Active learning and ExpBERT-based text classification for Crisis Scenarios

1 Abstract

Text classification models use natural language processing to analyse pre-trained text and assign labels. In crisis scenarios such as floods and earthquakes, text classifiers can be used to identify emergency information and forward categories of information on social media to the relevant agencies. However, the effectiveness of text classifiers relies heavily on large amounts of training data, which can be scarce and difficult to obtain [1]. In addition, training large amounts of labelled and unrepresentative data can delay the response time of the model as well as affect the accuracy of the model. Identifying actionable types of information, such as search and rescue requests, remains challenging.

Integrating human judgement and interpretation into the classifier training process and engaging the human-in-the-loop (HITL) process using active learning strategies for multi-classification problems can be an effective way to address these limitations. In conjunction with a previous technique, Representation Engineering with Natural Language Interpretation (ExpBERT), which not only relies on the input text itself, but also utilises Natural Language Interpretation (NLE) to improve the representation engineering in order to improve the performance of the classifiers [2]. The main goal of this thesis is to achieve the maximum achievable accuracy of an expbert-based text classifier by adding a small number of representative samples to the classification process, combined with human-provided interpretations [3].

This thesis explores the effectiveness of active learning for the ExpBERT-based text classification problem. During the iterative process, representative unlabelled instances are queried using sampling strategies such as uncertainty and diversity, and the annotator accepts and processes these instances after sampling. The original small amount of annotated data combined with the newly processed valid data will be used for retraining the model, and the final model using a small amount of data after a small number of iterations can achieve the performance of a model trained on a large amount of data. This experiment will create a comparison group to see the effectiveness, compare the impact of different sampling methods, and the performance of the active learning system using Monte Carlo with dropout on the CrisisNLP dataset.

The main conclusion of this paper is:

Models in Active Learning using a small amount of labelled data and a small number of iterations approached or exceeded the performance of models with the full amount of data.

Diversity sampling and Bayesian Active Learning by Disagreement sampling with dropout resulted in higher average model performance in each category after collecting explanations and training the model.

Uncertainty sampling has the most effect on classification performance for the category 'caution and advice'.

1 Introduction

This chapter begins with an introduction to the background of emergency response systems, generating the motivation for the experiment based on the background and emphasising the importance of this research. In addition, the work will be summarised, and the limitations of the traditional approach will be compared. Finally, the main objectives, as well as the challenges, are briefly summarised.

1.1 Background.

The focus of emergency time response systems for social platforms has mainly centred on creating better text classification algorithms to learn from small amounts of data. However, obtaining valuable annotated datasets can prove difficult [4]. ExpBERT, a pre-trained model for text classifiers, receives the classification labels and the corresponding explanations. The explanations include which keywords led to this classification.ExpBERT uses this information to incorporate it into the training of the model, allowing the model to learn from deeper semantic information to improve the generalisation of the model. Therefore, it is the use of representative data and understanding of keywords and phrases followed by annotation that can guide the model to accurate classification.

However, the throughput of general social platform data cannot complete annotating tens of thousands of data. As a result, many forms of active learning that enable the sampling of high information value data at low cost have been widely used in classification projects [5,6]. Active learning is an effective way to solve these problems by selecting a small number of highly informative unannotated samples available for human experts in the loop to annotate [7,8]. The new data after getting effective annotations determine the model's performance in the next iteration, so querying the most informative new instances using different acquisition functions is probably the most popular approach in active learning. Therefore, the query strategy has naturally become the focus of research in the field of active learning.

Meanwhile, neural network text classifiers are often not adapted to early uncertainty sampling [9, 10]. This is because the neural network weight parameters are fixed values, leading to overconfidence in the model's prediction of correct and incorrect data. However, by using dropout, model uncertainty can be introduced to some extent. During training, the dropout layer randomly "turns off" a portion of the input units, and this randomness makes the network weights no longer definite values, thus mimicking the uncertainty of the weights.

1.2 Motivation

Against the above background, this study focuses on the accuracy, Representativeness, and robustness of text classification systems.

1.2.1 Accuracy

The unifying feature of social networks represented by Twitter is the vast amount of data combined with Internet syntax. The ultimate goal is to capture urgent needs that require accurate categorisation of critical message categories while dispatching the relevant authorities to address the needs. However, the traditional text classification system is a performance enhancement done through extensive supervised learning with high costs and slow response time. Texts with different amounts of information have random distributions. The difference in the amount of information significantly affects the model's accuracy, and the results of capturing and recognising important emergency messages are mostly poor. Therefore, using active learning models to apply small amounts of data and improve accuracy is essential.

1.2.2 Representativeness

The ability to extract rich feature representations in large-scale unlabelled data reduces most of the bias in the emergency multiclassification active learning problem. By actively selecting samples for labelling, the model can select the most valuable and representative samples in each iteration to improve model performance. Therefore choosing an effective ACQUISITION FUNCTION to extract representative data has the most significant impact on performance and is the most worthy of improvement.

1.2.3 Soundness

The original model often shows a tendency to be overconfident when dealing with text categorisation tasks. This causes the model to make incorrect and unreliable predictions when fed with unknown content. And in many real-world applications, such as disaster response and medical judgement, inaccurate predictions can have serious consequences. Therefore, effective estimation of the model's uncertainty to reduce its tendency to be overconfident has become an important research topic. On the other hand, we can improve the Soundness of the model without introducing too much computational burden. This is of great value for neural networks in scenarios where highly accurate and reliable predictions are required.

1.3 Experimental Approach

Combining the background and motivation, this thesis culminates in designing a pool-based human-in-the-loop active learning text classification system. Unlike traditional text classification models, we use a small amount of annotated data to train and use cases filtered with the trained model combined with different acquisition functions. The extracted samples are then observed and analysed for key features using a human annotator in a loop that gives a label and a uniform interpretation of the extracted samples. These new use cases will be added to the original data, and the above loop is repeated until the performance reaches the full data performance.

To observe the performance under different query strategies, the implementation will use Least Confidence based uncertainty sampling, sentiment diversity sampling and random sampling as a benchmark. The specific acquisition function will be described in detail in Section II. In addition, in each loop, the spliced interpreted text will be pre-trained using ExpBERT to generate a deep semantic representation of the text, and the pre-trained result will be used as input to the neural network text classifier.

Finally, a structural change will be made to the traditional neural network model, using dropout in such a way that the model is more robust to small changes in the input, which improves the generalisation performance of the model. In addition, dropout can also alleviate the problem of overconfidence in the model by introducing noise and randomness, making the model's predictions less certain. Although dropout does not directly quantify uncertainty, the randomness and noise it provides can increase the robustness of the model and mitigate the problem of model overconfidence to some extent. Since the computational cost of solving for entropy in the original Bayesian Active Learning by Disagreement (BALD) algorithm was too high, the BALD algorithm was upgraded to utilise MC dropout while using Least Confidence to select the model with the most uniform predictive probability (i.e., no categories with exceptionally high predictive probability) of the Sample.

1.3.1 Objectives

Based on the experimental descriptions above, our objectives are summarised as follows.

-Develop a pool-based human-in-the-loop framework that incorporates pre-training as well as text classification models.

-Construct an annotator simulation flow and design loop abort criteria (e.g., rerun after reaching model performance under full data).

-Generate a baseline (model performance under full data without active learning or random sampling without annotation) to investigate whether active learning improves the performance of text classification models. Compare the impact of sampling with different query algorithms.

-Explored the impact of quality and quantity of explanations on this system. For example, we provided noisy interpreters or introduced randomness instead of exact interpretations.

1.3.2 Challenges

The main challenges of this project are whether there is a significant difference between different query strategies and how to improve the initialisation framework of the original ExpBERT text multiclassifier as well as the training framework to achieve a reduction in the amount of computation.

-It is worth considering that there are multiple query strategies for active learning text classification systems, and the application of different algorithms in the query strategies dramatically affects the performance of the active learning framework. The choice of query strategy needs to be considered in terms of both time complexity and representativeness.

-Secondly, the time consumption of neural networks with dropout mechanisms in the execution of training algorithms and active learning using MC dropout may cause an increase in computation, so how to reduce the amount of computation in each cycle to improve the response efficiency time is a challenge.

