Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

# Evaluation of Active Learning for Short Answer Grading

*Mohandass Muthuraja, Jeeveeswaran Kishaan*

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Paul G. Ploeger
First supervisor
M.Sc. Deebul Nair
Second Supervisor

January 2019

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

| | |
|---|---|
| Date | Mohandass Muthuraja, Jeeveeswaran Kishaan |

# Abstract

Your abstract

# Acknowledgements

Thanks to ....

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Assessment of the knowledge acquired by the students is one of the most important aspects of the learning process. Different forms of assessments that exist today include multiple choice questions, fill-in-the-blanks, essay questions and short answer questions. Prior works have shown that multiple choice questions and fill-in-the-blanks fail to capture the vital aspects of the acquired knowledge such as reasoning and self-explanation [? ]. In contrast, questions which require the students to construct responses in natural language have been found to be more effective in assessing their grasp on the subject matter [? ]. Essay questions and short answer questions belong to this category. Short answer questions are characterized by the following aspects[? ]:

- the question must require a response that recalls external knowledge instead of requiring the answer to be recognized from within the question
- the question must require a response given in natural language
- the answer length should be roughly between one phrase and one paragraph
- the assessment of the responses should focus on the content instead of writing style
- the level of openness in open-ended versus close-ended responses should be restricted with an objective question design

Automatic short answer grading essentially deals with using computational methods to predict the grades for students' answers, thus cutting down the effort and time of teachers / professors. Many automated approaches have been proposed in the past for grading short answer questions. These methods compare how similar the students' answers are to the one provided by the teacher or professor and assign a grade proportional to the magnitude of similarity [? ]. Different approaches of computing these similarity measures include handcrafted pattern matching, automatic pattern matching, lexical similarity (how particular words effect other words), semantic similarity (deals with the meaning of sentences), and entailments (whether one sentence leads to another without any contradiction). Fig 1.1 shows a general workflow of automatic short answer grading as a pipeline. After creating a dataset of all the students' answers and model answers written by the teachers, useful features are extracted from the answers using natural language processing techniques. A model is developed based on these features to predict the scores and deployed for use. Such a model learns only once from the available set of labeled data and there is no feedback mechanism to fine-tune the parameters based on the wrong predictions after the deployment.

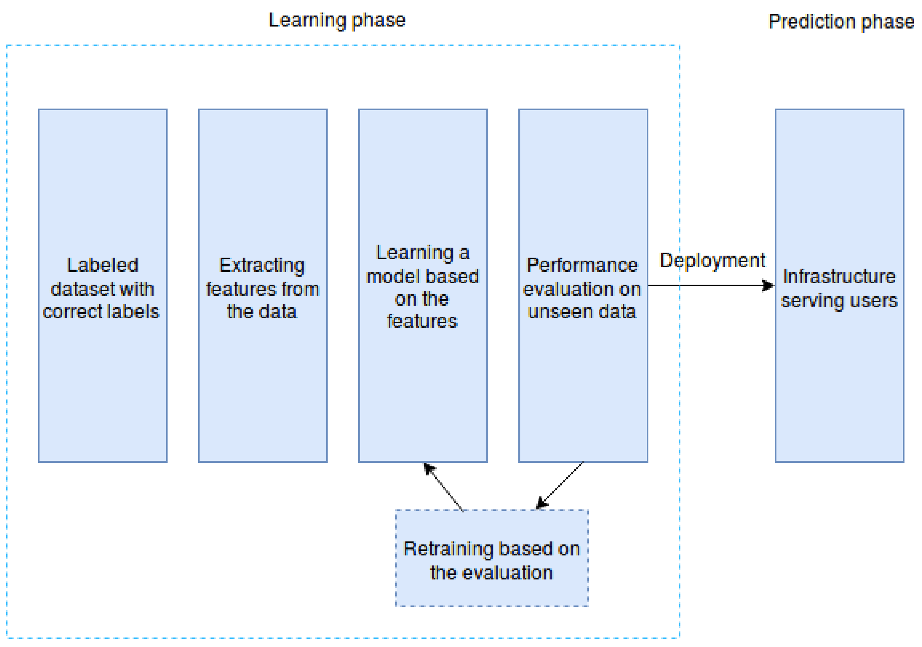

Learning phase
Prediction phase
Labeled dataset with correct labels
Extracting features from the data
Learning a model based on the features
Performance evaluation on unseen data
Deployment
Infrastructure serving users
Retraining based on the evaluation

Figure 1.1: Workflow of automatic short answer grading [? ].

This work proposes an approach where a generic scoring model tries to learn this task of measuring the correctness of each answer continuously with a human in the loop. During the learning stage, the system selects the best samples for the human to grade which would eventually contribute to its knowledge base. Thus, it tries to reduce the human effort of going through all the answers while improving it's understanding of the problem on a cyclical and iterative basis.

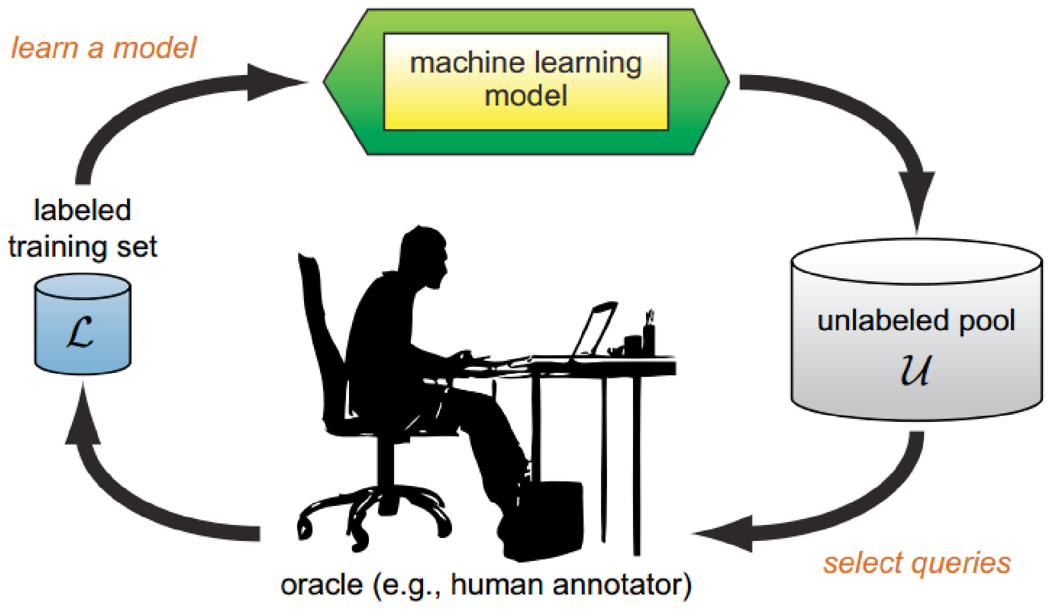


Figure 1.2: Workflow of active learning [? ].

Active learning seems to be the best choice for this task as it actively queries the human for grades of the samples it is most uncertain of. Fig 1.2 illustrates a typical workflow of active learning. It is a subfield of machine learning which works under the hypothesis that if the learning algorithm is allowed to choose the data from which it learns it will perform better with less labeled data and training [? ]. Such a model queries a user / human expert for the labels of certain data samples in such a way that it can learn to produce the desired outputs with higher accuracy. By actively selecting the data samples to label, it reduces a considerable amount of labeled training samples, thus, alleviating the problem of insufficient labeled data which is prevalent in supervised learning approaches. In addition, it would be a solution to deal with the diversity in answers as there is generally no single best response for an open-ended question.

