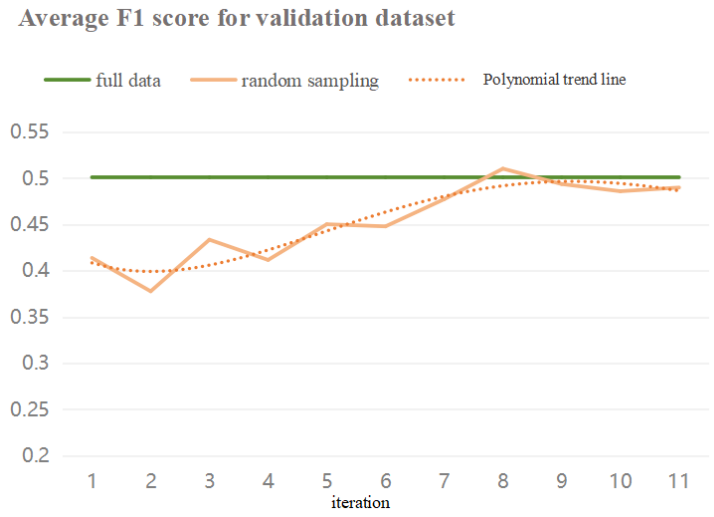

Figure 4.1: Average F1 score of full-data and random sampling strategies under validation dataset

Figure 4.1 reflects the performance comparison between the models trained under the **random sampling** and **full-data** training strategies. By the 8th iteration, the model's performance based on random sampling had already exceeded that of the model trained on the full data. However, this performance improvement was inconsistent, so we continued iterating. After the **9th iteration**, the growth in model performance began to slow down. Considering that further iterations would increase time and annotation costs, we terminated the experiment after the 9th iteration. This strategy allows for **achieving optimal performance at minimal costs**.

4.5 Execution Module

In this study, we conducted text classification by implementing a series of active learning strategies. The experiment is run either by connecting to a remote environment through PyCharm or locally. In most cases, an explanation is added in each iteration, with a total of 9 iterations being performed. At the conclusion of these iterations, the experiment for each active learning strategy took a total of **36 hours**. Using the MCD combined with the BALD active learning strategy, the time taken approaches 37 hours. The experimental execution module is primarily segmented based on active learning strategies. As the evaluation in Chapter 5 found that diversity sampling performed the best, different explanation annotation methods were employed to further explore the impact of explanations on performance.

- Random Sampling: Implemented through the "Whole_train.py" script, processing both full data mode and active learning mode.
- Senmatic-based Diversity Sampling: Based on the "diversity_sampling_whole.py" script, this method employs three different explanation annotation methods: preset explanations, manual input, and explanations generated by the OpenAI model (**an API key for OpenAI is required**).
- Uncertainty Sampling: Using the "uncertaintyWhole.py" script, the aim of this strategy is to select data points that present the most uncertainty or challenge for model classification by calculating the minimum confidence level.

- MCD-based BALD Strategy: Executed via the "BALD_MCD.py" script, it combines dropout neural networks with the Bayesian Active Learning by Disagreement (BALD) strategy, specifically seeking high uncertainty data points in model predictions through minimum confidence.

The primary focus of the experimental execution is to conduct a deep evaluation of the performance of various strategies in text classification tasks. This aims to ascertain which strategy and explanation annotation method is best suited to enhance the model's learning efficiency.

Chapter 5

Critical Evaluation

This chapter first sets out the expectations and framework for evaluation (Section 5.1). Based on five expectations, combined with the evaluation framework and the performance metrics provided in Section 4.3, the evaluation is carried out (Sections 5.2, 5.3, 5.4, 5.5, and 5.6). Results corresponding to each expectation are presented, and the evaluation results are analysed to see if the outcomes align with the preset expectations. Lastly, Section 5.7 provides a summary of the evaluation results.

5.1 Expectations and Evaluation Framework

Initially, there are several preset expectations for the Human-in-the-loop framework.

