Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Master's Thesis

Application of Model Interpretability Techniques for Short Answer Grading

Ravikiran Bhat

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. Dr. Paul G. Plöger
Prof. Dr. Peter Becker
M. Sc. Deebul Sivarajan Nair

March 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	Ravikiran Bhat

Abstract

Automated scoring of short answer type questions via Natural Language Processing (NLP) is a field that has a substantial body of associated research in order to develop methods to reduce workload of manual graders. However, procedures that associates a reasoning behind the automated score to enhance the reliability of the assigned grade is a task that has received little attention in literature. This work focuses on applying and evaluating different strategies of model interpretability to the task of automatic short answer grading as a means of rationalizing the predicted score to the human graders. Pre investigations were carried out on two state-of-the-art model-agnostic interpretation techniques, namely Local interpretable model-agnostic explanations (LIME)[21] and SHapley Additive exPlanations (SHAP)[12]. The influence of change in feature extractors and machine learning models on LIME was analyzed, and a quantitative comparison of the interpretation levels from LIME and SHAP was done for different case studies. The results showed that LIME and SHAP tend to give better explanations when answers are restricted to one sentence length, and do not have a large linguistic diversity. Furthermore, LIME's restriction of highlighting only unigram keywords as part of the explanation, and SHAP's exponential time requirement to compute feature attributions, made them unsuitable for our application. As a means of overcoming these challenges, we develop a solution that is model agnostic, is capable of highlights phrases (bigrams and trigrams) in the explanation, and improves the autograding and generation of explanations for new data by involving a human oracle both during initial annotation of data, and later in providing feedback for the predicted scores and explanations. A prototype web based GUI tool was developed to integrate the proposed solution. Tests run with the proposed solution showed a significant improvement in the level of interpretation of the highlighted explanation when compared against the pre investigation results for LIME and SHAP.

Contents

List of Figures

List of Tables

List of Abbreviations

ASAG	Automatic Short Answer Grading
BOW	Bag of Words
CAA	Computer Assissted Assessments
CNN	Convolutional Neural Network
GUI	Graphical User Interface
LIME	Local Interpretable Model-Agnostic Explanations
LRP	Layer-Wise Relevance Propagation
LSI	Latent Semantic Indexing
NLP	Natural Language Processing
TF-IDF	Term Frequency Inverse Document Frequency
SHAP	SHapley Additive exPlanations

1

Introduction

Assessment is a task that is an integral component of learning process in educational institutions and involves a human grader evaluating the results of an examination process[14][26] . The most common forms of classroom assessments involves graders manually grading and later providing feedback to the students on corresponding to the graded subject[17]. However manual assessments often involve certain monotonic and bias errors, and consumes a large proportion of the teacher's time [23]. Furthermore in situations where there are a limited numbers of graders, or a large number of students, the assessment process introduces an overhead which is magnified further when analyzing and grading open-ended answers[14][26]. However, students still require some form of assessment in spite of these challenges which makes the inclusion of Computer Assisted Assessments (CAA) preferable over manual assessments.

Applying computer-assisted assessment in large scale exams allows to assess student's answers instead of teachers in high efficiency[11]. The evaluation models largely used by teachers are for questions involving multiple choices or requiring essay type answers. Different from multiple choice type questions, essay type questions require answers to be constructed in natural language form. Essay type questions are categorized into two types: long answer essays and short answer type essays. Long answer essays are free text essays requiring student responses in in two or more paragraphs, and assessed based on writing style (such as punctuation and spelling) and content of the answers[20][1]. Short answer type essays on the other hand require

responses between a phrase and a paragraph, and are marked based on the content of the answers only while style is not given importance during grading[1]. Techniques for automatic grading of natural language responses branch depending upon the question type. Being efficient and cost-effective solutions, ASAG systems produce good results in terms of consistency by reducing the imperfection resulting from fatigue, bias, or ordering effects of the human raters[8]. In this work, we focus only on automatic short answer grading (ASAG).

When it comes to automated assessments, assessing multiple-choice type or True/-False type questions by a system can be easily handled as long as the correct answers are available since there is no requirement of a textual analysis of the answer before grading. On the other hand, short-answer questions look for responses that demonstrate the students' ability to incorporate and communicate their knowledge using natural language. As such , automating the assessments of such questions is more difficult and usually involves leveraging Natural Language Processing (NLP) tools to:

- extract information from free text
- generation of "post-processed text and statistics comprising of normalized word forms, annotations, and numerical measurements"[3].

Furthermore most of the automated short answer grading approaches requires a certain amount of data or domain knowledge from a human instructor who is responsible for constructing model templates and keywords for all questions which are used as the reference base against which the students' answers are compared against. Many automatic short answer grading systems have been proposed where the grading is concerned with semantic similarity with one or more model answers [25][17][9]. A point worthy of mention is that, depending upon the similarity measures used for grading, the grading model can be said to be either "rule-based" or "statistical" in nature. When using rule-based similarity measures, we either seek for specific ideas or facts in student answers through different pattern matching operations, or try to detect the presence or absence of concepts in student answers by assuming student answers to be a composition of different concepts[3]. In case of statistical similarity measures, information derived from a large corpora is used to identify

degree of similarity between words[6].

Figure 1.0.1 shows the workflow involved in a generic automatic short answer grading system.

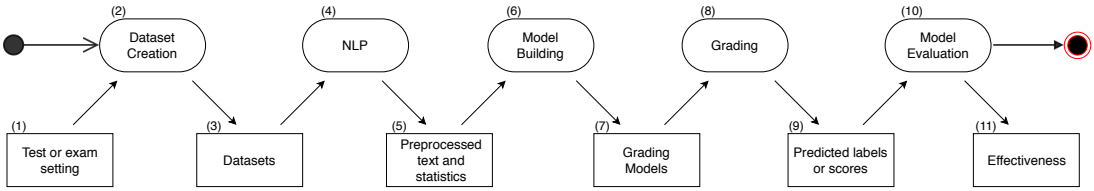

Figure 1.0.1: Generic pipeline of ASAG system development [3].

As seen in Figure 1.0.1, current trends in many short answer grading systems involves fitting a machine learning model on the features extracted using natural language processing techniques from the dataset of student answers, and calculating the score based on similarity measures obtained after comparison with model answers. This kind of grading model is also said to be "statistical" in nature. Such ASAG systems constitute a form of summative assessment system[3] which is only responsible for predicting the grades and does not involve any other form of feedback to either the students or the grader. Furthermore, being a supervised learning process, these systems work on a labeled dataset where a mechanism for handling wrong predictions does not exist.

1.1 Challenges and Difficulties

The trending automatic short answer grading systems have several limitations:

- It is not possible to account for and detect all possible patterns of concepts in students' answers and classify them separately according to different grades.
- Applying ASAG to domain specific subjects such as Mathematics often means the lack of training data to learn models. This in turn means that, ASAG systems are constrained to work with very limited sized datasets comprising of only the currently available student answers to be graded.
- Using similarity measures alone is not sufficient to get an accurate measure of the correctness of an answer. The commonly applied methodology when trying to

measure semantic similarity between words is to use a high dimensionality vector representation of words, and then using distance measures such as Manhattan distance and cosine distance to measure the semantic similarity. Word2Vec and GloVe are two popular toolkits that have been widely used to obtain word vector representations. However, these pre-trained word embeddings do not perform any useful role when used with answers that contain domain specific keywords or symbols such as those belonging to mathematical domain.