## 1.1 Motivation

Limited availability of teachers, online learning platforms, and individual or group study sessions done outside classrooms necessitated quick and efficient assessment of free text responses. In addition, almost 30% of the teachers' time is spent on grading the assessments [? ].Computer assisted assessment / automatic grading evolved as a solution to this problem and a lot of research has been done on automating the grading of essay[? ] and short answer responses[? ? ? ]. The focus of this work would be on improving the existing solutions to grade short answer questions.

## 1.2 Challenges and Difficulties

The methods discussed so far belong to a learning paradigm called supervised learning where the right answer for every question is available [? ]. Though these approaches were able to produce decent results, they suffer from many shortcomings such as;

- the failure to capture the different wordings/phrasing of the students while trying to answer the short answer questions. It is obvious that anticipating all different ways of answering his questions is practically impossible for the professor.
- lack of sufficient amount of labeled training data in the domain to learn the models. Reasons include annotation cost, privacy, availability, and the quality of the correct answers.
- inability to capture consistent patterns of misunderstandings among students. Ability to recognize such patterns would enable the automated systems to provide useful feedback to students as to why there was a reduction in marks awarded.

- accounting for small deviations in the answers which might affect the whole meaning of the sentence (for ex. in mathematical terms, though each and every word of the student's answer align with that of the professor, a small negation or inverse operation would change the whole meaning).
- finding a way to understand the underlying concept of various students' answers and bagging the similar ones (or the right and wrong ones separately) is also a very tedious task.
- being a passive learner, these models learn the rules once and apply them on new input answers. Thus, it would be very difficult to achieve robustness when applied to new data over time.

## 1.3 Problem Statement

This work differs from the approach illustrated in Fig 1.1 by incorporating a feedback mechanism which allows the model to learn continuously based on new labeled input samples. As illustrated in Fig 1.3, we propose a model which implements active learning for the task of short answer grading.

- Active learning will be evaluated as a potential solution to overcome the deficits of automated short answer grading in this work. Thus, issues such as high cost of labeled data, understanding common misconceptions among the students, lexical diversity in answers, and inferior performance will be addressed in this work.
- Various natural language processing techniques such as semantic similarity, entailment and the best features to be extracted from the answers will be experimented for the task of short answer grading.
- Active learning strategies will be evaluated based on aspects such as seed selection, instance sampling, and batch size. The best strategies will be selected for the implementation of the final model.

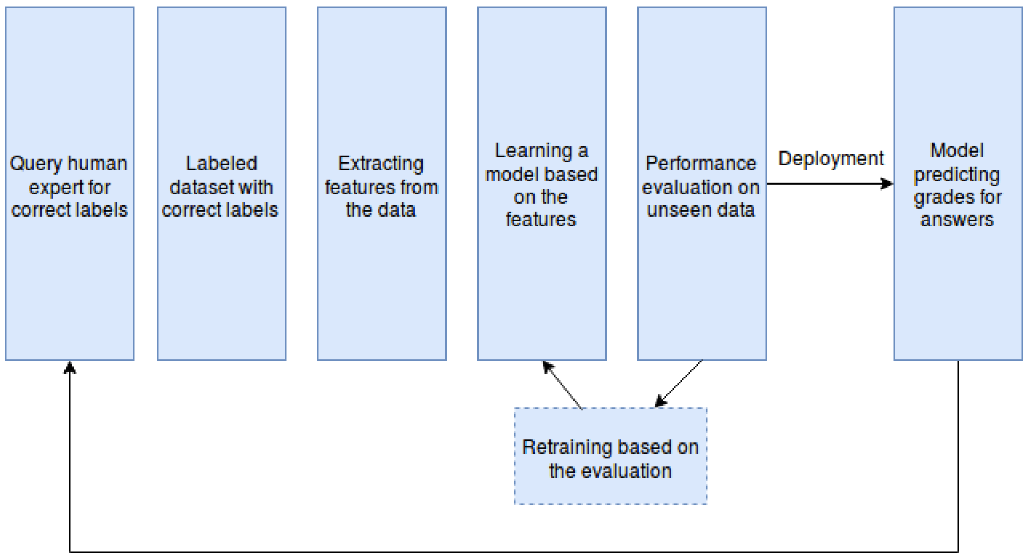

Query human expert for correct labels
Labeled dataset with correct labels
Extracting features from the data
Learning a model based on the features
Performance evaluation on unseen data
Deployment
Model predicting grades for answers
Retraining based on the evaluation

Figure 1.3: Workflow of active learning in automatic short answer grading

- A generic interactive model for assessments in different domains will be implemented based on the results of aforementioned experiments.
- All techniques will be analyzed on performance, reliability, amount of human action(clicks) required, flexibility for different domains, and user satisfaction.

2

# Natural Language Processing

Natural Language Processing evolved as a branch of artificial intelligence by combining fields such as computer science and linguistics. It helps machines to comprehend, manipulate and generate human language. Natural language processing gained interest as it seemed promising in various tasks such as human-machine interaction / communication, machine translation, spam detection, voice controlled devices, summarizing texts, and question answering in search engines [**?** ]. Thanks to the availability of a large number of unstructured data that is created everyday (for eg. social media and medical records), efficient algorithms and powerful computing systems, a lot of advancements have been made in this field in the last few decades.

Though natural language processing is an exciting field which enables everyone to communicate with the computers in their own language (as opposed to the programming language or low level language used by the computer), understanding and making sense of human language has many layers of difficulty. The complex and diverse nature of human language, unique sets of rules and grammars for different languages, new meaning of words, different meanings for the same word, and abbreviations restrict the ability of computers to make sense of natural language. Though humans are capable of expressing, perceiving, and interpreting very elaborate and nuanced meanings, they still lack at formally defining the rules that govern the language [**?** ]. This hinders the capability of taking a rule-based approach towards natural language understanding by computers as rules differ for every language and word ambiguations is commonplace.

Many Natural Language Processing tools and methods are used to bring the natural language into a format which could be made sense by computers. These techniques enable computational methods to manipulate data and make inferences from them such as how similar two sentences are, whether one sentence could be infered from another sentence, and determining the topic of an article. The most prominent NLP methods and tools are discussed below.

## 2.1 Pre-processing methods in NLP

Pre-processing the text helps in bringing the input data into a form which could be digested by the algorithm. While such methods would help in computational methods, it is important to note that some of the details could be lost from the original input and the key idea is to know the pros and cons of each of these methods and to apply them where necessary.

### 2.1.1 Tokenization

Tokenizations is a process of text segmentation which helps in splitting the text into sentences or sentences into words. Such sentences or words are called as 'tokens'. Common ways of doing this include splitting on whitespaces and splitting on punctuations. However, splitting a sentence based on whitespace or punctuations is not a straightforward task as complex words need careful attention while splitting them; e.g. "Los Angeles", "Mr. Mercedes", "i.e.", "Simon's cat", "on-campus housing" etc. To deal with such cases, rule-based or machine learning approached could be applied [**?** ].

### 2.1.2 Normalization

This method is utilized to bring all text into the same format so that manipulating them would be less biased and uniform across different forms of sentences and phrases. Simple tasks such as removing punctuations, converting numbers into their word forms, and converting all text to lowercase help in putting all the data on equal footing. Though these methods sound easy, attention must be given to minute details; e.g. converting "US" to "us" does not give the same meaning. Normalization in NLP is equivalent to normalizing all the image pixels in a computer vision task so that the algorithm doesn't give importance to more colorful or vibrant images. Other normalizing methods that are used in this report includes stemming and lemmatization.