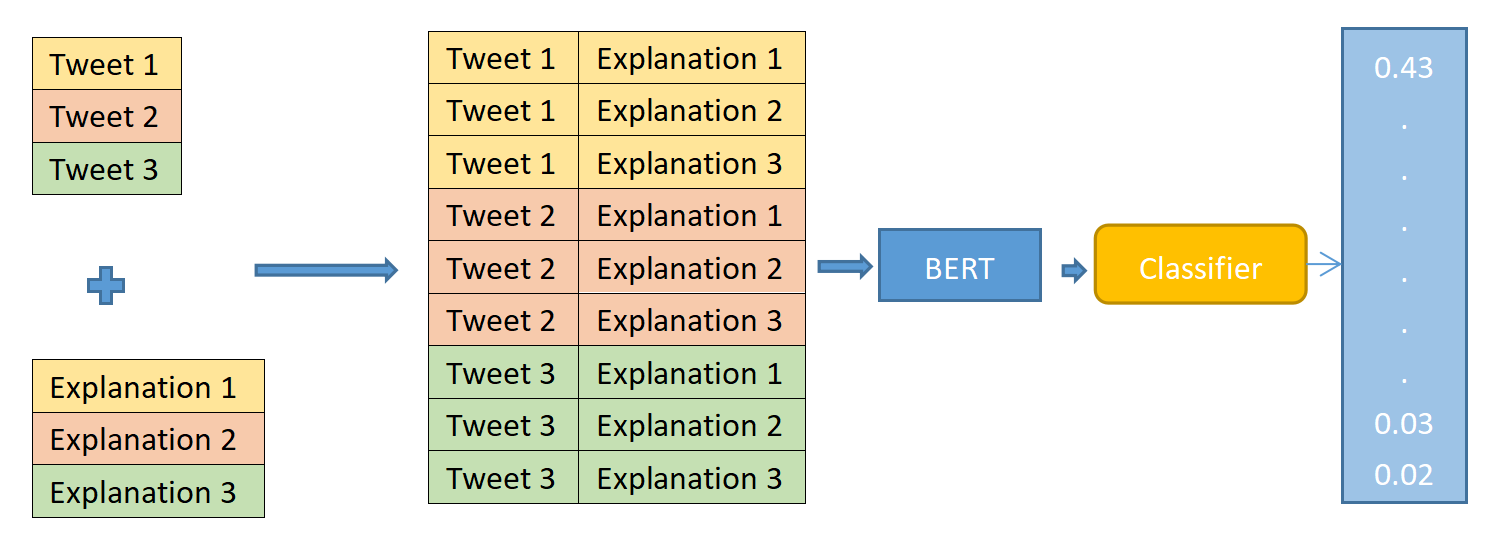
-Finally, for the framework design, how to appropriately set the proportion of sampling, the number of iterations and the number of explanations provided per iteration need to be investigated to achieve satisfactory performance.

2 Background.

This chapter describes the essential techniques to implement active learning to improve the performance of ExpBERT-based text classifiers, starting with the pre-training model used for the experiments (2.1 section), which will be applied to the dataset with training. The human-in-the-loop (2.2 section), combined with a query strategy based on active learning and its application (2.3 section), is the critical implementation part of this experiment, through which performance improvement can be accomplished. Section 2.5 describes the multi-model integration approach and the technique of simulating the human-in-the-loop using the Openai model in section 2.6. Finally, the feasibility of applying Bayesian theory to active learning is explored.

2.1 Representation Engineering with Natural Language Explanations (ExpBERT)

The ExpBERT model proposes a method to enhance language models' interpretability and knowledge fusion capabilities. It combines fixed interpretations provided by humans with tweets and learns from these interpretations to improve the model's performance [2]. Figure 1 visually shows how samples with explanations can be combined with a BERT model. Interpretations play a key role here; the quality of the interpretations has a much greater impact on the performance of ExpBERT than the number of interpretations, and by using high-quality interpretations the learning of the model can be guided.



First, each tweet is wholly connected to a set of pre-written explanations, generating a 3×3 set. These interpretations are independent of the particular tweet, and each tweet is connected to the same number of explanations. The preprocessed text and explanations are fed into BERT, which is used to generate features that "explain" each explanation [new 1]. A classifier can be trained to classify the input and explained representations using a Multilayer Perceptron (MLP) model, where the inputs are a representation of the input w(x) and a representation of the explanation z(x).

f(x) = MLP (w(x), z(x))

The fine-tuned BERT model on the MultiNLI natural language inference dataset will generate a feature vector for each input sample representing the entire input of length 786. The feature vectors of tweets and explanations are then concatenated to form a model of size 768 \* E, where E is the number of explanations, and used as input data for classifier model training and prediction. This model will be used in this thesis as a base pre-training model that will be integrated and initialised into the Human in the Loop system, as the model can handle instance vectors with explanations, and the optimised embedded representations can be used as inputs to the classification model to improve the performance of the model.

2.2 Human in the loop

Traditional natural language processing processes are not designed to take full advantage of human feedback. In contrast, human-in-the-loop, a key component of interactive systems, can be used to reveal model flaws that may not be apparent until real-world testing by simulating a human's role in the loop [11]. Godbole et al. (2004) [12] used support vector machines (SVMs) for active learning to extend a text classifier. They cleverly incorporated human inputs in feature engineering, term selection, document labelling, etc., into the model to make statistically sound judgements. This novel way of interaction between humans and machine learning algorithms is called human-in-the-loop (HITL) machine learning [13]. In this approach, depending on the needs and scenarios, different types of collaboration between humans and machines can be identified for different HITL machine-learning scenarios (Mosqueira-Rey et al., 2023) [14].

Active Learning (AL) (Settles 2009) [15]: the key to active learning is that the system maintains control over the process of model learning, and although humans act as a mediator to participate in the annotation of unlabelled data, humans are unable to select unlabelled data based on preference.AL will be applied as an optimisation framework in this thesis, and the application of active learning will be described in detail in Section 2.2.

Interactive Machine Learning (IML) (Amershi et al. 2014) [16] is an approach where humans maintain close interaction with the system. In this context, there are some key differences between Active Learning (AL) and Interactive Machine Learning (IML). Dudley and Kristensson (2018) [17] point out that while both AL and IML focus on selecting new data points to be labelled by the user, they are driven in different ways. Specifically, in AL, the selection is driven by the model itself, whereas in IML, the selection is driven by the user. Since IML is based on AL, they have some common disadvantages. However, IML also has several unique issues, particularly in relation to the complexity of human-computer interaction (HCI) technology. These endemic issues require more dedicated research to address (Michael et al. (2020)) [18].

Machine Teaching (MT) (Ramos et al. 2020) [19] is a method of training machine learning models through the guidance of a human teacher. Specifically, human teachers define the knowledge they intend to transfer to the model, emphasising the active involvement and guidance of the teacher in the learning process. Devidze et al. (2020) [20] point out that MT relies more on the teacher's expertise than on active learning while being less flexible in sample selection and handling complex tasks. Therefore, selecting appropriate learning methods according to specific needs is particularly important.

2.3 Active Learning

As the most popular learning scheme in HITL, active learning systems try to overcome the labelling bottleneck by asking questions to unlabelled instances and having them labelled by experts (e.g., manual annotators) [21]. In short, active learning works to find the most informative unlabelled raw data and gives it to an annotator for labelling. This process resembles the actual scenario of extracting information from source data. The annotator labels this source data and adds the labelled instances to the model training process. In this way, active learning can overcome the labelling bottleneck problem by using less annotated data to achieve performance under the full amount of data. The active learning approach is particularly applicable in this project where the number of instances is large, noisy, and the full-volume training task is heavy with low accuracy. Therefore active learning is suitable for this project to focus on those instances that may increase the accuracy rate and reduce the irrelevant instances to improve the efficiency of the emergency response system and the accuracy rate.