- Since open-style questions can elicit responses from students in more than one form due to the students not having the comfort of being able to recall the exact words or phrases matching the model answers, the absence of some form of diagnostic feedback to the human raters by the ASAG systems in order to indicate areas of correctness or incorrectness as a way of justifying the given grades[4] is a drawback in current ASAG systems.

1.2 Problem Statement

Keeping in mind the challenges of computer assisted assessment of free-text responses, this work aims at addressing the problem of determining the applicability of model interpretation as a solution towards satisfying the requirement of generating convincing justifications for autograded answers. With the constraint of working with a small sized dataset with a large lexical diversity in student answers, we use an active learning strategy of including humans in the loop during the grading process. This is done so that unlike in a supervised learning process, we can have a strategy that allows our learned model to adapt to any new incoming data. Furthermore, with no clear consensus on evaluating interpretability, active learning strategy comes with the advantage of allowing us to improve upon the interpreted model's explanation by relying upon a human rater's feedback. The proposed approach in this work is illustrated by the pipeline shown in Figure 1.2.1

Treating the problem of autograding of short answers as a binary classification problem where each answer is graded as correct or incorrect, we suggest a workflow where each student answer acts as the input to a model that produces an output consisting of:

- a predicted grade, and

- highlighted words in the answer indicating a positive or negative contribution towards the prediction.

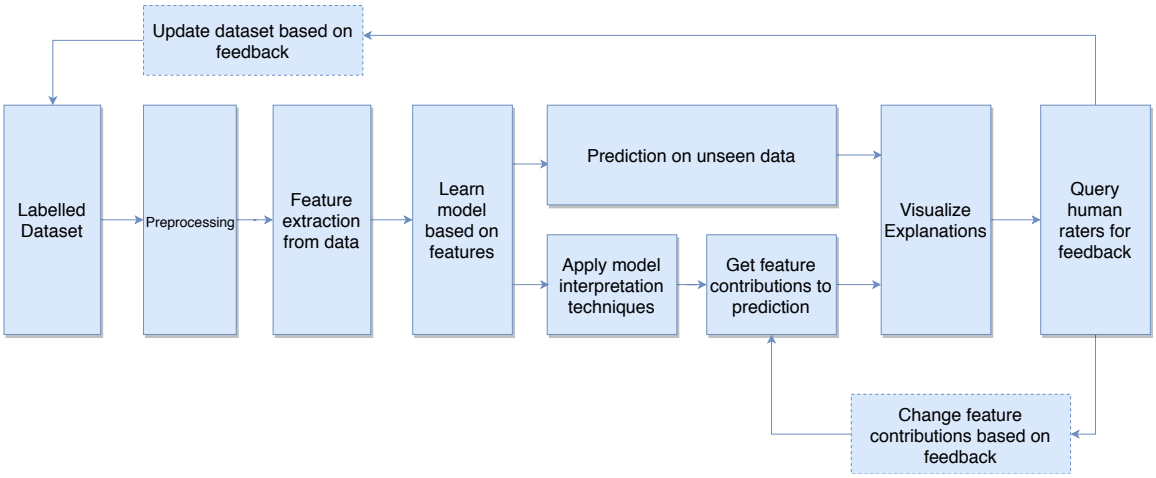

Figure 1.2.1: Pipeline of workflow involved in explaining the model predictions.

The main objectives that we aim to reach as part of this research are described below:

1. Interpretable machine learning algorithms will be applied to expose key patterns in model's decision criteria. The issue of an end-user's understanding and trust in the model's decision will be addressed through accompanying visualizations and assessment plots for the interpretations.

2. The limitation of the current model interpretation and explanation techniques will be addressed by developing a solution that involves highlighting phrases as part of the explanations.

3. Human raters' feedback on the displayed explanations will be employed in order to experiment on its usefulness in serving as a viable solution to improving the machine learning model's decisions as well as the consequent interpretation of the predictions.

1.3 Contributions

The main contributions that this research will achieve are outlined below:

- Investigative experiments on two model interpretation and explanations techniques from literature are carried out by applying them to the context of ASAG and the following are determined:

    1. Impact of different feature extractors on the final explanation resulting in the decision of the most suitable feature extractor for our solution design.
    2. Impact of different learning models on the final explanation resulting in the decision of most suitable model for our solution design.
    3. Comparison of the two model interpretation techniques and the level of interpretation achieved by them.

- A solution design that incorporates explanations in the form of highlighted phrases as part of the ASAG pipeline and involves human graders in the grading and feedback process. The design is driven by the insights and inspirations obtained from the preceding investigation experiments.

2

State of the Art

Machine learning models form the most basic as well as the core tool employed by data scientists for a diverse number of fields including business, banking, insurance or medicine. Using machine learning models can enable machines to perform exceedingly well compared to humans in several tasks, whether it is for prediction of weather or in playing games such as chess (or, since recently, the game of Go).

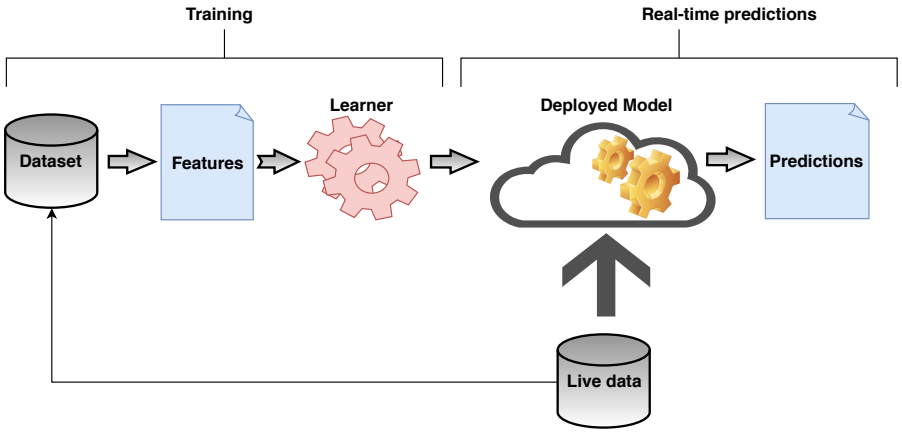

Figure 2.0.1: Basic workflow in machine learning.

However a trade-off exists between the increasing complexity of the models and the insights a human can gain about the data and the task that a machine is trying to solve. Model interpretability can be understood from Miller et. al's definition in [16] [18] : "*Interpretability is the degree to which a human can understand the cause*

of a decision." In other words, a model whose predictions can be easily comprehended by humans can be said to have a high interpretability.

Interpreting the predictions and the machine learning models plays a critical role in ensuring that the model is aligned with the problem we are trying to solve. However in predictive modeling, one needs to first answer the question: Is it sufficient to know the decision being predicted? Or is it necessary to know the reason for the prediction being made. Some applications and models do not in fact need interpretability. To illustrate, this can happen in some scenarios such as (but not limited to) the following cases:

- When models are used in low risk environments where the there is insignificant impact or consequences even in the presence of mistakes.
- Problems or applications like optical character recognition that have been extensively studied do not require interpretability as most of the issues with the model would have already been researched and solved over time[18].