### 2.1.3 Stop words removal

Stop words should be filtered out before processing the text further as they occur more frequently in text and contribute little to the overall meaning of it while serving the purpose of connecting different parts of a sentence. Some of the most common stop words in English include "the", "so", "and", "it", and "a" [? ]. As can be seen from the following example, stop word removal does not change the overall meaning much.

input = "The player hits the ball over the fence"
output = "player hits ball over fence"

### 2.1.4 Stemming and Lemmatization

These words convert the words into their base or common form. Stemming does this by dropping unnecessary characters, suffixes, prefixes, infixes, and circumfixes from a word in order to obtain a word stem [? ]. The results of stemming could be used to find relationships and similarities between large text documents. An example of stemming could be seen as follows.

input = "I started studying yesterday"
output = "I start studi yesterday"

Lemmatization converts the words to the base form after going through the vocabulary and doing a morphological analysis like part-of-speech of every word [? ]. Lemmatization can produce better results by utilizing WordNet's lexical database of English. It offers more accuracy at the cost of time as it is an intensive and slower process. Stemming is more useful where simple database queries are carried out and time is an important factor, while lemmatization could be used for more complex tasks such as sentiment analysis.

## 2.2 Word Embedding

Word embedding is a NLP task which converts text into an array of numbers or real valued vectors. Such a representation of words proved to be useful in many tasks such as semantic similarity measures, text classification, and machine translation. Though one-hot-encoding (where a sentence is represented by a matrix with its dimension equal to the number of words in the sentence and each row made up of 0's with a 1 in the dimension corresponding to the word) was used in the beginning as a word embedding model, it has issues such as being a sparse matrix, redundant memory, and inability to capture the context or the relationship between different words in the same sentence.

More compact and meaningful representations of words evolved later with the use of neural networks and a large corpora to learn the distributions of words on. The most popular algorithms which exist today for efficient word embedding include Google's Word2Vec [**?** ], Facebook's fasttext [**?** ], and Stanford's GloVe [**?** ]. These algorithms were able to capture both the semantic relationships and context of each word and represent them in a more compact way (dense vectors) [**?** ]. Fig 2.1 below depicts a 2D representation of word embeddings where similar words in context and meaning have been grouped together in the vector space.

## 2.3 Libraries and Toolkits

In order to do seamless experiments, a lot of libraries and toolkits came in handy while parsing, preprocessing the text, and extracting the features. These libraries include functions which would do the most repititive tasks in NLP that would require a significant time to code from scratch. The open source libraries which were used in our experiments include Numpy, Pandas, NLTK, Spacy, and scikit-learn. These will be discussed below.

#### Numpy

Numpy is a very useful library in Python which could be used to work with multidimensional arrays and matrices along with some high-level mathematical functions to operate on these arrays. In addition, it has functions with linear algebra, Fourier transform, and random number capabilities [**?** ].

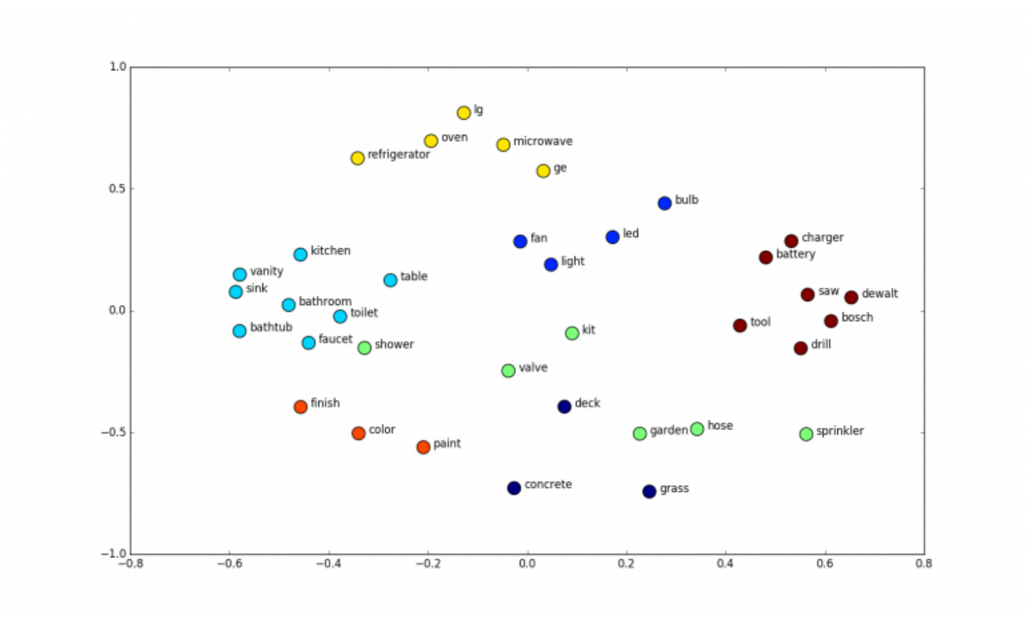


Figure 2.1: Example 2D word embedding space, where similar words are found in similar locations. [? ].

### 2.3.1 Pandas

Pandas provides high-end, easy-to-use datastructure and data analysis tools written in Python programming language [? ]. It was helpful in filtering out the required fields from the dataset and visualize the results of preprocessing steps in a structured manner.

### 2.3.2 NLTK - Natural Language Toolkit

Natural Language Toolkit is a platform for working on human language using Python. It is open-source and one of the most versatile libraries available today. It has a easy-to-use interface to over 50 corpora and lexical resources such as WordNet. In addition, it has functionalities for various NLP tasks including classification, tokenization, stemming, tagging, parsing, and semantic reasoning [? ].

### 2.3.3 spaCy

spaCy is a free open-source library for Natural Language Processing, written in the programming languages Python and Cython. It is more suitable for large-scale information extraction tasks as it is fast and efficient. With seamless interoperation with most of the deep learning libraries and frameworks. Non-destructive tokenization, Named entity recognition, Support for 31+ languages, 13 statistical models for 8 languages, Pre-trained word vectors, and Part-of-speech tagging are some of its features. Opposed to NLTK, which is widely used for teaching and research, spaCy is focused more on providing software for production usage [**?** ].

### 2.3.4 scikit-learn

scikit-learn is a simple and efficient tool for data mining and data analysis which is built on top of Numpy, Scypy and matplotlib. It is a open-source library written in Python programming language. The most prominent algorithms built in this library include classification, regression, clustering, dimensionality reduction, model selection and preprocessing. We used scikit-learn to employ different machine learning algorithms which would suit best for out task such as logistic regression and support vector machines (SVMs) [**?** ].

# 3

# Active Learning

Active learning is a special case of semi-supervised machine learning in which a learning algorithm is able to interactively query the user to obtain the desired outputs at new data points. [? ]

Supervised learning performs well when the model is trained with a large number of labeled instances. But the labeled instances are very difficult, time-consuming, or expensive to obtain. Active learning helps in overcoming the situations where there is plenty of unlabeled data is available which are difficult to label or the cost of labeling them is high. The main aim of the active learning is to improve the accuracy by labeling less number of instances. The instances that need to be labeled is determined by the active learning query strategies.[? ]

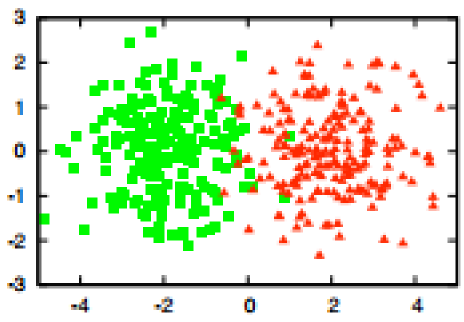


Figure 3.1: Sample data points in a 2D feature space [? ].

Let us consider the Fig 3.1. It consists of 400 data points in a 2D feature space generated using two Gaussian centered (-2,0) and (2,0) with standard deviation $\sigma$ = 1. These data points belong to two classes. (each class has 200 data points).