- Pool-based active learning can effectively enhance model performance (on both test and validation sets) by increasing the size of the annotated dataset.
- The addition of explanation annotations during the annotating process has a greater impact than using data with label-only annotations.
- Most strategies can surpass the baseline (random sampling) while exceeding or approaching the full-data dataset's performance before implementing the stopping strategy.
- We can identify the best active learning strategy, explanation annotation strategy, and model architectural strategy to quickly achieve high performance with a minimal amount of training data.
- The number and quality of the explanations provided greatly influence model performance.

Notably, the full-data dataset refers to the annotated dataset (with nine default explanations linked), where the training set makes up 72% of the data distribution, and the remaining percentage is divided between the validation and test sets. However, the training set share in active learning is 18%.

Based on these expectations, an evaluation framework is established in Table 5.1, which will be applied to both the test and validation sets using F1 score and accuracy as the performance metrics:

ID	Model structure	Dataset	AL strategy	annotation method
1	No dropout	full-data dataset	None	without explanation
2	No dropout	AL-based datasets	Random sampling	without explanation
3	No dropout	AL-based datasets	Random sampling	explanation-NoOpenAI
4	No dropout	AL-based datasets	Uncertainty sampling	explanation-NoOpenAI
5	No dropout	AL-based datasets	Diversity sampling	explanation-NoOpenAI
6	No dropout	AL-based datasets	Diversity sampling	explanation-WithOpenAI
7	With dropout	AL-based datasets	BALD	explanation-NoOpenAI
8	No dropout	AL-based datasets	Diversity sampling	two explanations each iteration

Table 5.1: Evaluation structure.

5.2 Expectation One

According to the expectations set in Section 5.1, a point-by-point evaluation is carried out. For the first expectation, it's necessary to observe whether active learning, by increasing the annotated dataset, can effectively enhance the model's performance. Hence, the learning curve mentioned in Section 4.3 needs to be inspected. The expectation is that within the learning curve, the model's performance consistently rises as the number of annotated instances increases. To reduce computational time, a representative evaluation method was chosen, specifically the one with **evaluation ID 2** from the evaluation framework (Table 5.1) in Section 5.1, employing a random sampling strategy. In each round, only the corresponding classification label is annotated without adding new explanations, instead using the default nine explanations.

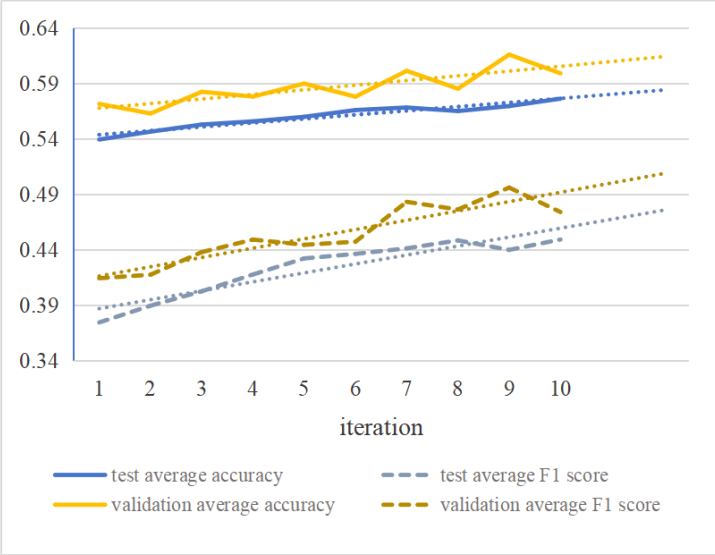

Figure 5.1: Random sampling only annotates labels

The evaluation results are illustrated in figure 5.1. The solid line represents the average accuracy for the validation and test sets, while the dashed line represents the average F1 score. By setting a linear trendline, it's evident that the learning curve's trajectory **aligns with the first expectation**. Simultaneously, the data's performance on the validation set is generally **better** than on the test set.

This is due to active learning, as mentioned in section 2.3, continuously supplying a portion of the annotated dataset to the original dataset for training. As a result, with the increase in input data volume, the classifier can learn more effectively from the abundant data and enhance its generalisation ability. The reason the performance on the validation set is better than that on the test set is evident from Table 4.2, which shows that the overall quantity of the validation set is smaller. Hence, there is less noise and fewer outliers. To simulate real-world scenarios, the test set is larger and noisier, leading to slightly lower overall performance.