The need for model interpretability arises when there is an incompleteness in the problem formulation [5] [18], i.e, the prediction from the model does not completely solve the original problem, and thus instead needs to be accompanied by a supporting explanation on how it arrived at the decision. Additionally, the more a machine learning model's decisions influences the real world, the greater the need to resolve the inconsistency between expectation and reality through some form of explanation. Another important reason that necessitates interpretability is to help detect bias that is picked up by the machine learning models from training data.

Closely related to model interpretability is the explanation given by the machine or algorithm. How well a explainer is able to "persuade" us (i.e, a human end-user) that it has achieved the intended objective will closely influence the actions of the recipient of the explanation. In other words, an explanation should be "human-friendly", in that it is able to convey the needed information together with the influencing features in a way that can be easily understood and interpreted by human end-users without having to spend too much time and effort. From the context of interpretable machine learning, good explanations can have some or all of the following properties:

- The explanation compares and highlights the major differences between the instance being predicted and a reference instance.
- Explanations are short, concise and visually intuitive.
- The most contributing feature (i.e the feature(s) that stands out) in a prediction is included and highlighted in the explanation.
- Explanations should at least exhibit local fidelity, i.e, it should correspond to how the model behaves in the vicinity of the instance being predicted[21].

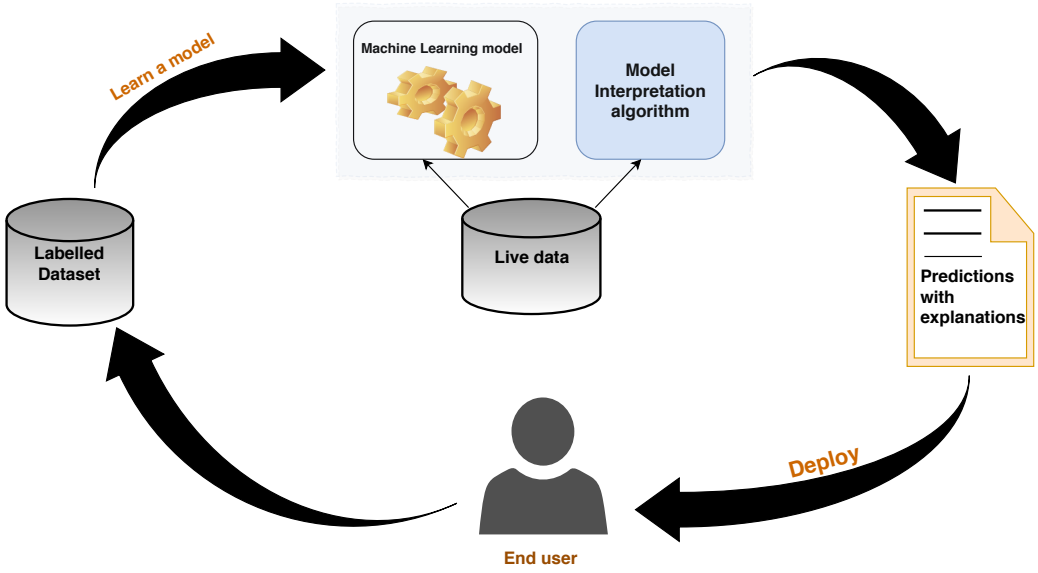

Figure 2.0.2: Basic workflow involving model interpretation and human end user.

According to [18] the methods for machine learning interpretability can be either intrinsic or post hoc. Intrinsically interpretable models are by definition model-specific as it involves selecting and training a model that is considered to be interpretable intrinsically on a modular level. Linear regression model, logistic regression, decision trees, decision rules are some common model types that belong to this category. These are also called by the terms white box models or interpretable models.

Post hoc models involve selecting and training black box models, and after training,

applying different interpretability techniques as required. By definition these model agnostic tools do not have access to the internals of the machine learning models and can only analyze feature input and outputs to the model. Most model interpretability methods in the end output some kind of summary statistics of the influence of features on the model predictions accompanied by visualizations of the summary.

2.1 Model-Agnostic Methods

2.1.1 Global Surrogate Models

Global surrogate models refers to interpretable models such as linear models, decision trees and naive Bayes that are trained to approximate the predictions of our black box models[18]. Interpreting the surrogate models allows us to explain and draw inferences about the black box model. Since the surrogate model used to fit a black box model does not need to know about the inner mechanism of the black box model, they are considered to be model agnostic.
One way of measuring how well the predictions of a black box model is replicated by the surrogate model is to use R squared measure that estimates the percentage variance captured by the interpretable model. More specifically, the definition for the R squared measure in [18] is given as:

$$R^{=}1 - \frac{\sum_{i=1}^{n}(\hat{y}_{*}^{(i)} - \hat{y}^{i})^2}{\sum_{i=1}^{n}(\hat{y}^{(i)} - \bar{\hat{y}})^2}$$

where $\hat{y}_{*}^{(i)}$ represents the surrogate model's prediction for the i-th instance, $\hat{y}^{i}$ represents the black box model's prediction and $\bar{\hat{y}}$ is the black box model's predictions' mean[18]. A value close to 1 for the R squared measure indicates that the black box model's behavior is approximated very well by the surrogate model.
The main idea behind the use of global surrogate models is that if we need to explain the predictions of a complex model, we can fit two or more surrogate models and offer multiple kinds of explanations [18].
When the distribution of the surrogate model's input is changed such that they are focused more around a single instance of data, then the model is not global any more, and we get a local surrogate model.

2.1.2 Local Surrogate Models (LIME)

When interpretable models are used to explain predictions of specific instances of black box models, they are refered to as local surrogate models. LIME, short for local interpretable model-agnostic explanation, is one of the most promising works in the field of model interpretation [21]. LIME is a method that approximates a black-box model locally in the neighborhood of a specific prediction, i.e, it fits local, interpretable models that can explain predictions for a specific instance of any classifier. The intuition behind LIME can be explained as follows:

- LIME requires an interpretable representation of the input to work with. Thus LIME first converts the dataset into an interpretable data representation. For text classifiers, this is a bag of words, i.e, a binary vector that specifies the presence or absence of a word.

- In order to explain why a black-box model gave a particular prediction, LIME tries to test how the model's prediction is affected on feeding it variations of the input. A new dataset is generated by LIME by perturbing the data input and getting the associated black box model's predictions for each perturbed input. This dataset is created for example, by adding noise to continuous features, removing words or hiding parts of the image.

- Each sampled data point (i.e, each perturbed input) is weighted by their distance from the original instance. Later these samples are used to train an interpretable model such as regression or a decision tree classifier, which is locally true (dashed line in Figure 2.1.1).