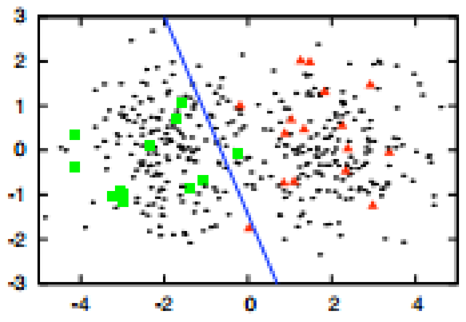


Figure 3.2: Decision boundary obtained by random sampling [**?** ].

Fig 3.2 shows the decision boundary obtained using a supervised learning approach after selecting 30 data points randomly for labeling. The line denotes the linear decision boundary of a logistic regression model is sub-optimal. This model achieves 70 % accuracy on the remaining unlabeled points because of the poor selection of data points for labeling.

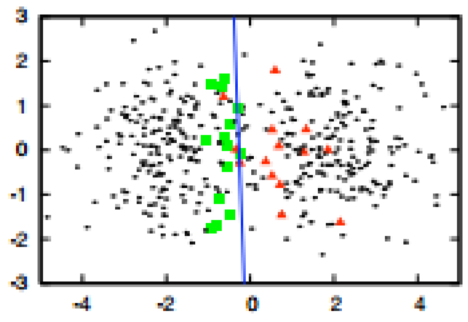


Figure 3.3: Decision boundary obtained by sampling using active learnig strategies [**?** ].

Fig 3.3 shows the decision boundary obtained using active learning approach. The active learning approach selects 30 points close to the decision boundary for labeling. By using the label for these data points, the classifier achieves 90 % accuracy which is significantly better than the approach by random sampling of data points. [**?** ]

## 3.1 Active Learning Scenarios:

There are three main problem scenario that can occur when the learner queries the user. They are

- membership query synthesis
- stream-based selective sampling
- pool-based sampling

### 3.1.1 Membership query synthesis:

In this scenario, the learner is allowed to choose any unlabeled data points which can include the a new query that the learner generates instead of the data points sampled from the underlying distribution.

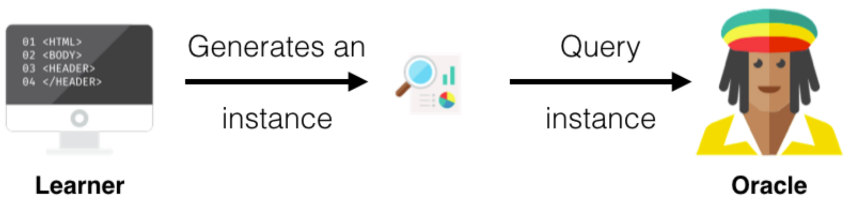


Figure 3.4: Membership query synthesis [? ].

Fig 3.4 shows the workflow of membership query synthesis.The shortcoming of this setting is that the query generated by the learner can sometimes be unrecognized for the human to label. When applying this setting to the task of handwritten classification, the learner generates hybrid characters that no natural semantic meaning. [? ]

### 3.1.2 Stream-based selective sampling:

In this setting, the instances are drawn from the data source in a sequence and the learner decide whether to query this particular instance or not.

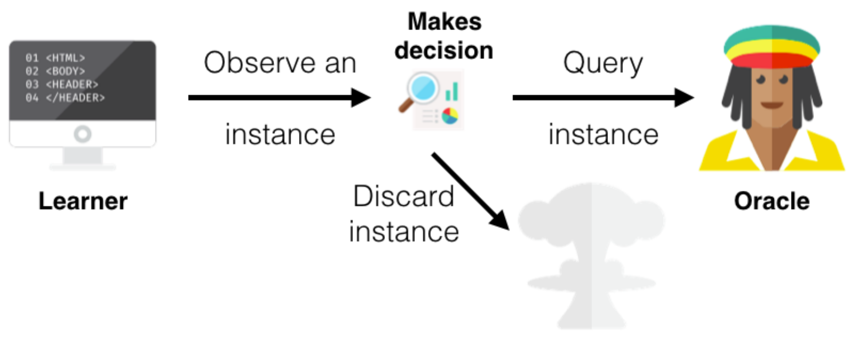


Figure 3.5: Stream-based selective sampling [? ].

Fig 3.5 shows the workflow of membership query synthesis.The decision on querying the instance is decided by various query strategies that will be discussed in the upcoming section. [? ]

### 3.1.3 Pool-Based Sampling:

The pool-based labeling setting consists of two pools of data which include a large set of unlabeled data and a small set of labeled ones. The instances are queried based on the usefulness to evaluate the other instances in the unlabeled pool.

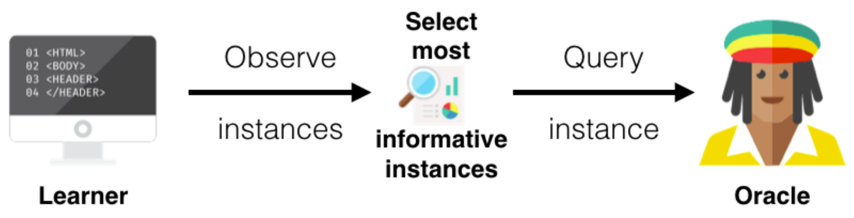


Figure 3.6: Pool-based selective sampling [? ].

Fig 3.6 shows the workflow of membership query synthesis.It has been used in a wide range of real-world problems such as text classification, information extraction, image classification and retrieval, video classification and retrieval and speech recognition. [? ]

## 3.2 Query strategy frameworks:

Passive learning approach has a large amount of labeled data which are sampled from an underlying distribution is used to train a model. Then the trained model is used for prediction. But in active learning, the learner query the most informative instance or the best instance which is the main difference between active and passive learning. There are various strategies to select the most informative query which are discussed below. [? ]

### 3.2.1 Uncertainty Sampling:

Uncertainty Sampling is the most commonly used query strategy framework. Here, the learner queries the most uncertain instance. There are three subdivisions of uncertainty sampling which includes

**Least confident uncertainty sampling:**

In the least confident uncertainty sampling, the most uncertain instance is selected. The uncertainty measure of the particular sample is the uncertainty of the class label with the highest posterior probability. The sample whose uncertainty is more is queried.

| Datapoints | Class A | Class B | Class C |
|---|---|---|---|
| D1 | 0.1 | 0.8 | 0.2 |
| D2 | 0.35 | 0.15 | 0.50 |

In the above table, the uncertainty of data point D1 is the uncertainty of the class label with the highest posterior probability which is 0.2 and similarly for D1, it will be 0.5. It is clear that the data point D2 is more uncertain which will be queried. [? ][? ]

**Margin uncertainty sampling:**

Least confident strategy only considers information about the most probable label thereby it ignores the information about the other label information. In margin uncertainty sampling, it considers the first two most probable labels and their difference is used to determine the margin. The data points with less margin are considered here and it is queried.

| Datapoints | Class A | Class B | Class C |
|---|---|---|---|
| D1 | 0.1 | 0.7 | 0.3 |
| D2 | 0.35 | 0.10 | 0.55 |

In the above example, the data point D1 has a margin size of 0.4 (0.7  0.3) and for data point D2, the margin size will be 0.2(0.55  0.35) . The data point D2 has the least margin between the most likely predicted labels. Hence data point D2 will be queried. [? ][? ]

**Entropy uncertainty sampling:**

The most general and popular uncertainty sampling measure is entropy sampling. The volume of information needed to encode a distribution is represented as an information theoretic measure which is termed as entropy. Both the least confident and margin based strategies select the instance that has a posterior class probability close to 0.5. But the advantage of entropy-based selection generalizes easily to probabilistic multi-label classifiers. [? ][? ]

### 3.2.2 Query-By-Committee:

### 3.2.3 Expected Model Change:

### 3.2.4 Expected Error Reduction:

### 3.2.5 Variance Reduction:

### 3.2.6 Density-Weighted Methods:

### 3.2.7 Uncertainty Sampling:

# 4
# State of the Art

## 4.1 Concept mapping based methods:

Concept mapping based techniques refers to splitting student answer into various concepts and graded based on the existence or non-existence of individual concepts[? ].