2.3.1 The AL process and scenarios

The active learning process is illustrated in Figure 1, which shows the process of pool-based active learning. Firstly, the machine learning model is initially trained using the labelled set of instances, L. Then, the model performs feature extraction on the unlabelled sample set and selects representative unlabelled samples based on the query strategy provided to the human annotator. The human annotator annotates the selected models, and the annotated samples are removed from the unlabelled sample set U and added to the labelled sample set L. By iterating the above steps, the number of labelled samples gradually increases and the model performance improves. Finally, once the termination criterion is reached (e.g., model performance at full data volume is achieved), the active learning process is aborted.

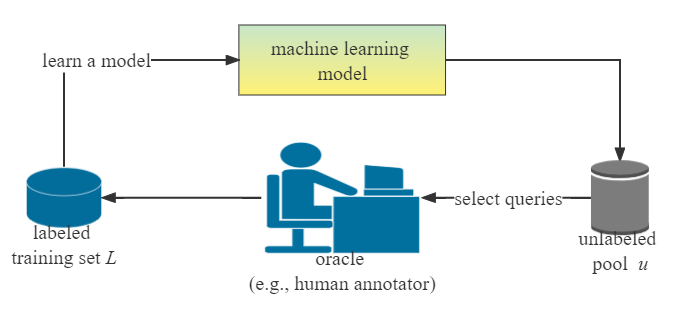


Figure 2: Pool-based active learning process

According to Settles' (2009) [15] classification, there are three main scenario settings for active learning: membership query synthesis [22, 23], stream-based selective sampling [24], and pool-based active learning [ 25].

Membership query synthesis: the query strategy for this scenario is generated by the model. That is, the machine learning model can request the labelling of unlabelled instances which are not sampled in the underlying natural distribution [15]. For more absolute problem domains, such as in applying regression prediction to absolute coordinates, effective query synthesis is often effective. This is because it can resolve data distributions for simple d and construct reasonable data for human labelling. However, Baum and Lang (1992) [26] show that for complex tasks such as natural language processing, the model may produce strings of text that are not easily understandable, resulting in humans being unable to make judgements about the confusing text. In the context of deep active learning, the scenario of member query synthesis can be addressed by generative adversarial networks (GANs) used for data augmentation, as GANs are capable of generating instances with a high degree of plausibility [27].

Stream-Based Selective Sampling: the difference in this scenario setting compared to Membership query synthesis is that this approach can be independent of the input distribution. Stream-based sampling draws one unlabelled instance at a time from the actual distribution, and the model decides whether to request the labels of the instances using an "informativeness metric" or a "query policy", which is equivalent to a biased random sampling [28].

Pool-based AL: As a commonly used setting for active learning, pool-based AL differs from stream-based sampling in that pool-based AL adopts a greedy mechanism for comparing the entire dataset before selecting the best query, but the latter receives the data individually for evaluation. However, when the dataset is extensive, it may be computationally time-consuming due to the need to evaluate the entire data. Apart from that, this method is suitable for scenarios where the cost of manual annotation is significant by selecting the most valuable data for annotation, thus maximising the value of the annotation input.

2.4 Acquisition Functions

In this section, various acquisition functions in active learning are considered, and the ExpBERT-based text multiclassification model will be used in the implementation part of the thesis to achieve performance improvement by combining different acquisition functions.

Random sampling

Random sampling selects instances randomly, neither based on predictions nor on data as well as models, and is often used as a benchmark for the task. This is because, it may ignore the latent ground information, which reduces the learning efficiency. In this case, random sampling will be used as a bottom line against the more complex strategies mentioned below, especially when the pool of markers is too large [29].

Uncertainty sampling

Uncertainty sampling is a special active learning strategy that improves model determination accuracy by querying for instances that are the most difficult to determine their classification and labelling them. In binary classification problems with probabilistic models, the a posteriori positive probability of such instances is closest to 0.5 [30]. However, for more complex multi-label classification problems, an entropy-based approach will be used. The more uniform the probability distribution, the higher the entropy, the higher the uncertainty of the random variable and the more informative it is. When the probabilities are concentrated in a few data points, it indicates lower uncertainty and less information.



 represents the probability distribution in classification .

In the field of text classification, an alternative is commonly used for the measurement of uncertainty, namely least confidence [31]:



denotes the most probable class label. For binary text classification, this method is equivalent to entropy-based algorithm effectiveness. Consider those samples that have the highest probability of model prediction but lower confidence. This is done by selecting the samples with the smallest maximum probability for labelling.



represent the most confident.，is the next best confidence.。

**Semantic-based Diversity Sampling**

Peng, Hao, et al. (2023) [32] proposed a semantic-based diversity sampling approach that can be applied to text classification. The difference with the process of measuring confidence using uncertainty sampling is that the semantic-based diversity sampling approach uses Euclidean distance to eliminate redundancy in text samples semantically. This ensures that a richer, less repetitive sample is provided to the model (learner) in the subsequent process. This abstraction approach uses the greedy k-centre algorithm of Sener and Saveravarese (2017) [33] for clustering operations. The dataset  contains  unlabelled texts and divides  into  batches, with each sample set containing  instances. It is the result of encoding the dataset. First, select vectors from  to initialise the clusters . Here examples will be considered as cluster centres. Then the k-centered algorithm searches  from , which is a set that includes members that are not in . It is the furthest from the centre of all clusters. The algorithm chosen is formulated as follows:



Where:



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

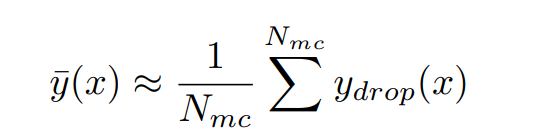
Bayesian active learning by Disagreement:

BALD (Bayesian Active Learning by Disagreement) is an active learning strategy in text classification that combines Bayesian inference and uncertainty measures to select information-rich samples. Uncertainty is measured in BALD by estimating the category probability distribution for each sample and calculating the inconsistency between predictions (Houlsby, N (2011)) [35]. The learner (model) maximises the uncertainty of the model parameters by input x. H[y|x, D] represents the uncertainty of the target variable. The second term of the equation represents the average uncertainty measured by the calculation of the expected value of the uncertainty (entropy) of H[y|x, θ] conditional on the parameter θ obeying the posterior probability distribution p(θ|D) of the training dataset D.



To better compute the acquisition function, Gal et al. in 2017 proposed a specific sampling function using the Monte Carlo (MC) dropout method [36]. MC dropout is a regularization method during training that simulates the Bayesian network's posterior distribution by randomly deactivating some neurons during the training process [37]. This allows for the selection of samples with the highest uncertainty by making multiple predictions for each sample and calculating the inconsistency between the model's predictions.

Essentially, MC dropout is a regularization method that probabilistically deactivates certain neurons to prevent overfitting. During the inference process, by inputting to the neural network a finite number of times (T times), dropout is used to produce varied results in the prediction phase [38]. Consequently, after quantifying the uncertainty of these varied predictions, they can be used to estimate the model's uncertainty for unlabelled samples. This is then applied in the Bayesian Active Learning by Disagreement (BALD) sampling strategy in active learning to select unlabelled data with the greatest uncertainty. The inference results of the MC dropout are obtained in the following manner:



Wherein, ydrop represents the varied outputs from the dropout network, x is the input to the network, and Nmc is the number of sampling times required to obtain the distribution. Ultimately, BALD selects samples with the greatest information gain for labeling by comparing the BALD values of each sample and then adds them to the labeled training set. In addition, BALD can substitute the entropy calculation originally used with the previously mentioned Least Confidence, to select samples where the model's prediction probability is the most uniform (i.e., there isn't any category with particularly high prediction probability).

Small-Text Library

The text classifier in this project tends to focus on one model, potentially overlooking the application of other viable models. However, the time cost of switching models and active learning strategies, along with code redundancy, will significantly impact the progress of experiments. The Small-Text Library integrates popular libraries that can be applied in the Python environment, such as scikit-learn, transformers, and PyTorch [39]. The pool-based active learning framework for text classification connects interfaces of query strategies, classifiers, and stopping policies. It not only provides an advanced active learning framework for text classification tasks but also offers a range of classifier and query strategy components. This allows for a mix-and-match approach to active learning experiments and applications, making active learning easy to implement in the Python ecosystem. Compared to the widely used ModAL library [40], Small-Text offers more flexible customization services, with the former focusing more on model integration and the selection of query strategies. However, this all-encompassing approach may not necessarily improve the performance of projects based on ExpBERT, as performance disparities can vary depending on the situation.