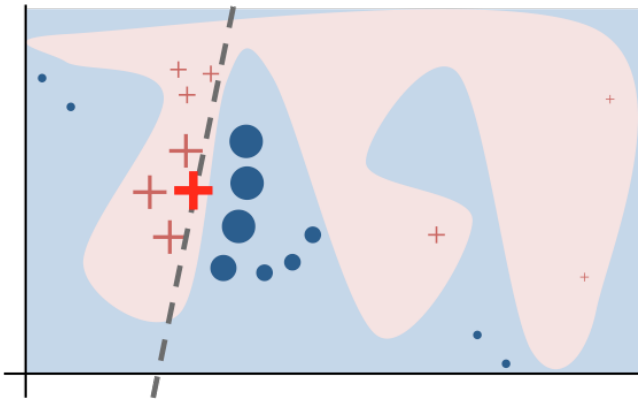

Figure 2.1.1: The two regions pink and blue, respectively symbolize two classes predicted by a black-box model. The big red cross represents the instance to be explained. The smaller crosses and dots are the sampled data points. The local explanation model is the grey dash line. (Image source : *homes.cs.washington.edu/ marcotcr/blog/lime*)

- The explanation created by approximating the underlying model locally shows the contribution of each feature to the prediction of a data sample.

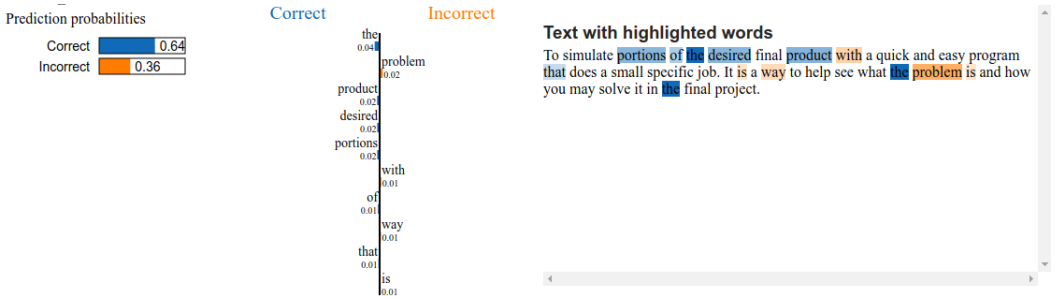

Figure 2.1.2: Example of LIME applied to classification of a textual answer as correct or incorrect.

While LIME is able to give short, human-friendly explanations, a few shortcomings in LIME have been noted[18]:

1. The current implementation of LIME uses only linear models to approximate local behavior. While this assumption may hold for a small region around the data sample, when expanding the region, this may not hold true, and a linear model will no longer be powerful enough to explain the behavior of the original model.

2. It has been observed that the results when attempting to apply LIME on answers in Mohler's dataset when using Random Forests as the model were very poor with the explanations clearly not being correct, i.e, the contributing features for the predicted class were incorrect in many cases. This could happen because of a wrong distance function, or kernel width. A hard coded kernel and kernel width is used in LIME to define the neighborhood. The question of how to find the optimal kernel width remains an issue. The kernel width is the parameter determining how much the perturbed samples are weighted based on their distance from the original instance. When it is set too low, the explanation will focus only on getting the original prediction right. When it is too large, the explanation tries to approximate the model globally which would result in the explanations not being very good.

3. Explanations tend to be unstable, i.e, variations are sometimes seen if we happen to repeat the sampling process[18].

2.1.3 Shapley Value Explanations

SHAP (SHapley Additive exPlanations) is an approach introduced by Lundberg et al [12] where the main idea is to use game theory to interpret model predictions. More specifically, the concept of SHAP leverages the concept of Shapley values from game theory to score the influences of features on model outputs and use them to explain the predictions. The definition of Shapley values in literature is given as "the average marginal contribution of a feature value over all possible coalitions"[18]. In other words, by computing the marginal contributions for all possible coalitions for an input feature, and then averaging over them, an importance value for the feature is determined. The higher the value assigned to a feature, the larger its attribution to a prediction, and the sum of the feature attributions for all features approaches the model prediction[24]. A simplified view of the idea behind SHAP can be seen as follows: the task of predicting for an instance is the "game", the actual prediction minus the result of the model represents the "gain" or the "reward", and each feature value in the instance that collaborate to get the "reward" are the "players" [18][13]. The Shapley value thus determines how the "payout" should be distributed fairly

among the features[18].

Since SHAP uses such an exhaustive approach to calculate the feature attributions, it theoretically guarantees the properties of accuracy and consistency. Accuracy indicates that the "explanations are truthfully explaining the ML model"[24]. Consistency indicates that the explanations follow humans' intuition, i.e, more specifically, it indicates that a feature's attribution value will not decrease if that feature is unaffected by other features when the model changes.

It should be noted that SHAP assigns an importance value for each feature for a prediction, i.e, it does not say anything about how the prediction changes if that particular feature is missing.

In their work in [12], the authors provide a way to compute Shapley values as a model-agnostic approach as well as a really efficient algorithm for tree based models. SHAP's python library provides many plotting methods to explain a model's prediction. Figure 2.1.3 is an example of a plot that can be used to see explanation for one instance prediction.

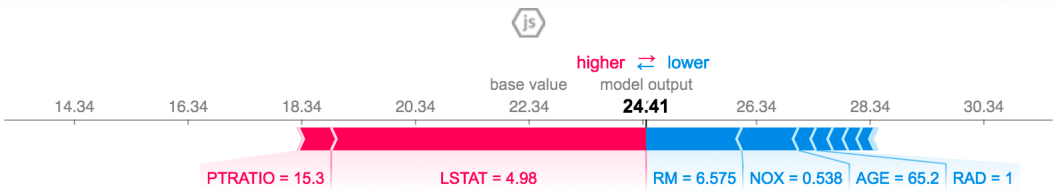

Figure 2.1.3: Example SHAP plot to visualize the a prediction's explanation. (Image source: *https://github.com/slundberg/shap*)

The authors in [12] mention that features in red are those that push the predictions higher from the base value (which is nothing but the average output over the training set) towards the model's output, while those in blue push the prediction lower. The Figure 2.1.4 shows an example type of plot summarizing the impacts or effects of each feature on the model output.

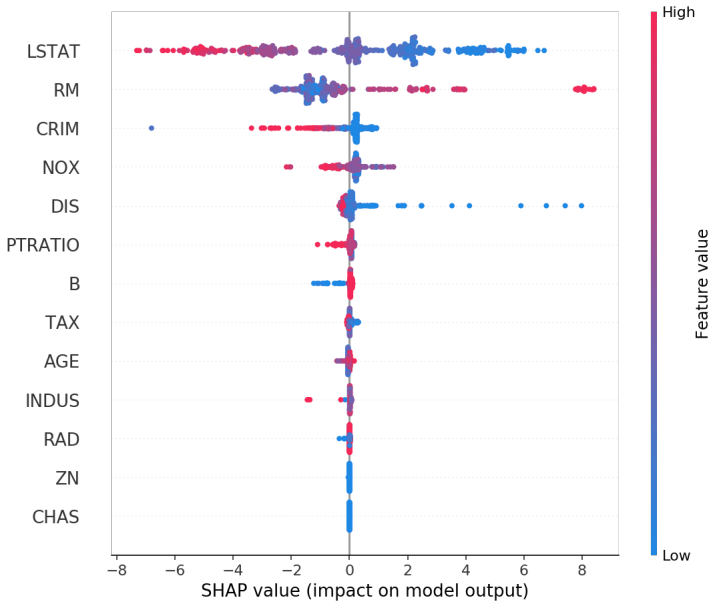

Figure 2.1.4: An example plot showing importance of different features impacting home price in an area. (Image source : *https://github.com/slundberg/shap*)

Plots such as those in Figure 3.3.10 allow us to visualize and get an understanding of the most important features for our model.
Christoph Molnar in [18] highlights a few advantages if using Shapley value explanations including the following:

- It allows you to compare a prediction to a subset or even to a single datapoint, and not just to average prediction of the whole dataset. This is something not possible in LIME.
- Unlike LIME, the difference between prediction and average prediction would be distributed fairly between feature values of the instance.