- Callear et al.(2001) [? ] developed a computer-assisted assessment (CAA) termed as Automated Text Marker (ATM) in which the answers are split into their smallest viable unit of concepts. All the atomic concepts are given some weight for the purpose of grading

- Leacock et al. (2003)[? ] developed an automated scoring system called C-rater for ETS(Educational Testing Service) technologies to grade responses to content-based short answers based on the syntactical matching(subject, object, and verb). This uses deep natural language processing to determine the correctness of student responses. The preprocessing steps done by the c-rater systems include spelling correction, determining the grammatical structure of each sentence, resolving pronoun reference and analyzing paraphrases.

- Brill et al. (2002) [? ] developed a question answering system called AskMSR. It uses the techniques such as query-reformulation, n-gram mining, filtering, and n-gram tiling. This system reformulates queries as declarative sentence segments to help query-answer matching. The shortcoming of this approach is that it works only when the (exact) content words appearing in a query appears also in the answer.

## 4.2 Information extraction based methods:

Information extraction system refers to extracting patterns from student answer followed by a series of pattern matching operations that includes regular expression or parse trees [? ].

- Bachman et al. (2002) [? ] developed a short answer scoring system called WebLAS. This system extracts regular expressions from a model answer to generate a scoring key. This grading system finds important segments of teacher answers through parsing and prompt the teacher to confirm the weights. It also prompts the teacher to confirm or decline the semantically similar alternatives.
- Mitchelle et al. (2002) [? ] discuss a software called AutoMark, which is developed to achieve robust grading of short answers for open-ended questions. This approach employs information extraction techniques to provide computerized marking of short free-text responses. Student answers are first parsed, and then intelligently matched against each mark scheme template, and a mark for each answer is computed.
- Oxford UCLES is an information extraction short answer scoring system that was developed at the Oxford University. which uses hand-crafted patterns by human experts to compare the answers with the model answer by Pulman et al. (2005) [? ]. This work compares information extraction with both hand-crafted and machine learning assisted pattern matching. The results of attempting to use machine learning strategies to extract patterns were not satisfactory. It was concluded that hand-crafted patterns perform better than machine learned patterns.
- Jordan and Mitchell (2009) [? ] developed a graphical user interface(FreeText Author) for short answer grading whereby the teachers model answer is converted into syntactic-semantic templates for the student answers to be matched against. The question authors dont have to be well versed in Natural

Processing Techniques as the model creates the patterns from the correct answers by its own.

- Raheel Siddiqi (2010)[? ] proposes a grading system that works on the structure of students' answer. It simply uses question answer markup language (QAML) to represent the required answer structures. The evaluation process starts with spell checking and some basic linguistic analysis, then the system matches the students answer text structure with the required saved structure to compute the final mark.
- Meurers et al. (2011) [? ] and Hahn et al. (2012) [? ] used semantic analysis to align student and target answers, including functional roles such as subject/object, gives better performance over similarity measures.
- Higgins et al. (2014) [? ] work describe the importance and the efficiency of syntactically-informed features like n-gram features, language model features, dependency features, k-nearest neighbor features and discourse segment features in short answer grading.
- Ramachandran et al. (2015) [? ] developed a short answer scoring system to provide effective scoring using automatic pattern extraction. Word-order graphs and semantic metrics(Lexico-semantic matching technique) were used to identify important patterns automatically from human-provided rubric texts and top-scoring student answers. Patterns were automatically extracted using two algorithms namely, generating patterns containing unordered content tokens and generating patterns containing sentence structure or phrase pattern information.

## 4.3 Corpus based methods:

Corpus-based methods are the statiscal methods that uses large document corpora(wikipedia, google etc.,) to obtain informations like synonyms, degree of similarity,frequency of term pairs etc. [? ],

- Mihalcea et al.,(2006) [? ] found comparable results to corpus-based measures by using word-to-word similarity measures.

- Nielsen et al., (2009) [? ]proposed a dependency-based classification component called Intelligent Tutoring System. In this approach, the instructor answers are parsed, enhanced, and manually converted into a set of content-bearing dependency triples or facets. The system uses a decision tree trained on part-of-speech tags, dependency types, word count, and other features to attempt to learn how best to classify an answer/facet pair.

- Mohler and Mihalcea (2009) [? ] use the similarity between the students answers and the teachers answers for automatic short answer grading. This work addresses topics such as comparison of knowledge-based and corpus-based measures of text similarity, evaluation of the effect of domain and size on the corpus-based measures, and a novel technique to improve the performance of the system by integrating automatic feedback from the student answers. Eight knowledge-based measures of semantic similarity and two corpus-based semantic similarities for short answer grading were successfully compared. They also created a dataset of questions, students answers and the grades for those answers given by two human annotators. Making the dataset open-source contributed a lot to other researchers to benchmark their implementations on it.

- Gomaa and Fahmy (2012) [? ] use string similarity and corpus-based similarity for automatic short answer grading. In addition to comparing different similarity measures, they compared the usefulness of two different corpora as well and this gave some insight into the useful features of a corpus. This work also reinforced the fact that short answer grading could be formulated as a similarity task.

## 4.4 Machine learning based methods:

Machine learning based approaches refers to training a model utilizing the features extracted using natural language processing techniques [? ].

- Mohler et al. (2011) [? ] like his previous approach uses the similarity between the students answers and the teachers answers for grading but incorporated several graph alignment features along with lexical semantic similarity measures This approach uses machine learning techniques and concludes that the student answers can be more accurately graded by incorporating the graph alignment features along with semantic measures. An attempt was also made to align the dependency graphs of the student and the instructor answers in order to make use of a structural component in the automatic grading of student answers.

- Fast, simple, and high-performance short answer grading system was proposed by Sultan et al. (2016) [? ] that uses text similarity features such as word alignment, embeddings, question demoting, term weighting and length ratio was proposed. A supervised learning model is trained using this features to predict the grades for short answer grading.

- Basu et al. (2013) [? ] developed an approach termed as Powergrading. This approach of divide and conquer the short answers proved to be useful in reducing the number of actions required for grading them by extending the impact of a small number of user actions when grading resources are limited. Their work incorporates clustering in short answer grading to cluster the similar answers together. This work gave an efficient solution to one of the main deficits of automated short answer grading, namely, capturing the modes of misunderstanding among the students answers.

- Zesch et al. (2015) [? ] Wekas k-means clustering to cluster the data, with k equal to the desired number of training instances. This approach first uses clustering to get clusters, then the human annotator labels the data and then a classification model is built.

## 4.5 Active learning in the context of Natural language processing

- Dmitriy et al. (2011) [? ] proposed an unsupervised language modeling based technique in order to overcome the slow learning rate due to random seed selection. In the classification task, rare classes were captured to improve the faster learning progress. This method is applied to word sense disambiguation.

- Rosa et al. (2012) [? ] explore the active learning algorithms to overcome the need for large training datasets. Three existing active learning algorithms (distance-based (DIST), diversity-based (DIV), and a combination of both (CMB)) were used to classify text from five datasets. The results indicate that the appropriate active learning algorithms (DIST and CMB algorithms) can yield performance comparable to that of passive learning with considerably smaller training sets.

- Andrew et al. (1998) [? ] formulated an approach that uses a combination of active learning approaches along with Expectation Maximization(EM) to overcome the need for the large labeled dataset. The results indicate that by labeling half of the overall dataset, the performance of fully labeled dataset is achieved.