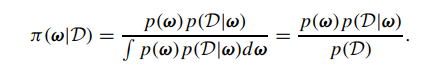
ChatGPT Annotator Simulation Explanation Generation

ChatGPT (Generative Pre-trained Transformer) [41] is a pre-trained language model developed by the OpenAI team in 2018. The core algorithm is the Transformer, a deep neural network structure based on a self-attention mechanism, boasting powerful sequence modeling and representational learning capabilities. Through pre-training and fine-tuning, this model can analyze and generate natural language text, aiding in various scenarios such as automated responses, intelligent customer support, language translation, etc. Therefore, as a time-saving and efficient text explanation system, this project will utilize 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 help enhance classification performance. In this configuration, active learning leverages ChatGPT's potent generative model [42] to produce human-readable explanations based on input text. A common approach is to use ChatGPT for keyword extraction on sampled texts of specific categories. These extracted keywords are then combined to form explanations. One or more of these explanations are selected and added to the original set of explanations, which, combined with the text, is passed to the ExpBERT model. By analyzing sampled texts in this manner, it addresses the deficiencies in human memory and word frequency calculation capabilities, identifying crucial feature words and phrases that best aid the model in understanding the intrinsic data structure.

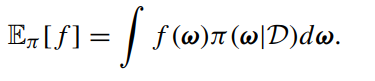
Active Learning for Bayesian Neural Networks

In contemporary machine learning and deep learning applications, Bayesian neural networks have emerged as a critical component due to their flexible inference methods. By introducing uncertainty in weights and predictions, Bayesian neural networks provide a rich and adaptive modeling framework. This not only bolsters the reliability of the model but also enhances its generalization capability, enabling it to adapt to noise and outliers. What differentiates it from conventional neural networks is that the weights within a Bayesian neural network aren't static. The idea behind the model is to amalgamate the advantages of neural networks with stochastic modeling, offering probabilistic assurances for predictions [43]. Here, the weights are inferred based on our observations, posing an inverse probability problem addressed through Bayes' theorem [44]. The weights, represented by w, signify latent variables that aren't directly observable from the true distribution. Bayesian methods allow us to obtain the posterior distribution of model parameters given the observed data, p(w∣D).

Concurrently, the likelihood function p(D∣w) in multi-classification problems combines the predictions of the neural network (transformed through Log Softmax) with the actual observed class labels to quantify their consistency. After determining the likelihood term and the prior distribution p(w), the posterior distribution can be computed based on Bayesian theory.



After obtaining the posterior distribution, predictions will be made using marginalization:



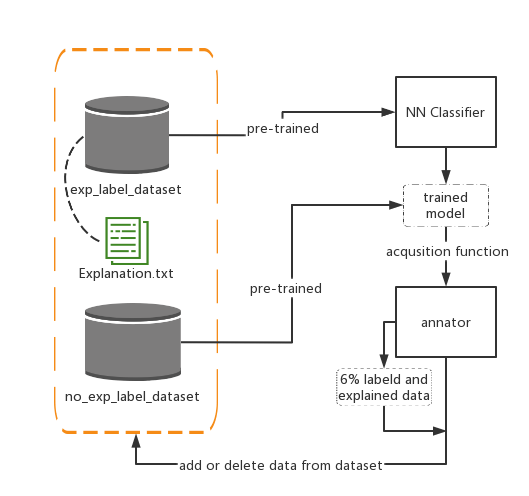
However, integrating over model parameters typically requires computationally expensive processes, such as MCMC (Markov Chain Monte Carlo) sampling or variational inference. This can pose a problem when dealing with large-scale datasets, especially in the context of active learning, where multiple iterations and model updates are often needed. Consequently, the Bayesian Neural Network (BNN) approach might extend the training time of emergency systems [45].

3 Design

This chapter on design will provide an overview of the active learning framework in Section 3.1, followed by a step-by-step breakdown and design based on the order of execution for active learning tasks. Starting with pre-training on the data in Section 3.2, we then move to the design of a classifier model that obtains embedded representations in Section 3.3. Section 3.4 describes the design of active learning sampling strategies from various perspectives using the trained classifier model and analyzes the implementation methods and functions of different sampling strategies. Finally, the design of the annotator simulation feature is covered in Section 3.5.

3.1 Overall Process

The entire experimental framework is built based on the pool-based active learning technique mentioned in Chapter 2. It involves sampling from a fixed pool of unlabeled samples and using a trained model to evaluate these samples. What sets this apart from traditional active learning is our revised requirement for datasets, introducing two datasets: one with labels and explanations (exp\_label\_dataset) and the other without labels and explanations (no\_exp\_label\_dataset). The exp\_label\_dataset includes datasets for testing, validation, and training. To make full use of the ExpBERT pre-trained model, each round of annotation provides not only labels but also adds explanations to the default explanation set, Explanation.txt. These three main collections will be fully involved in the three main modules, iterating continuously. These three modules are training, annotation, and update:



Training: During the initial training phase, 20% of the dataset with labels and explanations is used. Each sample in the exp\_label\_dataset is linked to the default nine explanations (label descriptions) for the first round of training. After the first round of training, the trained model will evaluate the no\_exp\_label\_dataset and calculate the "information content" of each sample from perspectives like uncertainty or diversity. Subsequently, the data with the highest "information content" is selected for annotation. The overall design and process of the active learning architecture are illustrated in the following diagram.

Annotation: In the annotation process illustrated in the diagram, the annotator will summarize key sentences from the sampled text as explanations and provide classification labels for each text. The added number of explanations will exceed the default number of explanations, updating the Explanation.txt set. After quantification using the acquisition function, we will fully annotate the sampled data and add it to the exp\_label\_dataset, forming the dataset needed for the next round of iteration. These datasets will be preprocessed by the ExpBERT model and then fed into the neural network text classifier.

Dataset Update: Subsequently, the annotated 6% of data will be removed from the no\_exp\_label\_dataset. With each iteration, the volume of data in the exp\_label\_dataset will increase by six percent, and one or more explanations will be added. This process is repeated until the evaluation score of the validation set approaches or exceeds the evaluation score obtained using the full volume of data.

3.2 Pre-training

Following the framework design mentioned above, all datasets require a pre-training process before training the text classifier. As mentioned in Chapter 2 regarding ExpBERT, pre-trained models are typically trained on large volumes of text data, capturing deep bidirectional language representations, which often results in higher performance on specific tasks. In many text classification scenarios, using pre-trained models as a base and fine-tuning can significantly reduce the training time for text classification models. Text is usually input into the model to obtain its embedded representation (embeddings). This embedded representation can then be used as input for downstream text classifiers.

The amount of explanations corresponding to the annotated dataset will increase by new\_exp during active learning iterations. Thus, every time the pre-training step is run during an iteration, the updated training, testing, and validation set texts need to be reconnected with the updated explanations to serve as input for the pre-training model. As a result, each pre-training model will process an additional amount of data equivalent to num\_tweets \* new\_exp. This is a time-consuming part of the experiment in its implementation.

For the unannotated dataset, to be closer to real-world situations, only the text data needs to be processed during pre-training. However, the trained neural network expects to receive inputs of the same dimension as during training. Therefore, for the embedded representation of unlabeled data, the method F.pad(embedding, (0, pad\_amount)) is used to ensure that the embedding tensors of the two datasets match.

Given that the embedded representations obtained from the original model are of a very large scale, resulting in prolonged computation time, the training process employs Natural Language Inference (NLI) to fine-tune BERT. Through NLI, three vectors of different sizes are output, making the embedding size of a sample only 768 plus the number of explanations and three times the text size. For the non-fine-tuned model, the extended dataset, which is fully connected between tweets and explanations, is passed directly through BERT. Moreover, to improve computational efficiency with each iteration, pre-training adopts a pool-based approach to speed up calculations. The 'Pool' class from Python's 'multiprocessing' library is utilized to process data batches in parallel. On machines with multiple cores, multiple batches can be processed simultaneously, allowing for flexible control of processes and increased processing efficiency.