It should be noted that with n features, there are 2^n possible coalitions to be considered in addition to the simulation of the "absence" of a feature during computation of the feature attribution leading to an exponential increase in time complexity with the increase in the number of features[18][13].

2.1.4 Counterfactual explanations

Counterfactual explanations are useful in explaining predictions of individual instances in interpretable machine learning by expressing causal situations in the form: "If X had not occurred, Y would not have occurred" [18]. According to Christoph Molnar in [18], the name "counterfactual" comes due to "imagining a hypothetical reality that contradicts the observed facts". When explaining the prediction of an instance, the prediction will be seen as the "event", and the input feature values to the model that resulted in the predicted outcome will be the "causes". We can analyze the changes to the prediction outcome by changing the feature values of an instance. More specifically, we are interested in cases where the prediction flips to a different class or reaches a certain threshold. Once this is known, depicting the smallest change in the feature values of an instance required to change a prediction outcome forms the counterfactual explanation of a prediction. A more in depth theory on counterfactuals can be found in Christoph Molnar's book in [18].

Martens et. al in [15] extends this theory to document classification where they try to explain why a document was or was not classified as a particular class. More specifically, a document D's classification is explained as a set of words E such that removing all words in E leads to the classifier producing a different classification. It should be noted that this explanation is defined in the form of a set of words and not as a vector. In addition, this explanation is minimal in the sense that removing a subset of E from the document will not change the classification. This paper applies Best-first search algorithm with search-space pruning to get the explanations.

2.2 Other Methods

2.2.1 Layer-Wise Relevance Propagation

Layer-Wise Relevance Propagation (LRP) is a new technique used to explain which elements of a classifier input are important to achieve a certain classification decision [2]. LRP redistributes the output prediction score causing the classification, back to input space via a backward propagation procedure where each intermediate classifier layer is allocated relevance values. The prediction score for a class will be

equal to the sum of the relevances per layer.
Mathematically, LRP is interpreted as a *Deep Taylor Decomposition* of a neural network[7]. The basic intuition behind LRP can thus be understood as propagating backwards through our neural network starting from the output, weighting the neurons with the most contribution to the higher-layer, and then repeating this procedure[27]. The aggregate weights at the end constitute a relevance which is used to form a heatmap. Figure 2.2.1 gives a graphical illustration of the LRP procedure.

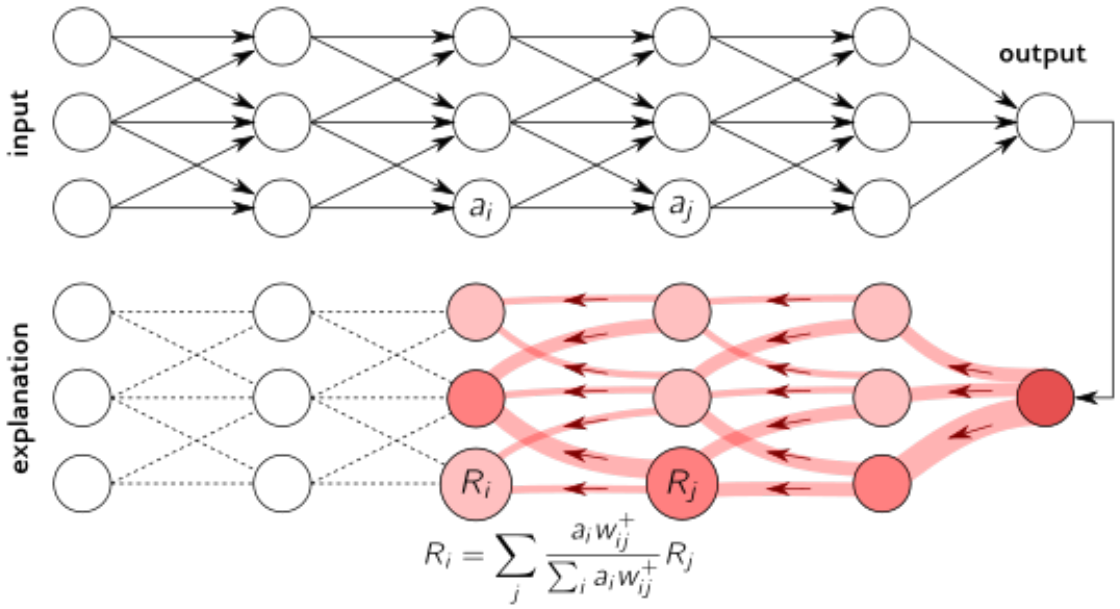

Figure 2.2.1: Illustration of LRP procedure. (Image source: *http://www.heatmapping.org/*).

The formula in the figure can be explained as follows:

- R_i = relevance value for neuron $\boldsymbol{i}$. This value is determined recursively based on the relevance value computed for the child neurons of $\boldsymbol{i}$.
- $\boldsymbol{j}$ represents the neurons in the network such that there is a path from $\boldsymbol{i}$ to $\boldsymbol{j}$ [27].
- w_{ij} = connection weights in the network.
- a_i = output of neuron $\boldsymbol{i}$.

Using the relevance values computed for all neurons, we sum over the dimension of the input vectors and assign a scalar relevance value for each input variable to the network which indicates how much the corresponding input contributes to the overall classification decision.

Arras et al's work on LRP in [2] reveals that this approach can be effectively applied to different domains including text, speech, audio, images, videos, EEG signals, etc[7]. Figure 2.2.2 below shows an example of applying LRP to predictions of 2 machine learning models, namely CNN and a bag-of-words SVM classifier for a topic categorization task.

Figure 2.2.2: LRP heatmaps of a document with positive relevance mapped to red and negative to blue [2].

2.2.2 Tree interpreter

The tree interpreter algorithm is a concept developed by Saabas[22] which is specialized to interpret Random Forest predictions. The main idea is to determine contributions of the features used for a particular prediction by tracing the decision paths of a forest. The contribution of each feature can be elucidated by calculating how the training set mean changes on following a prediction path.

Predictions from decision trees and random forests are decomposed into a bias (i.e, average of the overall training data) and the constituent terms from each independent

variable in the model [19]. An illustration of a decision path being decomposed into bias and individual contributions is shown in Figure 2.2.3.