- Simon et al. (2002)[? ] approach deal with using support vector machine active learning for text classification. They provided an algorithm for choosing which instances to request next. It uses pool-based active learning instead of random seed selection. The results show that employing active learning algorithm can reduce the effort of labeling instances in both standard inductive and transductive settings

## 4.6 Limitations of previous work

# 5
# Experimental Setup

## 5.1 Dataset

Mohler et al (2009)[? ] consists of questions from Introductory comuter science assignments along with answers given by undergraduate students. The assignments were monitored by the University of North Texas. The dataset consists of 3 assignments, each assignment consists of 7 short answer questions and each question has answers given by 30 students. Overall dataset consists of 630 students answers, their grades randing from 0 to 5, and the model answers written by the instructors.

Mohler et al (2011)[? ] created a dataset from the data structure course University of North Texas and the dataset consist of 2273 answers written by 31 students for 80 questions. The grades for this 2273 answers were given by two human expert graders in that particular domain.

## 5.2 Data Preprocessing

Each of the answer in the dataset were processed using various NLP preprocessing techniques. These methods are as listed below;

- Normalization - The answers were converted into lowercase so that all the text are in equal footing. Following that, the punctuations in the answers were removed using a regular expression function.

- Stop words removal - The most common occuring words such as ’the’, ’it’ etc. were removed from the student answers. NLTK corpus’s stop words dictionary was used for this task.
- Lemmatization - Each and every word in the student’s answers were converted into their common form / root form and the resulting text were used in the feature extraction stage. Textblob’s lemmatize function, which is learned based on the WordNet’s lexical database, was used for this task.

## 5.3 Features Used

The answers which are preprocessed as discussed in the above section were used to extract features in this stage. In this experiment, a simple feature extraction method called Bag-of-words approach was used. Each word in the students’ answers is tokenized and every token is counted based on the number of occurrences. Scikit-learn’s feature extraction method called ’CountVectorizer’ was used for this task which produces a sparse representation of the count of each words as a matrix.

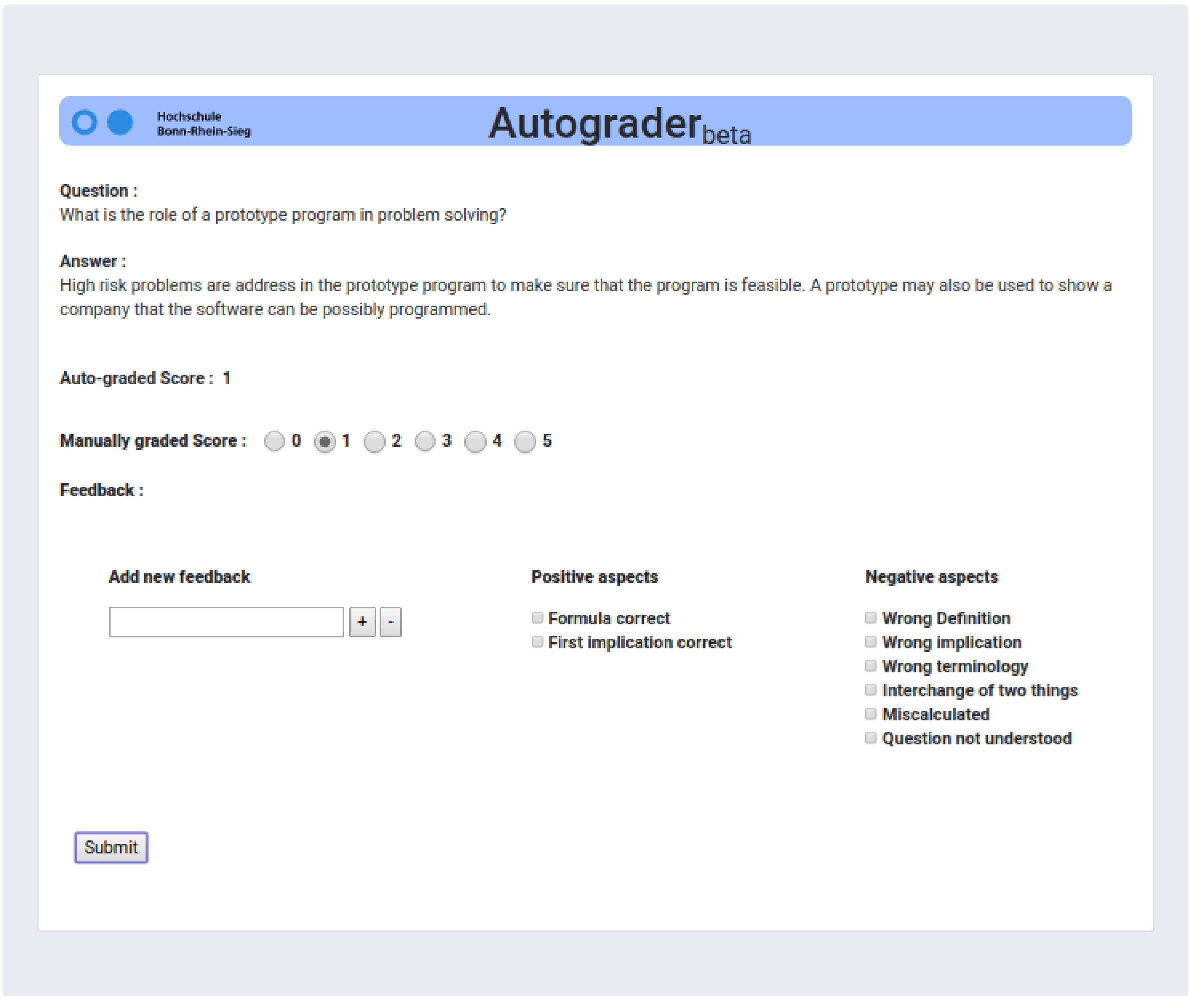
### 5.3.1 Interface

A Javascipt based web interface was used to display the process of querying the labels from a human expert / professor. Vue.js was used in the backend for the dynamic nature of display (changing the question and answer being displayed at a time). The interface consists of Fields for question, answer, grade predicted by the model, professor’s grade, positive feedback and negative feedback. A screenshot of the interface used in this experiment is shown in Fig.5.1.

## 5.4 Experimental Setup

### 5.4.1 Experiment 1

The experiments were conducted on the datasets discussed above. A Graphical User Interface was designed to label the queried instance (correct grades for each answer) by the professor. The details of this GUI would be discussed in the Interface section below.



**Question :**
What is the role of a prototype program in problem solving?

**Answer :**
High risk problems are address in the prototype program to make sure that the program is feasible. A prototype may also be used to show a company that the software can be possibly programmed.

**Auto-graded Score :** 1

**Manually graded Score :** 0 [✓ 1] 2 3 4 5

**Feedback :**

| Add new feedback | Positive aspects | Negative aspects |
|---|---|---|
| | Formula correct | Wrong Definition |
| | First implication correct | Wrong implication |
| | | Wrong terminology |
| | | Interchange of two things |
| | | Miscalculated |
| | | Question not understood |

Figure 5.1: Interface used in the experiment for the querying purpose

The raw questions and answers from the dataset were preprocessed using various NLP techniques and the required features were extracted from them to learn a model. The model would be learned initially from a few labeled instances (graded answers) and then it learns from newly labeled answers using a machine learning paradigm called Active Learning. The experimental pipeline of the experiment is shown in the figure below.