3.3 Classifier Model Design

Based on the framework diagram mentioned above, we use neural networks as our classifier model. The evaluation phase will compare the performance of traditional neural network classifiers and neural network classifiers with dropout when integrated with active learning techniques. This experiment will make structural adjustments to the traditional neural network. The final model structure will be built using PyTorch [reference to be inserted later].

3.3.1 Before dropout:

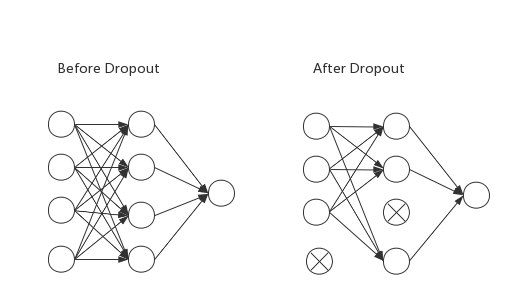
In terms of model structure, the neural network in this experiment uses a single hidden layer with 100 neurons. The number of output features, or 'output\_size', is 9 (representing the number of labels/categories). The forward propagation design can be represented by the following formula, where wi and bi are the weights and biases of the linear layer i, respectively. After passing through an activation function, we get the model's output y. This experiment will use the ReLU non-linear activation function, inducing sparsity in the network and thus mitigating overfitting issues.



Regarding the training approach, the regular neural network employs standard loss computation methods, such as cross-entropy [reference]. Moreover, the steps of forward propagation, loss computation, backward propagation, and optimization are executed separately, i.e., using the standard methods provided by PyTorch.

3.3.2 After dropout:

This paper improved the model structure by employing a neural network model with dropout as the text classifier. Hinton mentioned that Dropout can prevent neurons from becoming overly dependent on other specific neurons within the network, thereby enhancing their independence and bolstering the model's generalization capabilities [reference]. Applying dropout to a neural network is equivalent to using a "sparse" network composed of units that survive the dropout process. A neural network with n units can have up to 2^n possible subsets of active neural networks. For every presented training case, a new sparse network is sampled and trained [reference for dropout]. Dropout provides an efficient approximation to various neural network structures by randomly discarding units. The process of dropping units is illustrated in the following diagram, achieved by removing certain neurons from the network and severing connections.



Dropout Neural Net Model. Left: A standard neural net with 1 hidden layer. Right:

An example of a thinned net produced by applying dropout to the network on the left.

In this paper, the 'dropout\_prob' parameter is set to 0.2 during neural network initialization, meaning there's a 20% probability that a neuron will be temporarily removed from the network during each forward propagation. With this, the original forward propagation formula of the neural network will be altered:

h = W1\*x + b1，a = activation\_fn(h)，r = Bernoulli(p), a\_dropout = r \* a y = W2\*a\_dropout + b2

Here, Bernoulli(p) represents a random variable following the Bernoulli distribution, and "\*" denotes element-wise multiplication, meaning some elements of the vector a are randomly set to 0.

3.4 Sampling Strategies

Within our neural network classifier, we employed four distinct sampling strategies to boost 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) using MC dropout. These strategies each harnessed and exploited the unique attributes of the data in different ways, collaboratively advancing the efficiency and accuracy of model learning within the active learning framework. By exploring sampling from different perspectives, we aimed to uncover the best approach to enhance model performance.

3.4.1 Random Sampling:

During the sampling process, the acquisition function evaluates nearly 70% of the unlabeled dataset. After each evaluation, 6% of the unlabeled dataset is presented to the annotator for labeling. Random sampling ensures unbiased sample selection. By setting a random seed, samples are drawn directly from the unlabeled dataset without considering the initial trained model. The model's performance under this influence serves as our baseline.

3.4.2 Uncertainty Sampling Based on "Least Confidence":

This paper employs the Least Confidence method to compute uncertainty. As described in Chapter 2, for multi-label text classification problems, Least Confidence is often used as an alternative to calculating entropy. In this experiment, by setting the model to evaluation mode, the maximum probability value is sought within the probability distribution 'prob\_dist' produced by the model, serving as the uncertainty score. The number of labels, denoted as 'num\_labels', is calculated and the most confident prediction is normalized. The normalization formula is as follows, where 'normalized\_lc' yields the normalized least confidence value:

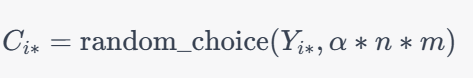
normalized\_lc= (1 - max(prob\_dist)) \* (num\_labels / (num\_labels - 1))

By sorting in ascending order based on the least confidence values, the top 6% of the samples are selected as the output for this uncertainty-based method.

3.4.3 Semantics-based Diversity Sampling:

In this study, a sampling method based on semantic diversity was employed. As introduced in Chapter 2, this approach aims to identify rich and less redundant samples during the learning process, offering a perspective distinct from uncertainty sampling. For the dataset used in the experiment, there is a significant imbalance in the number of samples for each category. Some categories account for only 2% of the entire dataset. Such data imbalance might lead to inadequate training for certain categories, resulting in greater uncertainty during the evaluation process. If sampling is solely based on uncertainty, there's a potential risk of oversampling from a specific category, which would imbalance the distribution of sampled categories and subsequently affect the model's learning performance. To utilize samples from all categories more equitably during the active learning process, the experiment adopted a semantic diversity-based sampling strategy.

For this method, it's required to load the unlabeled dataset using post-training embedding vectors. By setting the trained neural network model to evaluation mode, probabilities corresponding to input embedding vectors are calculated. Referring to Chapter 2, the embedding vector of each text is denoted as Yi∗, where i represents the index of the text. Suppose the dataset is divided into n batches, each with m instances. Initially, a subset of embedding vectors is randomly chosen as the center of the initial clustering Ci∗. In this experiment, Ci∗is represented as follows, setting the proportion of the initial clustering center to the entire data, denoted by α, to 0.2:



Then, the greedy k-centers algorithm is employed to search for embedding vectors in Yi∗that are not present in Ci\*. The aim is to find the embedding vector farthest from the existing clustering center and update the set of clustering centers. This process can be represented as:



Dist is used to compute the semantic distance between the embedding vectors and the clustering centers. In the end, we return the index of each clustering center in Ci\* within YI\*, denoted as Oi. This allows us to extract corresponding samples from the unlabeled dataset.

3.4.4 MC Dropout-Least Confidence BALD

In this study, we adopted a sampling strategy based on Monte Carlo (MC) dropout to identify samples with significant model uncertainty, enabling more effective model training. This strategy is rooted in the 2016 research by Gal and Ghahramani [citation], in which they proposed that dropout could be used as a method of approximate Bayesian inference while training neural networks.

Similar to other active learning strategies, the embedding representation E of the unlabeled dataset is loaded. What differs is that the text classification model involves a neural network with a dropout mechanism. Moreover, the model should be in training mode, not in evaluation mode. This is because MC dropout requires dropout to be activated during training. After J predictions, a list of prediction probabilities pi is produced. This can be expressed as:



Wherein, pi represents the list of prediction probabilities for the embedding vector Ei.

Unlike the traditional BALD strategy, this experiment uses the negative maximum of the predicted probabilities to measure uncertainty, rather than the conventional BALD calculation method.



Subsequently, the indices of all samples along with their corresponding BALD scores are stored in a list, which is then sorted in descending order based on the BALD scores. This allows for the identification of 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 (Bayesian Active Learning by Disagreement) algorithm that integrates Monte Carlo Dropout and the Least Confidence strategy. When faced with imbalanced datasets, the model often tends to predict the categories that dominate in quantity. By employing the least confidence strategy, which selects samples with the greatest prediction uncertainty, we can prioritize those underrepresented categories that the model tends to underestimate during data collection, thereby enhancing the model's performance in handling these categories. Compared to only applying the least confidence uncertainty sampling, this approach can capture a more comprehensive view of the model's uncertainty.

3.5 Annotation Process

After the active learning strategy-guided sampling process, we obtain a set of unlabeled texts with high informational content. For these texts, annotators need to categorize them based on their content and decide which guiding explanations to add during the current iteration.