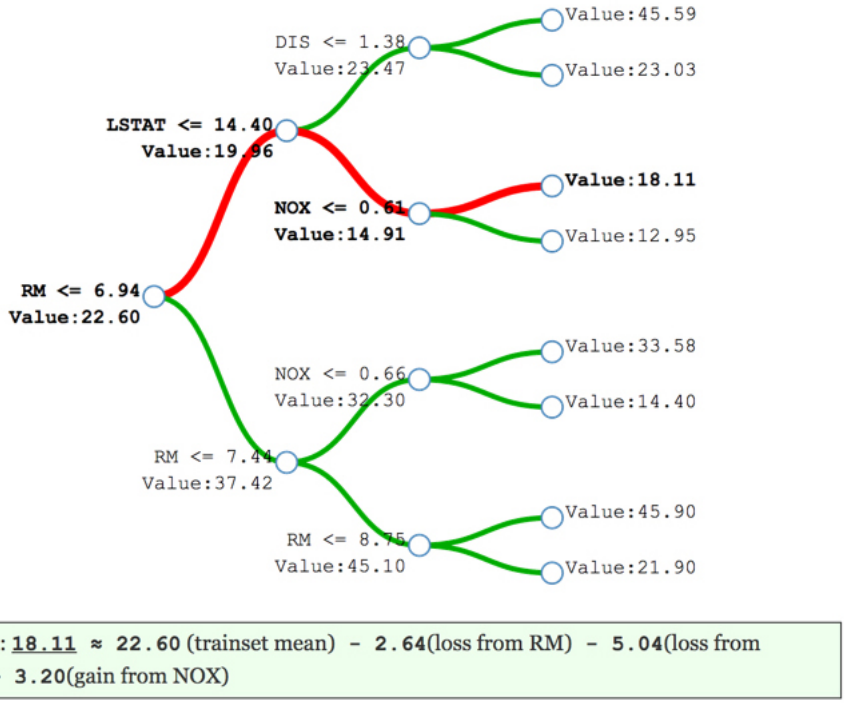

Figure 2.2.3: A decision tree with a highlighted decision path [19].

Treeinterpreter has global scope when it is used to "represent average contributions of independent variables to an overall decision tree", and has local scope when it is used to explain predictions of specific instances[19]. Treeinterpreter helps to increase understanding "by displaying average, ranked contributions of independent variables to predictions", and enhances trust when the "displayed contributions conform to human domain knowledge" or expectations [19].

2.2.3 Maximum activation analysis

In an article on machine learning interpretation by O'Reilly in [19], one of the methods outlined that help to understand complex machine learning models is maximum activation analysis which is described as a technique where specific instances

are isolated depending upon the maximum response of some model hyperparameter. O'Reilly in [19] gives the following description of maximum activation analysis:

> "*In maximum activation analysis, examples are found or simulated that maximally activate certain neurons, layers, or filters in a neural network or certain trees in decision tree ensembles. For the purposes of maximum activation analysis, low residuals for a certain tree are analogous to high-magnitude neuron output in a neural network.*"

All illustration that gives a better understanding of maximum activity analysis is shown in Figure 2.2.4

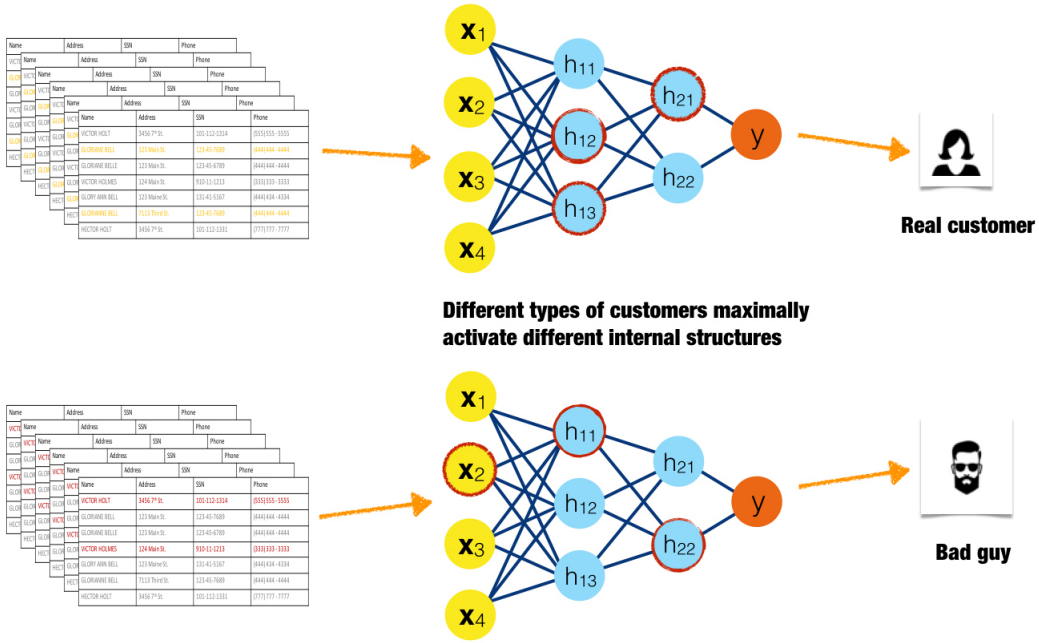

Figure 2.2.4: Maximum activity analysis illustrating different inputs activating different neurons in a neural network [19].

As shown in the Figure 2.2.4, maximally activated neurons (highlighted in red) differ depending upon the inputs, and is thus clear that each input class is treated differently by the model's internal structure. According to O'Reilly, by determining the relation between specific observations and the parts of the response function they activate to the highest degree, maximum activation analysis helps to gain a better clarification of the internal mechanisms of complex models and thus gain a better

understanding of it.
The authors mention that complex response functions can be locally explained in a much more comprehensive manner by pairing maximum activation analysis with LIME. Furthermore, trust is enhanced by showing that a complex model uses different internal mechanisms for handling different records but identical internal mechanisms for similar records. Trust is further heightened when maximum activation analysis is used as a form of sensitivity analysis and when the human domain knowledge matches the interactions[19].

3

Pre Investigations

This chapter describes a series of investigative experiments which are responsible for determining the main areas of focus in our solution design. A description of data preparation steps involving the selection of the datasets, applying preprocessing steps using NLP techniques and categorization of questions is first discussed. We then give a detailed explanation of the test case driven experiments and analyze their results. The observations and important insights from the experiments is summarized in a discussion at the end of the chapter.

3.1 Data Preparation

1. To evaluate the model interpretation approaches, experiments were conducted on datasets from two sources. The two datasets are described in the following text:

   Mohler's Dataset

   The dataset consists of a set of questions taken from assignments given out for the Data Structure course at the University of North Texas [17]. The answers were collected from a class of 31 undergraduate students via an online learning assignment. The dataset was created by working on student answers for 80 different questions and a total of 2273 answers. The answers were scored independently by two human graders on scale ranging from 0 (completely incorrect) to 5 (perfect answer).

Neural Network Dataset

This dataset consists of 17 questions administered as part of the Neural Networks course examination at the University of applied sciences Bonn-Rhein-Sieg. The answers were collected from a class of 39 students from an online examination conducted using Jupyter Notebooks. The dataset was created by using the student answers for the first 17 questions which required mandatory answers. The answers were scored on an integer scale as either 0(completely incorrect), 1 (partially correct) or 2 (perfect answer).