In summary, this Human-in-the-loop pool-based active learning framework indeed effectively boosts the model's performance, whether on the test set or the validation set.

5.3 Expectation Two

In Section 5.1, the second expectation investigates the effectiveness of the Human-in-the-loop framework when based on ExpBERT. It explores whether, during the annotation process, adding explanations in addition to labels can lead to higher performance. Thus, we utilise the evaluation methods corresponding to **evaluation IDs 2 and 3** from Table 5.1 for a comparative experiment. We examine whether the results of evaluation ID 3 **(with added explanations)** are superior to those of evaluation ID 2 **(with**

only added labels). In the evaluation process for ID 3, the number of explanations increased by one in each round. Here, the evaluation results use the average F1 scores and average accuracy scores for the validation and test sets. The performance from the final iteration is displayed in Table 5.2.

	Average F1 score	**Average accuracy score**
Evaluation 2 for validation set	0.6159	0.599
Evaluation 3 for validation set	0.625	0.6131
Evaluation 2 for test set	0.4489	0.5761
Evaluation 3 for test set	0.5056	0.6216

Table 5.2: Results of evaluation ID 2 and evaluation ID 3

From Table 5.2, it is evident that all evaluation results for ID 3 (where explanations were added along with label annotations) are **higher** than those for ID 2 (where only labels were annotated). This is because, as mentioned in Section 1.1, **explanations** include keywords and sentences related to classification information. Adding these explanation links allows the model to learn from a deeper semantic level. After pre-training with the ExpBERT mechanism, the received embeddings enhance the classifier's generalisation ability. Therefore, the model's performance is higher in the evaluation framework for ID 3.

Thus, annotating with not just labels but also adding explanation annotations can effectively enhance the classification model's performance. The conclusion is that the second expectation has been met.

5.4

Chapter 6

Conclusion

A compulsory chapter, roughly 10% of the total page-count

The concluding chapter(s) of a dissertation are often underutilised because they're too often left too close to the deadline: it is essential to allocate enough time and attention to closing off the story, the narrative, of your thesis.

Again, there is no single correct way of closing a thesis.

One good way of doing this is to have a single chapter consisting of three parts:

1. (Re) summarises the main contributions and achievements, in essence summing up the content.
2. Clearly state the current project status (e.g., "X is working, Y is not") and evaluate what has been achieved with respect to the initial aims and objectives (e.g., "I completed aim X outlined previously, the evidence for this is within Chapter Y"). There is no problem including aims which were not completed, but it is important to evaluate and/or justify why this is the case.
3. Outline any open problems or future plans. Rather than treat this only as an exercise in what you *could* have done given more time, try to focus on any unexplored options or interesting outcomes (e.g., "my experiment for X gave counter-intuitive results, this could be because Y and would form an interesting area for further study" or "users found feature Z of my software difficult to use, which is obvious in hindsight but not during at design stage; to resolve this, I could clearly apply the technique of Bloggs *et al.*.

Alternatively, you might want to divide this content into two chapters: a penultimate chapter with a title such as "Further Work" and then a final chapter "Conclusions". Again, there is no hard and fast rule, we trust you to make the right decision.

And this, the final paragraph of this thesis template, is just a bunch of citations, added to show how to generate a BibTeX bibliography. Sources that have been randomly chosen to be cited here include:

Bibliography

[1] Charu C Aggarwal, Xiangnan Kong, Quanquan Gu, Jiawei Han, and S Yu Philip. Active learning: A survey. In *Data classification*, pages 599–634. Chapman and Hall/CRC, 2014.

[2] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *Ai Magazine*, 35(4):105–120, 2014.

[3] Dana Angluin. Queries and concept learning. *Machine learning*, 2:319–342, 1988.

[4] Amos Azaria. Chatgpt usage and limitations. 2022.