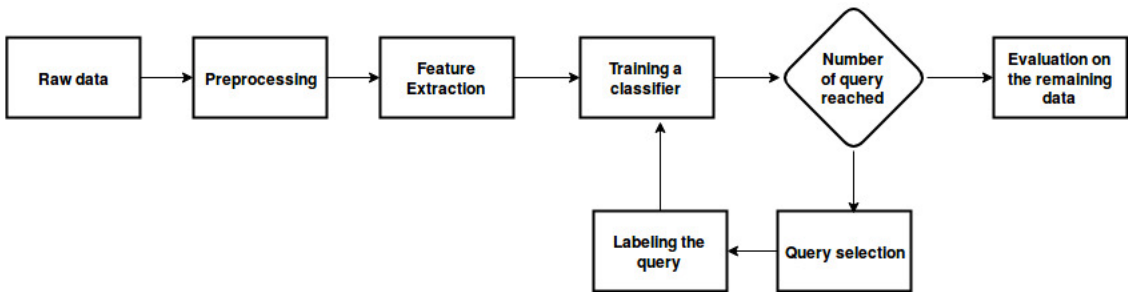


Figure 5.2: The basic pipeline of our experimental design

### Algorithm and Methodology

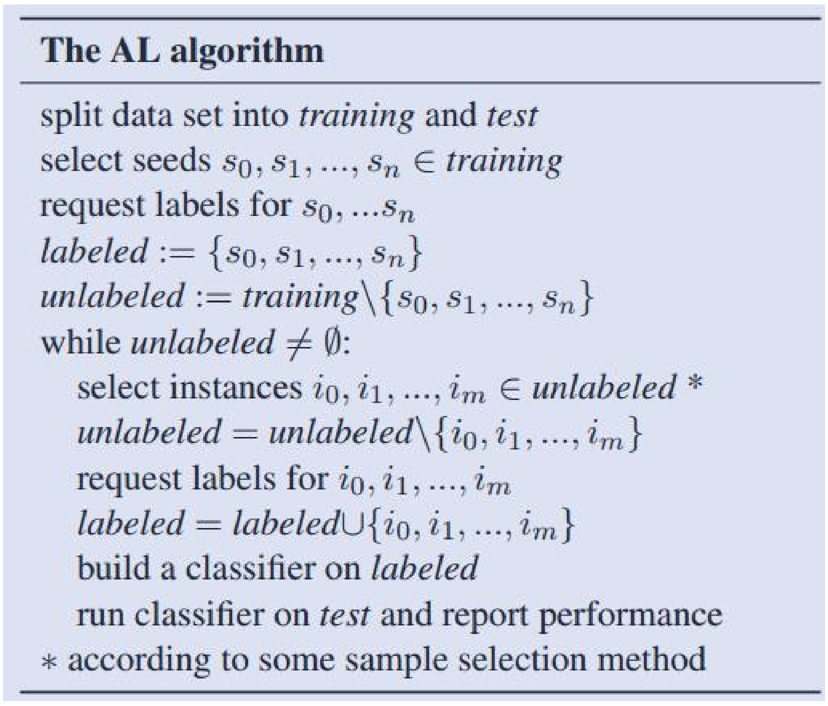
Once the dataset is cleaned and preprocessed as discussed above, Scikit-learn's Logistic regression classifier was used to classify the answers based on the features extracted. Two tasks were performed using this model.

- Binary classification - The answers which had more than 3 and up to 5 marks were considered as a positive class (1) and the ones which had less than 3 were considered the negative class (0).

- Multi-class classification - The grades of each answer was rounded off to a whole number between 0 to 5 and a classifier is trained to predict the correct class for each of them in these five classes.

The classifier was initially trained on one answer per each class and the labels / grades for the next answer for the model to learn on was selected using the Pool-based uncertainty sampling strategy discussed above in the Active Learning section. The best answer to be queried next is selected based on how uncertain / confused the algorithm is about the different classes it can take. The algorithm shown in the Fig. 5.3 below explains this process.

In our experiments, the number of instances to query from the human expert was set to 40. This is only about 7% of the Mohler et al. 2009 dataset and just 1.6% of the Mohler et al. 2011 dataset.



**The AL algorithm**

```
split data set into training and test
select seeds $s_0, s_1, ..., s_n \in$ training
request labels for $s_0, ...s_n$
labeled := $\{s_0, s_1, ..., s_n\}$
unlabeled := training$\setminus\{s_0, s_1, ..., s_n\}$
while unlabeled $\neq \emptyset$:
   select instances $i_0, i_1, ..., i_m \in$ unlabeled *
   unlabeled = unlabeled$\setminus\{i_0, i_1, ..., i_m\}$
   request labels for $i_0, i_1, ..., i_m$
   labeled = labeled$\cup\{i_0, i_1, ..., i_m\}$
   build a classifier on labeled
   run classifier on test and report performance
* according to some sample selection method
```

Figure 5.3: Pseudocode for general, pool-based active learning. [? ].

### 5.4.2 Experiment 2

In this experimental setup, users were asked to use the interface with and without the influence of active learning on just 40 answers and were asked to give feedbacks on their user experience.

The raw questions and answers from the dataset were preprocessed using various NLP techniques as discussed above and the required features were extracted from them to learn a model. The model would be learned initially from a few labeled instances (graded answers) and then it learns from newly labeled answers using a machine learning paradigm called Active Learning.

#### Algorithm and Methodology

Once the dataset is cleaned and preprocessed as discussed above, Scikit-learn's Logistic regression classifier was used to classify the answers based on the features extracted. Two tasks were performed using this model.

- Without active learning - In the non-active learning phase, the questions and answers were displayed to the user one by one, without any indication of the correct answer. The users had to decide between two grades (0 for a wrong answer and 1 for a correct answer). This setup reflects the usual way of grading answers.

- With active learning - In the phase with active learning, the answers predicted by the active learning model (which was trained with five queries) was displayed to the user. The predictions were set in the radio button for each answer so that given the correct prediction, the user doesn't have to just submit it and move on to the next question, thus reducing the effort and time of clicking the right grade.

The classifier was initially trained on one answer per each class and the labels / grades for the next answer for the model to learn on was selected using the Pool-based uncertainty sampling strategy discussed above in the Active Learning section. The best answer to be queried next is selected based on how uncertain / confused the algorithm is about the different classes it can take.

$*$

# 6

# Evaluation and Results

Both the Mohler datasets were used in our experiments of grading with Active Learning and accuracy was used as an evaluation metric.

## 6.1 Experiment 1

The tasks carried out could be categorized into two parts, namely, binary classification (where the grader predicts whether the answers are correct or wrong) and multi-class classification (where the grader predicts a whole number grade for each answer after being trained from the rounded off answer-grade pairs).

### 6.1.1 Binary Classification

The grades of the answers which are below 3 were considered as wrong answers and the others were considered as correct answers. The task includes training the initial model (Logistic Regression Classifier in our experiment) with one answer-grade pair for each class (correct and incorrect) and then 40 answers were queried from a human for the correct grade. These answers which were queried from the human was selected based on the Active Learning strategy called uncertainty-based sampling.

A comparison was made between this strategy and Supervised Learning. As can been seen in Fig. 6.1, active learning was able to achieve an accuracy of 88.14% while supervised learning was able to reach only 82.23%.

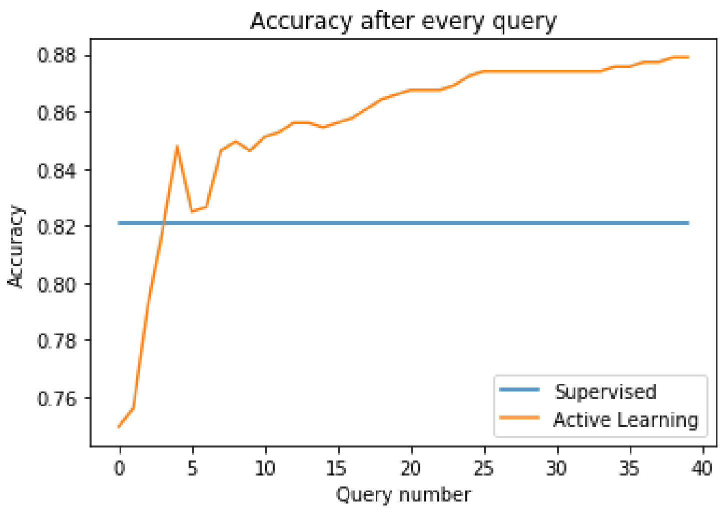


Figure 6.1: Comparison in terms of accuracy between active learning and supervised learning in the Mohler 2009 dataset.