3.5.1 Label Annotation:

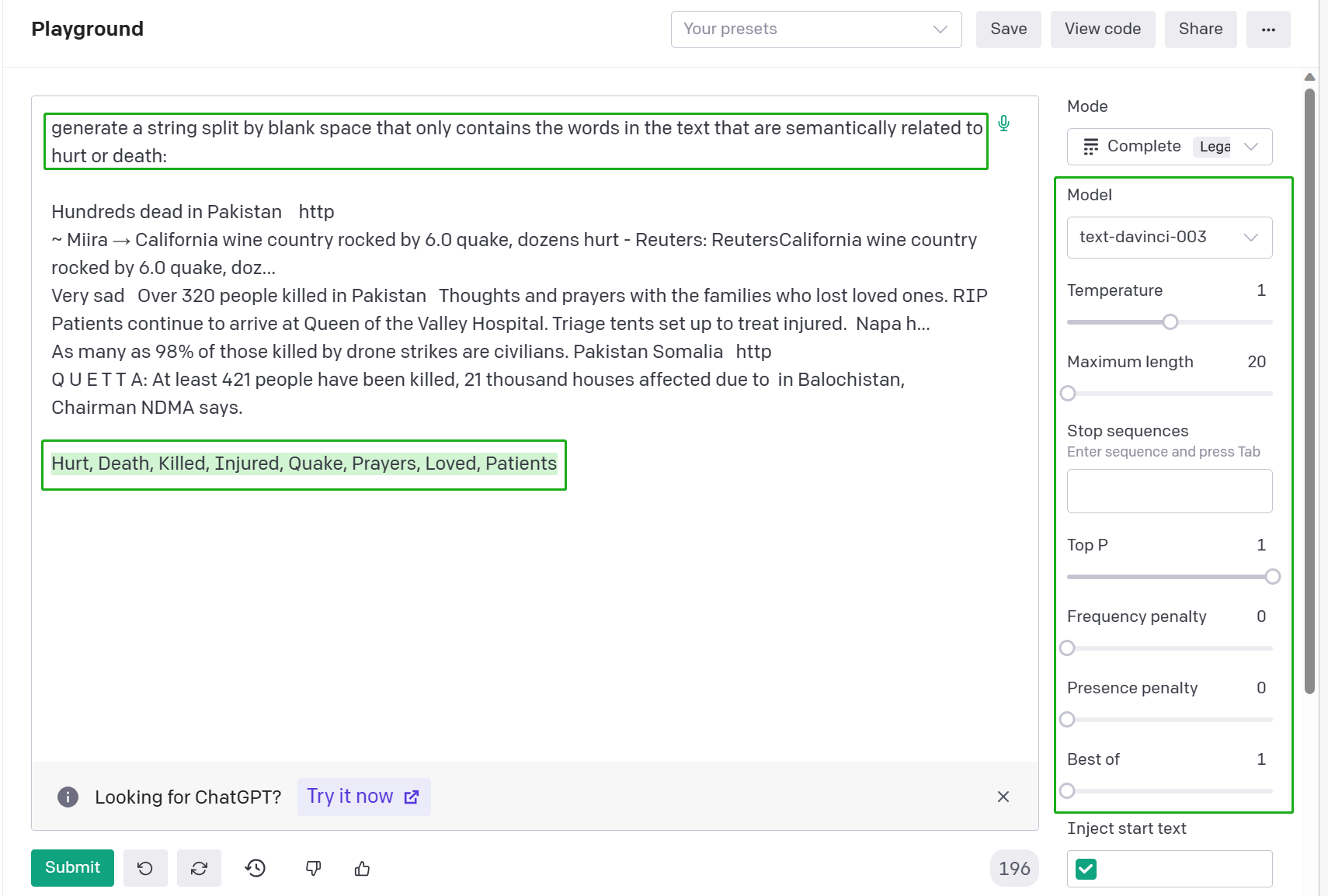
The focus of this experiment is to explore whether adding an explanatory annotation feature during the active learning process can enhance the performance of the text classifier. To simulate the ideal scenario, we assume that in each iteration, annotators have a complete understanding of the text classification, thus the labels provided will accurately correspond to the source data. During the evaluation phase, to neutralize the potential influence of added labels and text volume on model performance, we will compare two scenarios: one that only adds labels through active learning and another that adds both labels and explanations.

3.5.2 Explanation Annotation:

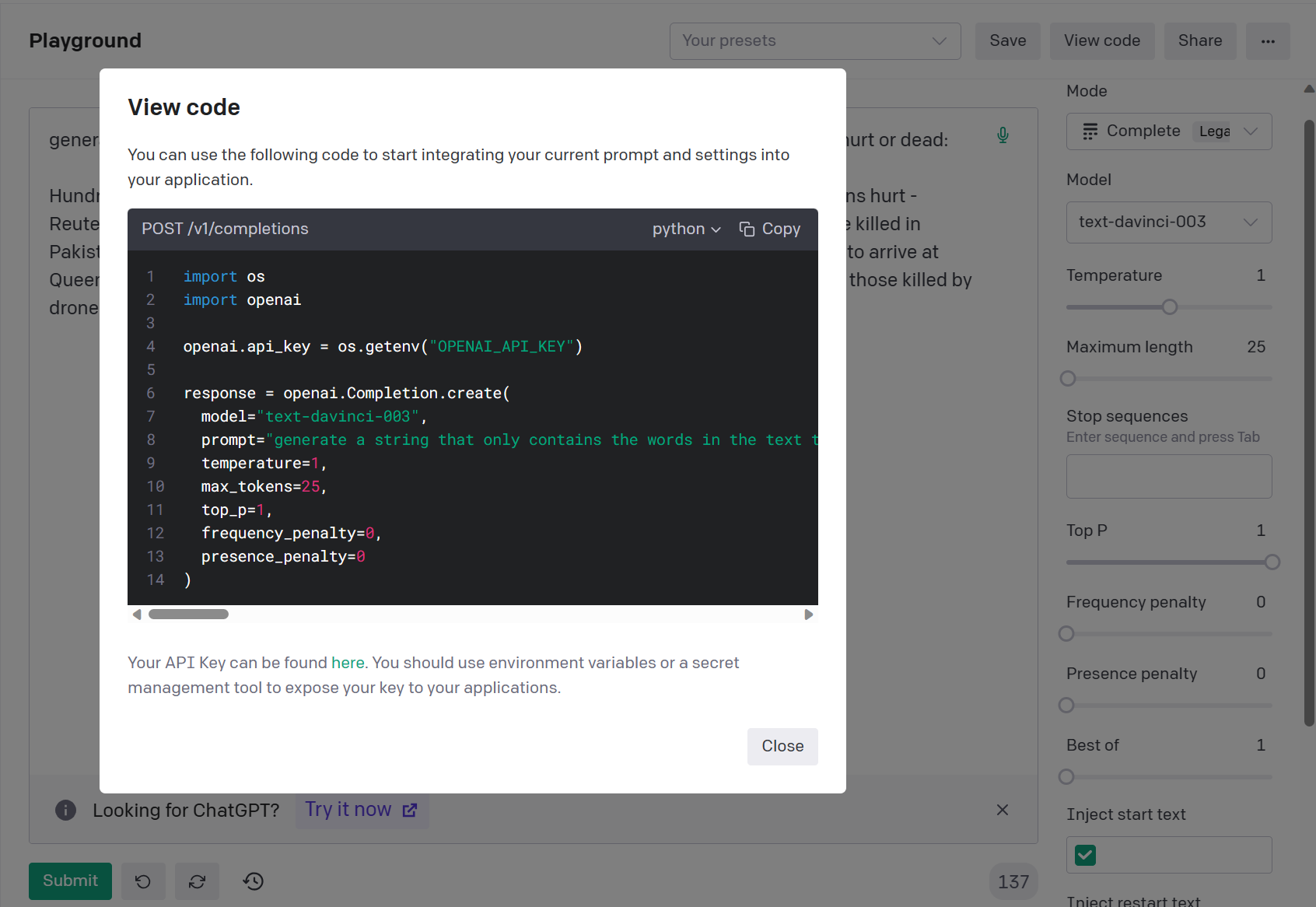
Three distinct explanation annotation strategies are designed for this experiment. The provided explanations will be stored in the 'Explanation.txt' as displayed in the structure diagram of this chapter, available for pre-training text concatenation tasks. System operators can choose among these three strategies based on manpower allocation or cost-control considerations, better supporting the model's learning process:

The first strategy involves pre-examining the texts in each iteration and simulating potential added explanations, just as an annotator would. Preset explanations are stored in the 'annotator.txt' file. When the annotation phase is executed, explanations are automatically extracted from this file. While this method can reduce manpower consumption, it lacks adaptability and is only suited for the datasets related to this experiment.