[5] Yoram Baram, Ran El Yaniv, and Kobi Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5(Mar):255–291, 2004.

[6] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[7] Michael Bloodgood. Support vector machine active learning algorithms with query-by-committee versus closest-to-hyperplane selection. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 148–155. IEEE, 2018.

[8] Robert Burbidge, Jem J Rowland, and Ross D King. Active learning for regression based on query by committee. In *Intelligent Data Engineering and Automated Learning-IDEAL 2007: 8th International Conference, Birmingham, UK, December 16-19, 2007. Proceedings 8*, pages 209–218. Springer, 2007.

[9] Ido Dagan and Sean P Engelson. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, pages 150–157. Elsevier, 1995.

[10] Tivadar Danka and Peter Horvath. modal: A modular active learning framework for python. *arXiv preprint arXiv:1805.00979*, 2018.

[11] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134:19–67, 2005.

[12] Rati Devidze, Farnam Mansouri, Luis Haug, Yuxin Chen, and Adish Singla. Understanding the power and limitations of teaching with imperfect knowledge. *arXiv preprint arXiv:2003.09712*, 2020.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] John J Dudley and Per Ola Kristensson. A review of user interface design for interactive machine learning. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 8(2):1–37, 2018.

[15] Yifan Fu, Xingquan Zhu, and Bin Li. A survey on instance selection for active learning. *Knowledge and information systems*, 35:249–283, 2013.

[16] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[17] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International conference on machine learning*, pages 1183–1192. PMLR, 2017.

[18] Ethan Goan and Clinton Fookes. Bayesian neural networks: An introduction and survey. *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018*, pages 45–87, 2020.

[19] Shantanu Godbole, Abhay Harpale, Sunita Sarawagi, and Soumen Chakrabarti. Document classification through interactive supervision of document and term labels. In *Knowledge Discovery in Databases: PKDD 2004: 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa, Italy, September 20-24, 2004. Proceedings 8*, pages 185–196. Springer, 2004.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[21] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*, 2020.

[22] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[23] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[24] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

[25] Rong Hu, Brian Mac Namee, and Sarah Jane Delany. Active learning for text classification with reusability. *Expert systems with applications*, 45:438–449, 2016.

[26] Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. Pytorch. *Programming with TensorFlow: Solution for Edge Computing Applications*, pages 87–104, 2021.

[27] Muhammad Imran, Prasenjit Mitra, and Carlos Castillo. Twitter as a lifeline: Human-annotated twitter corpora for nlp of crisis-related messages. *arXiv preprint arXiv:1605.05894*, 2016.

[28] Akhil Kadiyala and Ashok Kumar. Applications of python to evaluate environmental data science problems. *Environmental Progress & Sustainable Energy*, 36(6):1580–1586, 2017.

[29] Ross D King, Kenneth E Whelan, Ffion M Jones, Philip GK Reiser, Christopher H Bryant, Stephen H Muggleton, Douglas B Kell, and Stephen G Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, 2004.

[30] K Lang and E Baum. Query learning can work poorly when a human oracle is used. ieee intl. In *Joint Conference on Neural Networks*, 1992.

[31] Florian Laws and Hinrich Schütze. Stopping criteria for active learning of named entity recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 465–472, 2008.

[32] David D Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. In *Acm Sigir Forum*, volume 29, pages 13–19. ACM New York, NY, USA, 1995.

[33] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 2022.

[34] Bill MacCartney. *Natural language inference*. Stanford University, 2009.

[35] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[36] Andrew McCallum, Kamal Nigam, et al. Employing em and pool-based active learning for text classification. In *ICML*, volume 98, pages 350–358. Citeseer, 1998.

[37] Richard McCreadie, Cody Buntain, and Ian Soboroff. TREC incident streams: Finding actionable information on social media. 2019.

[38] Chris J Michael, Dina Acklin, and Jaelle Scheuerman. On interactive machine learning and the potential of cognitive feedback. *arXiv preprint arXiv:2003.10365*, 2020.

[39] Robert Munro Monarch. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster, 2021.