Similar task was carried out on the Mohler 2011 dataset. As can been seen in Fig. 6.2, active learning was able to achieve an accuracy of 87.31% while supervised learning was able to reach only 87.73%. It could be inferred that with proper features and more queries, active learning could perform better than supervised learning.

### 6.1.2 Multi-class Classification

The grades were rounded off to the nearest whole number (which ended up in six classes in the range of 0 to 5). The task includes training the initial model (Logistic Regression Classifier in our experiment) with one answer-grade pair for each of class and then 40 answers were queried from a human for the correct grade. These answers which were queried from the human was selected based on the Active Learning strategy called uncertainty-based sampling.

A comparison was made between this strategy and Supervised Learning on the Mohler 2009 dataset. As can been seen in Fig. 6.3, active learning was able to achieve an accuracy of 52.21% while supervised learning was able to reach only 61.78%.

Similar task was carried out on the Mohler 2011 dataset. As can been seen in

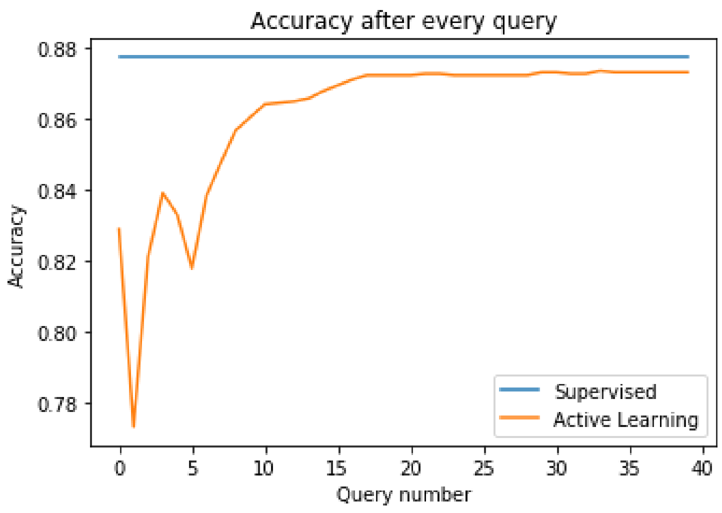


Figure 6.2: Comparison in terms of accuracy between active learning and supervised learning in the Mohler 2011 dataset.

| | | **Active Learning** | **Supervised Learning** |
|---|---|---|---|
| **Mohler 2009** | Binary classification | **88.14%** | 82.23% |
| | Multi-class classification | 52.21% | **61.78%** |
| **Mohler 2011** | Binary classification | 83.31% | **83.73%** |
| | Multi-class classification | 48.44% | **59.71%** |

Table 6.1: Experiment results on both Mohler 2009 and Mohler 2011 dataset.

Fig. 6.4, active learning was able to achieve an accuracy of 48.44% while supervised learning was able to reach only 59.71%. It could be inferred that with proper features and more queries, active learning could perform better than supervised learning in the multi-class classification.

The results have been tabulated in the table 6.1 below.

## 6.2 Experiment 2

As discussed in the Experimental Setup section above, users were asked to test the system with and without the help of active learning. A questionaire was prepared to get feedbacks from them about their experience which is given below.

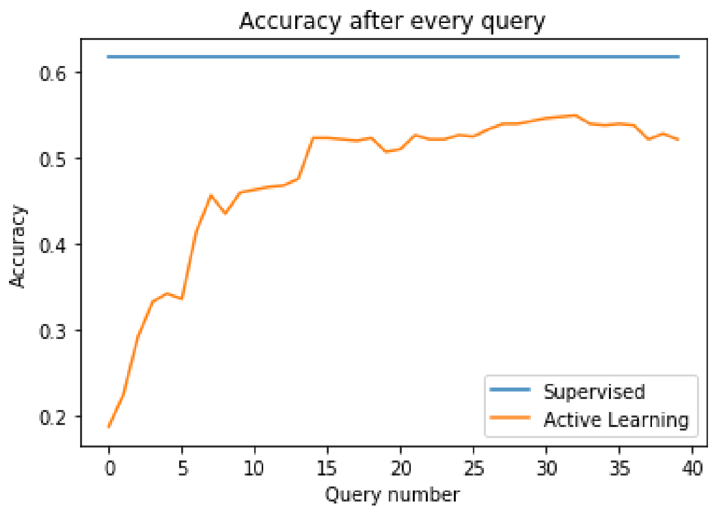


Figure 6.3: Comparision in terms of accuracy between active learning and supervised learning in the Mohler 2009 dataset.

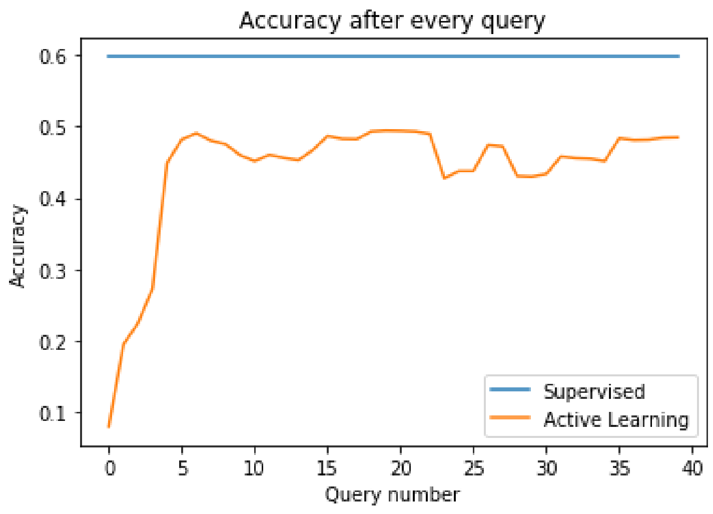


Figure 6.4: Comparision in terms of accuracy between active learning and supervised learning in the Mohler 2011 dataset.

### 6.2.1 User 1

Total clicks taken without active learning - 90 clicks

Total clicks taken with active learning - 52 clicks

- How easy is our grading to use ?
  In a scale of 1-10 , I would rate it as 6.

- Did AI assisted grading reduce the effort of grading ?
  Yes. The difference was very clear. In manual grading I need to select the grades for all the answers and have to submit the grade. While in case of AI assisted grading most of the times I just need to verify the graded answer. Hence lots of effort for grading is reduced.

- How about the readability of Questions and answers ?
  The readability of question and answers was fine. Still, Various fonts can be tried to help the user find the better font for reading.

- How about the positioning and sizing of various elements (radio button,submit button)?
  Size of the radio button can be increased. Submit button can be on the right side as most of the users are right handed.

- What features you think need to be added to enhance the experience ?
  Same question with different student answers can be displayed in a single page.

### 6.2.2 User 2

Total clicks taken without active learning - 90 clicks
Total clicks taken with active learning - 56 clicks

- How easy is our grading to use ?
  In a scale of 1-10 , I would rate it as 7.

- Did AI assisted grading reduce the effort of grading ?
  Yes.

- How about the readability of Questions and answers ?
  Readability was satisfactory.

- How about the positioning and sizing of various elements (radio button,submit button)?
  Positioning can be improved a bit as I have to move the cursor here and there for selection. Instead it would be great if the movement of the cursor was in a sequential manner.

- What you think need to be added to enhance the experience ?
  Lot can be added to make the user feel comfortable. Various display options can be created. Autograding score can be embedded in other options other than radio button. Shortcuts can be created.

# 7 Conclusions

## 7.1 Contributions

## 7.2 Lessons learned

## 7.3 Future work

# A

# Design Details

Your first appendix

# B

# Parameters

Your second chapter appendix