The second strategy allows users to directly provide their inputs as explanations. This strategy requires users to analyze the selected text's characteristics and provide their explanations. In each iteration, annotators will focus on ten prominent texts of a particular category for swift analysis. This method integrates human judgment and is also applicable to other emergency-related data, offering a degree of flexibility.



The last mode employs the OpenAI model to automatically provide explanations, aiming to reduce manpower expenditure. In this experiment, the "text-davinci-003" engine was chosen for text analysis [reference]. As depicted in the aforementioned diagram, developers can simulate the prompt they wish to input to the model in the Playground through the OpenAI API developer platform. On the right-hand navigation bar, users can select the model, temperature, and maximum response length. The best-performing hyperparameters are chosen based on the reasonableness of the responses. Testing revealed that the "text-davinci-003" model greatly outperforms the "text-ada-001" model in terms of response reasonableness and accuracy. After debugging, it was found that setting the "temperature" to 1, Top\_P to 1, and the maximum response length to 25 yielded the most reasonable outputs from the model. Playground provides the configuration code in the Python environment, and the final model configuration is illustrated in the subsequent diagram. In order to successfully connect to the OpenAI interface, it's necessary to apply for an OpenAI key and load this key before generating a response.



Using this model, keyword analysis can be performed on five prominent texts from a particular category. The keywords that semantically align most closely with the category's label description are selected and concatenated into a single string. The final returned string, which best reflects the characteristics of the text and aids the model in learning, is added as an explanation. However, a drawback is that it's not conducive to generating multiple explanations at once or for active learning requirements of hundreds of tokens. While it reduces manpower costs, the generation of each token incurs a certain financial expense.

4. Implementation

4.1 Experimental Environment

This experiment uses Python 3.9 as the primary development language for natural language processing tasks. Highly suited for natural language processing tasks, Python boasts numerous data science and machine learning libraries. In this experiment, functionalities provided by libraries such as PyTorch, Pandas, and Numpy are leveraged for data analysis, preprocessing, and machine learning [reference]. Python's high compatibility allows for the use of PyCharm in this experiment to connect to remote High Performance Computing Systems (HPC) or to run using Google Colab in conjunction with Google Drive. The experiment employs PyTorch integrated with Python to construct dynamic neural networks. By combining the automatic gradient computation of PyTorch, the features of the Dynamic Computational Graph, and the capacity to accelerate calculations using CUDA, the neural network model maintains high flexibility and accuracy [reference].

4.2 Dataset

4.2.1 CrisisNLP

The CrisisNLP dataset is used in this experiment, which includes data id, text information, and descriptions of the category to which they belong. There are nearly 16,000 pieces of data in total. This dataset needs preprocessing, splitting, pre-training, and classification. CrisisNLP is an open-source resource specifically designed for research and development in natural language processing (NLP) related to humanitarian emergencies and crisis response. The dataset includes large-scale interactions from social media, covering various 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 and distribution of the labels are shown in the table below:

|  |  |  |
| --- | --- | --- |
| 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% |

Based on the table distribution, it can be observed that the dataset is highly imbalanced. Label 7 has the highest proportion and the information it provides is quite noisy, which accurately reflects the distribution of emergency information in real-life situations. This poses a challenge for the text classification task. Therefore, this dataset is extremely valuable to researchers as it records data and distribution corresponding to people's genuine reactions and behaviors in various crisis situations. Hence, researchers can develop more effective systems tailored for real data imbalance situations to address these crises, such as automatically detecting crisis-related social media posts or predicting the trend of a crisis through social media data analysis.

4.2.2 Data Preparation

Preprocessing:

Before inputting into the pre-trained model, preprocessing the tweets in the dataset is a crucial step. The aim is to minimize text noise and enhance the learning efficiency of the model. Firstly, normalize internet jargon by converting informal abbreviations or slang into their standard forms and splitting camel-case words. Next, use the emoji library to replace emojis in tweets with their corresponding text forms, helping the model better understand the meanings of these symbols. Lastly, clean or replace web links, usernames, and "#" tags in tweets since these elements typically don't aid much in the model's learning. Such preprocessing steps help reduce noise in tweets, thereby boosting the model's learning efficiency and performance.

Dataset Split:

For active learning tasks, the dataset needs to be split into unannotated and annotated datasets. At the same time, the annotated dataset will be further split into training, validation, and test sets. Active learning requires data to be added or removed in each round. However, to ensure that the model's performance on unseen data can be fairly tested in each iteration of active learning, the number in the test set should not change arbitrarily. Thus, the test set, unannotated dataset, and annotated dataset need to be split outside the active learning loop. The training and validation sets will expand with each iteration as the dataset grows, but their ratio remains unchanged. Since the data volume corresponding to Label 1 is only about 2% of the total data, to ensure that there's data corresponding to Label 1 in all datasets, the 'StratifiedShuffleSplit' is used to maintain the original class distribution in the data. Compared to random splitting, this can better evaluate the performance of the classifier.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Amount | | Proportion |
| annotated datasets | Training datasets | 2890 | 18% |
| Validation datasets | 331 | 2% |
| Test datasets | 1606 | 10% |
| Unannotated datasets | 11241 | | 70% |

Under the total data volume, the combined length of the training and validation sets of the annotated dataset will be 12,845, while the test set will consistently remain at a length of 1,606 throughout the entire data volume and throughout the active learning loop. To align with the principle of active learning, which seeks high performance with minimal data, the initial iteration of active learning will not utilize the entire annotated dataset. As indicated in the table above, the count is merely thirty percent of the entire dataset, standing at 4,827. Meanwhile, the dataset without annotations is set to a length of 11,241. With each loop, four percent of the dataset will be selected for annotation. In the subsequent evaluation chapter, it's discovered that after 9 iterations, the model's performance closely approximates that of the validation set under the full dataset. At this point, the length of the dataset without annotations remains at 6,741.

4.3 Performance Metrics

The selection of performance metrics can significantly influence the effectiveness of active learning projects. Proper evaluation criteria can measure and compare the impacts of various active learning strategies and model architectures on the emergency systems in human learning. This comparison helps in determining the choice of models, active learning strategies, and parameters. To evaluate the performance of active learning, data is first divided following the criteria of section 4.2.2. Afterward, training begins on the training set. After each round of pool-based active learning iteration, the trained model is assessed on the test and validation sets using the predefined performance metrics. After each iteration, a learning curve is plotted based on these metrics.

To comprehensively evaluate the model's performance, the experiment uses two performance metrics: accuracy and F1 score. Accuracy measures the proportion of instances correctly predicted by the model to the total number of instances and can be a somewhat biased metric. In contrast, the F1 score is the harmonic mean of Precision and Recall. Compared to accuracy, it eliminates bias and is more common in multi-class problems. The formula is as follows:

F1\_score=2/(1/Precision+1/Recall)

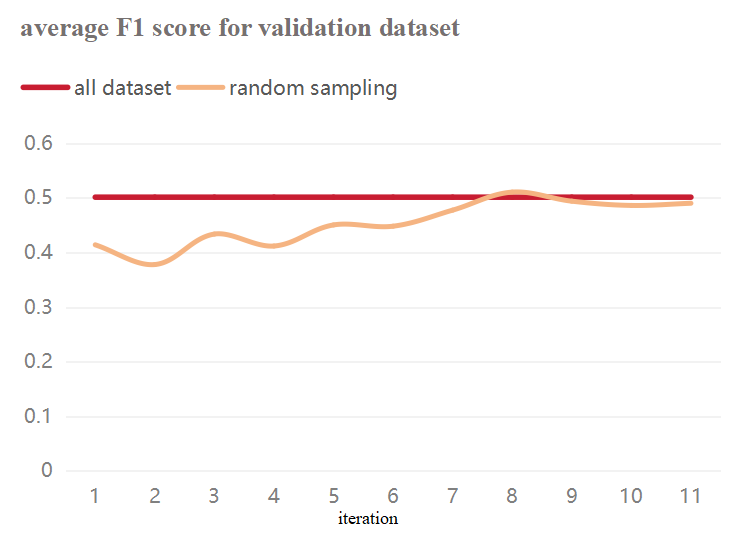
Where Precision is the proportion of positive instances among those predicted positive by the model, and Recall is the proportion of actual positive instances that the model predicts as positive. The F1 score, when dealing with imbalanced datasets like CrisisNLP, provides a more comprehensive performance measure.