2. **Data Preprocessing**

A separate model is learned after extracting features from the answers to each question. Before extracting features and learning a model, the answers from the two datasets are subject to preprocessing using NLP techniques.

- By leveraging NLTK library and a regular expression function, the model answers and student answers were tokenized into words and all words were converted into lowercase form.
- The segmented text obtained from the previous step was then subjected to stopword removal where words such as "a", "the", "is", etc. that do not contribute further meaning to phrases are removed.
- Lemmatization was performed in order to obtain the base / dictionary form of a word after removing the inflectional endings in each word.

3. **Categorizing and Selection of Questions**

Since a new model is learned for each question, we attempt to narrow down the type and attributes of the questions for which our interpretation is most suitable and gives the best results in terms of human understanding. Towards this end, categorized the questions into different types and are used as case studies in our experiments. We do the categorization by looking at the type of information a question is trying to extract from a student as well as the penned down model answers for that question. For each category, we select sample questions to act as our case studies in our experiments. The samples

are selected such that each question has at least 25% or higher number of student answers belonging to each class of grades.

The questions which are used as test cases belong to the following 3 types:

- Short 1 sentence answers requiring presence of specific keywords matching those in the model answers. The following are example questions taken from Mohler's dataset that belong to this category:

Q.No.	**Question**	**Model Answer**
1.1.	What is the role of a prototype program in problem solving?	To simulate the behavior of portions of the desired software product.
1.5.	What is a variable?	A location in memory that can store a value.
2.7.	What is the role of a header-file?	To store a class interface, including data members and member function prototypes.
4.5.	How many dimensions need to be specified when passing a multi-dimensional array as an argument to a function?	All the dimensions, except the first one.

Table 3.1: Sample test cases for first category.

- Closed end questions used to check retention and mandatorily requiring specific terms and keywords as the answer. In other words, these are questions that can be answered in a few words (max 5-6 words). Following are a few questions belonging to this category:

Q.No.	**Question**	**Model Answer**
2.1.	What is typically included in a class definition?	Data members (attributes) and member functions.
2.5.	How many constructors can be created for a class?	Unlimited number.
7.4.	How are linked lists passed as arguments to a function?	By reference.
11.1.	What are the elements typically included in a class definition?	Function members and data members.
12.10.	How many steps does it take to search a node in a binary search tree?	The height of the tree.

Table 3.2: Sample test cases for second category.

- Questions requiring descriptive answers used test student's knowledge on specific concepts. The answer length varies around 2 to 4 sentences. Following example questions belong to this category:

Q.No.	**Question**	**Model Answer**
3.3.	How does the compiler handle inline functions?	It makes a copy of the function code in every place where a function call is made.
12.4.	Briefly, how does selection sort work?	It selects the minimum from an array and places it on the first position, then it selects the minimum from the rest of the array and places it on the second position, and so forth.
12.8.	What is the Euler tour traversal of a tree?	A walk around the tree, starting with the root, where each node is seen three times: from the left, from below, from the right.
12.9.	How do you delete a node from a binary search tree?	Find the node, then replace it with the leftmost node from its right subtree (or the rightmost node from its left subtree).

Table 3.3: Sample test cases for third category.

It should be noted that all questions in the Neural Network dataset consisted of very long descriptive answers that could only be categorized into the third category. We use the Neural Networks dataset in experiment 1 to verify the results observed from Mohler's dataset, while in all the other succeeding experiments, we mostly use the questions categorized above as our case studies.

3.2 Benchmarking Experiments

Two of the existing model interpretation techniques from literature, namely LIME and SHAP are the main focus of the experiments carried out in this section. The investigations conducted in this section help us determine the most suitable feature extractors as well as the machine learning model for our solution design. In addition, the observations from the experiments are used to narrow down the shortcomings of the two model interpretation techniques that we would like to address in our future solution design. The experiments in this section therefore serve as a benchmark that heavily influences the direction of our approach discussed in the next chapter.

3.2.1 Experiment 1 - Influence of feature extractors on LIME explanations

The first experiment conducted uses Bag of Words (BOW) approach as the basis for extracting ngrams and use them as predictors for the scores. Multinomial Naive Bayes classifier provided by Scikit-learn was then used as the learning model in order to classify the discrete features (word count or fractional counts in this case). We then apply LIME (discussed in the previous section) on the unseen test cases to get an explanation that is locally faithful to the classifier. This experiment is divided into 2 parts as follows:

1. Scikit-learn's 'CountVectorizer' is used as our feature extractor to get a sparse matrix representation of the token counts from the student answers in the training set.

2. We experiment the effect of applying Term Frequency Inverse Document Frequency (TF-IDF) normalization to the sparse matrix of occurrence counts

on the final LIME explanation. Scikit-learn's 'TfidfVectorizer' is thus used in place of 'CountVectorizer' to get a matrix of TF-IDF features.

The accuracy of the learning model is the main performance metric used in the experiments. No specific evaluation metrics exist to analyze the 'goodness' of a generated explanation. As such we rely on a combination of the model accuracy and visual observation to analyze and come to a conclusion about how good or bad a generated explanation is.

3.2.2 Experiment 2 - Influence of machine learning models on LIME explanations

In this setup, we experiment with different learning models in order to test their influence on the interpretation and explanation given by LIME. The feature extractor used in this experiment is depends upon the results of the previous experiment. Apart from the Naive Bayes classifier used in the previous experiment, we use the following learning models:

- We use a convolutional neural networks (CNN) model similar to the one proposed by Kim Yoon for sentence classification in [10]. The motivation behind experimenting with a CNN model is that convolutional filters can supposedly be used to learn representations beyond 5-grams efficiently without having to represent the whole vocabulary. The network used looks as shown below:

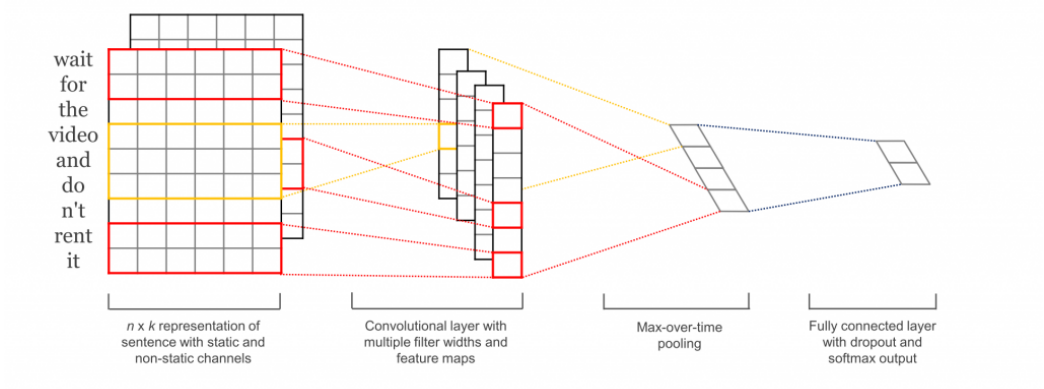

Figure 3.2.1: Model architecture for an example sentence [10].