[40] Eduardo Mosqueira-Rey, Elena Hernández-Pereira, David Alonso-Ríos, José Bobes-Bascarán, and Ángel Fernández-Leal. Human-in-the-loop machine learning: A state of the art. *Artificial Intelligence Review*, 56(4):3005–3054, 2023.

[41] Vikram Mullachery, Aniruddh Khera, and Amir Husain. Bayesian neural networks. *arXiv preprint arXiv:1801.07710*, 2018.

[42] Shikhar Murty, Pang Wei Koh, and Percy Liang. Expbert: Representation engineering with natural language explanations. *arXiv preprint arXiv:2005.01932*, 2020.

[43] Tomoyuki Myojin, Shintaro Hashimoto, and Naoki Ishihama. Detecting uncertain bnn outputs on fpga using monte carlo dropout sampling. In *Artificial Neural Networks and Machine Learning–ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15–18, 2020, Proceedings, Part II 29*, pages 27–38. Springer, 2020.

[44] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[46] Hao Peng, Shixin Guo, Dandan Zhao, Yiming Wu, Jianming Han, Zhe Wang, Shouling Ji, and Ming Zhong. Query-efficient model extraction for text classification model in a hard label setting. *Journal of King Saud University-Computer and Information Sciences*, 35(4):10–20, 2023.

[47] Sumanth Prabhu, Moosa Mohamed, and Hemant Misra. Multi-class text classification using bert-based active learning. *arXiv preprint arXiv:2104.14289*, 2021.

[48] Alessandro Prest, Cordelia Schmid, and Vittorio Ferrari. Weakly supervised learning of interactions between humans and objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):601–614, 2011.

[49] Gonzalo Ramos, Christopher Meek, Patrice Simard, Jina Suh, and Soroush Ghorashi. Interactive machine teaching: a human-centered approach to building machine-learned models. *Human–Computer Interaction*, 35(5-6):413–451, 2020.

[50] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

[51] Christopher Schröder, Lydia Müller, Andreas Niekler, and Martin Potthast. Small-text: Active learning for text classification in python. *arXiv preprint arXiv:2107.10314*, 2021.

[52] Christopher Schröder and Andreas Niekler. A survey of active learning for text classification using deep neural networks. *arXiv preprint arXiv:2008.07267*, 2020.

[53] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[54] Burr Settles. Active learning literature survey. 2009.

[55] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[56] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.

[57] Yucheng Shi, Hehuan Ma, Wenliang Zhong, Gengchen Mai, Xiang Li, Tianming Liu, and Junzhou Huang. Chatgraph: Interpretable text classification by converting chatgpt knowledge to graphs. *arXiv preprint arXiv:2305.03513*, 2023.

[58] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[59] John E Tomaszewski. Overview of the role of artificial intelligence in pathology: the computer as a pathology digital assistant. In *Artificial intelligence and deep learning in pathology*, pages 237–262. Elsevier, 2021.

[60] Evgenii Tsymbalov, Maxim Panov, and Alexander Shapeev. Dropout-based active learning for regression. In *Analysis of Images, Social Networks and Texts: 7th International Conference, AIST 2018, Moscow, Russia, July 5–7, 2018, Revised Selected Papers 7*, pages 247–258. Springer, 2018.

[61] Devis Tuia, Michele Volpi, Loris Copa, Mikhail Kanevski, and Jordi Munoz-Mari. A survey of active learning algorithms for supervised remote sensing image classification. *IEEE Journal of Selected Topics in Signal Processing*, 5(3):606–617, 2011.

[62] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[63] Xufeng Zhao, Mengdi Li, Cornelius Weber, Muhammad Burhan Hafez, and Stefan Wermter. Chat with the environment: Interactive multimodal perception using large language models. *arXiv preprint arXiv:2303.08268*, 2023.

Appendix A

An Example Appendix

Content which is not central to, but may enhance the dissertation can be included in one or more appendices; examples include, but are not limited to

- lengthy mathematical proofs, numerical or graphical results which are summarised in the main body,
- sample or example calculations, and
- results of user studies or questionnaires.

Note that in line with most research conferences, the examiners are not obliged to read such appendices.