In addition to these, the learning curve can also serve as a performance evaluation criterion. By observing the changes in the learning curve, we can gain a deeper understanding of the model's learning capabilities and stability. Tensorboard was employed to log the learning curve throughout the active learning iterations. The learning curve depicts the relationship between the model's performance and the number of samples used for training (number of active learning iterations), assessing whether an increase in the number of annotated datasets can effectively improve the model's performance. Furthermore, the experiment also logs the model's performance under each epoch during the training process through Tensorboard, ensuring the model achieves the best performance without overfitting after each active learning iteration.

4.4 Termination Strategy

Within the active learning framework, the termination strategy is used to decide when to stop the learning process or when not to request new data from annotators. The design of the termination strategy directly affects annotation costs, time consumption, and the final performance of the model. Although the termination strategy is somewhat similar to the model's hyperparameters, it does not directly influence the model's training process. Instead, its primary objective is to reduce annotation costs and save time while maintaining performance optimization. This strategy needs to balance time efficiency and performance output, aiming for a learning process that's both efficient and high-performing.

For the training of neural network models, the setting of hyperparameters directly affects the training process and model performance. In earlier research, various hyperparameters have been thoroughly investigated and determined. Here, we referenced these studies and chose hyperparameters suitable for neural network training, namely a batch size of 8 and a learning rate of 0.00005.



When formulating the termination strategy for active learning experiments, it's crucial to focus on whether the model's performance has saturated. For this experiment, the primary concern is the average F1 score of the model on the validation set. Specifically, we aim to observe when the model based on random sampling strategy surpasses the performance of the model trained on all data, or when the rate of performance growth starts to decelerate. Charts show the performance comparison of models under both the random sampling strategy and the full-data training strategy. By the 8th iteration, the performance of the model based on random sampling has exceeded that of the full-data trained model. However, this performance improvement was not stable, leading to the decision to continue iterating. After the 9th iteration, the growth in model performance began to slow down. Considering that further iterations would increase both time and annotation costs, the experiment was decided to be terminated after the 10th iteration. This strategy allows for optimal performance at the least cost.

4.5 Execution Module

In this study, text data classification was conducted through the implementation of a series of active learning strategies. The experiment was carried out via a remote environment connected through PyCharm. Throughout the process, an explanation was added in each iteration, with a total of 9 iterations run, taking a cumulative time of 36 hours.

Random Sampling:

Implemented using the “Whole\_train.py” script, this approach handled both the full data mode and the active learning mode.

Diversity Sampling:

Based on the “diversity\_sampling\_whole.py” script, this method has three different ways of providing explanations for labels: preset explanations, manually inputted explanations, and explanations generated by the OpenAI model (an API key for OpenAI is required).

Uncertainty Sampling:

Executed via the “uncertaintyWhole.py” script, the objective of this strategy is to select data points that present the most uncertainty or challenge for the model's classification.

Dropout NN Sampling Based on BALD:

Implemented using the “BALD\_MCD.py” script, this combines the Dropout neural network with the Bayesian Active Learning by Disagreement (BALD) strategy, specifically searching for high uncertainty data points in model predictions through minimum confidence.

The primary objective of running these experiments was to conduct an in-depth evaluation of the performance of various strategies in text classification tasks. This aims to determine which strategy is most suitable for enhancing the model's learning efficiency.

5 Evaluation

5.1 Expectations and Evaluation Structure

Initially, there are several preset expectations for the framework involving human interaction in the loop:

1: Active learning can effectively enhance model performance (on both the test and validation sets) by increasing the annotated dataset.

2: Incorporating unlabeled datasets annotated with explanations has a greater impact than simply adding datasets labeled without explanations.

3: Most strategies can outperform the baseline (random sampling), and before implementing the termination policy, they either surpass or come close to the performance on the full dataset.

4: It's possible to identify the best active learning strategy, explanation annotation strategy, and model architecture strategy to achieve high performance with limited data.

5: The quantity and quality of provided explanations significantly affect the model's performance.

Based on these expectations, an evaluation framework is set up, and this framework will be applied to both the test and validation sets.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | NN structure | datasets | strategy | Annotate explanation | Performance metrics |
| 1 | None dropout | Full-datasets | None | No exp | Average F1 score, average accuracy, F1 score(label 0-8) |
| 2 | None dropout | Active learning datasets | Random sampling | No exp | Average F1 score, average accuracy, F1 score(label 0-8) |
| 3 | None dropout | Active learning datasets | Random sampling | Exp - no openAI | Average F1 score, average accuracy, F1 score(label 0-8) |
| 4 | None dropout | Active learning datasets | Uncertainty sampling | Exp - no openAI | Average F1 score, average accuracy, F1 score(label 0-8) |
| 5 | None dropout | Active learning datasets | Diversity sampling | Exp - no openAI | Average F1 score, average accuracy, F1 score(label 0-8) |
| 6 | None dropout | Active learning datasets | Diversity sampling | OpenAI exp | Average F1 score, average accuracy, F1 score(label 0-8) |
| 7 | Dropout | Active learning datasets | BALD sampling | Exp - no openAI | Average F1 score, average accuracy, F1 score(label 0-8) |

5.2 Expectation One:

In line with the expectations outlined in Section 5.1, a segmented evaluation is conducted. For the first expectation, it's essential to observe if active learning effectively enhances model performance by augmenting the annotated dataset. Hence, the learning curve mentioned in Section 4.3 needs to be observed. The hope is that, within the learning curve, the model's performance continuously rises as the number of annotated instances increases.

To reduce computational time, a representative evaluation method is used. Therefore, from the evaluation framework in Section 5.1, the evaluation method with ID 2 is chosen, utilizing a random sampling strategy, where in each round only the corresponding labels for the classes are annotated. The evaluation results are shown in the graph below. The solid lines represent the average F1 score for the validation and test sets, while the dashed lines denote the average accuracy. By establishing a linear trend line, it's apparent that the learning curve's trend, whether on the validation or the test set, aligns with the first expectation. Thus, this active learning framework has indeed effectively improved the model's performance.

5.3 Expectation Two:

The second expectation in Section 5.1 delves into the efficacy of the human-in-the-loop framework when built upon ExpBERT. It explores whether, during the annotation process, adding explanations beyond just the labels can enhance performance. Thus, the evaluation methods corresponding to ID 2 and ID 3 in the table from Section 5.1 are employed for comparative experimentation. The aim is to determine if the evaluation results from ID 3 (with added explanations) outperform those from ID 2 (with labels only). In the evaluation process of ID 3, the volume of explanations increases by one in each iteration. Here, the evaluation outcomes are determined using the average F1 scores and average accuracy scores corresponding to the validation and test sets. Data from the final iteration is displayed in the table below.

|  |  |  |
| --- | --- | --- |
|  | F1 average | Accuracy average |
| Val ID 2 | 0.6159 | 0.599 |
| Val ID 3 | **0.625** | **0.6131** |
| Test ID 2 | 0.4489 | 0.5761 |
| Test ID 3 | **0.5056** | **0.6216** |

From the table, it is evident that all evaluation results for ID 3 (with added explanations and labeled tags) outperform the results for ID 2 (with labels only). Therefore, annotation that not only labels but also provides additional explanations can effectively enhance the model's performance. The final conclusion is that Expectation Two has been met.