We do not rely on any pre-trained word vectors when embedding words into low-dimensional vectors in the first layer. 1D convolution layer is used with multiple filter sizes of 2,3 and 4 words. The convolution layer's result is maxpooled into a long feature vector. For regularization, dropout is employed on the penultimate layer, and a softmax layer is used for classification of the result.

- We then experiment with a regression model and compare the three models based on the observations of the validation accuracy and its corresponding influence on the explanations for all case studies.

3.2.3 Experiment 3 - Comparison SHAP and LIME's interpretability

In this experiment, we apply Shapley value explanations as the model-agnostic interpretation technique for our predicted scores. Similar to the previous experiment, TF-IDF is fixed as the feature extractor, and Multinomial Naive Bayes classifier is used as our learning model. The SHAP (SHapley Additive exPlanations) library introduced by Lundberg et al.[12] is used to get feature attributions for each prediction. The goal of this experiment is driven towards achieving the following results:

- We use SHAP values to visualize a global overview of the impact of the top k features on the model output.
- We analyze the global interpretation obtained from SHAP and contrast it with the explanations generated by LIME.

We first record our observations from observation made from the plotting obtained from SHAP. This is followed by a quantitative benchmarking to compare the level of interpretations obtained from LIME and SHAP for all case studies from each category of questions.

3.3 Benchmarking Experiments' Results and Analysis

The experiments were carried out by selecting representative samples of question and answers from both datasets. During the experiments, the grading is treated as a

binary classification task. In other words, student scores above 3 were labeled as correct and student answers graded 3 or lower were labeled as incorrect when using Mohler's dataset. In case of Neural Networks dataset, answers graded 2 were labeled as correct and all other grades (i.e, 0 or 1) were labeled as incorrect.

During selection of representative samples for the experiments, to prevent data being biased, only those questions where answers included sufficient number of labels of both classes were selected in order to ensure that the model would generalize well on unseen data.

3.3.1 Experiment 1 - Influence of feature extractors on LIME explanations

A new model is learned for every question. We start with the first sample test case belonging to the first category described in the data preparation section of this chapter. This sample test case is also used to learn a model in all the following experiments. Using Naive Bayes classifier, the model achieves an accuracy of 90.90% for both CountVectorizer and TfidfVectorizer as feature extractor for the first two case studies in category 1 (i.e questions 1.1 and 1.5). Identical low accuracies of 36.36% and 54.54% was similarly observed for questions 2.7 and 4.5 as well. The interpretations seen using LIME is similar with both feature extractors for majority of the student answers in the test data. Consider the following instance where LIME gives the same explanation for both CountVectorizer and TfidfVectorizer:

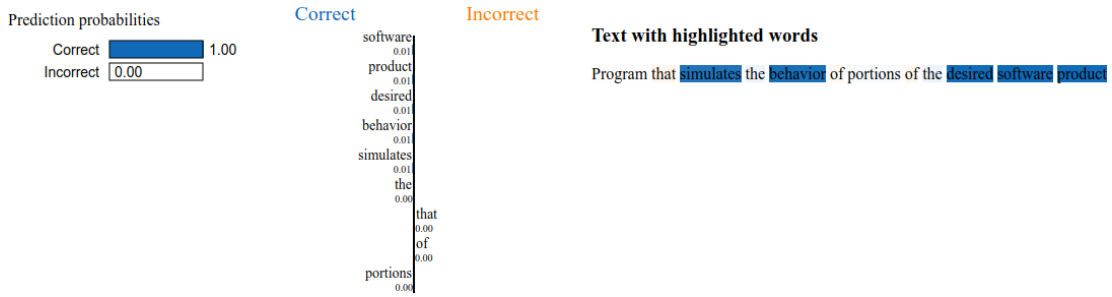

Figure 3.3.1: An example student answer that is predicted as 'Correct' and identical explanation in both cases (i.e, when using CountVectorizer and TfidfVectorizer) .

Observing the predictions and explanations on unseen data, it was seen that

while the identified features (i.e, unigrams) contributing towards a decision was identical in most cases for both feature extractors, a few instance explanations where TfidfVectorizer was used as feature extractor showed minor differences by highlighting additional features in the explanation. One such example is shown in Figure 3.3.2 for a sample instance from question 1.1:

Figure 3.3.2: In the above figure, (a) shows the interpretation for an 'Incorrect' student answer with CountVectorizer as the feature extractor. (b) shows the interpretation when TfidfVectorizer is used.

The experiments repeated by learning a new model for each case study in the second and third category demonstrated identical results in terms of accuracy as well explanations with minor differences for both feature extractors similar to that observed in case of the tests conducted for the first category. Some general observations made on the resulting LIME explanations from the tests on the instances from the three categories of questions are summarized in the following text:

- For the selected case studies in the second category, it was observed that for accurately predicted scores, the explanations tend to correctly highlight the required unigrams in the explanation. Figure 3.3.3 illustrates example explanations for sample instances from question 2.5:

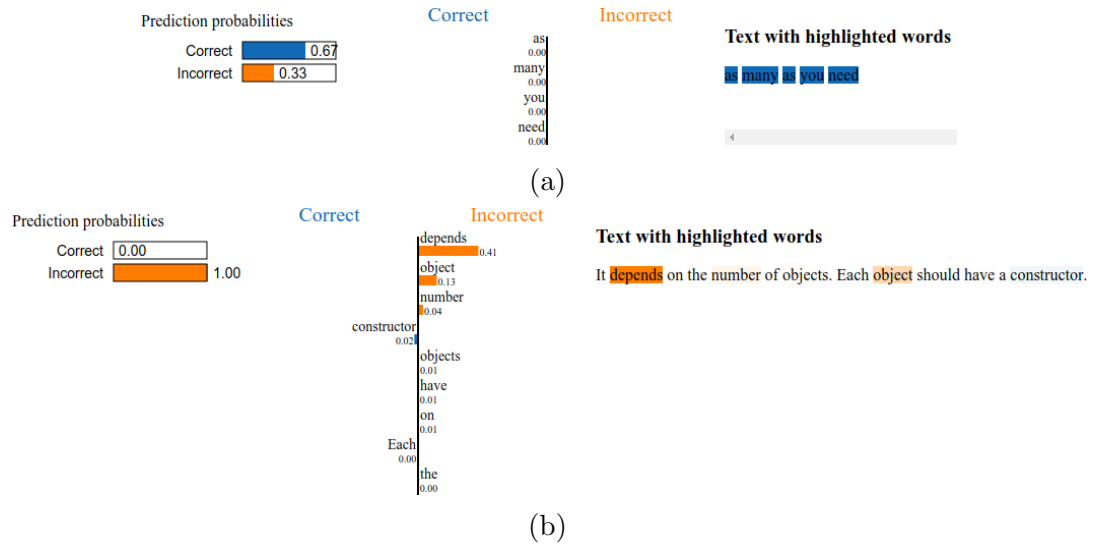

Figure 3.3.3: Sample explanations generated for (a) 'Correct' and (b) 'Incorrect' student response.

- Due to the lower model accuracy, there are higher inaccurate predictions and for such predictions we get unacceptable predictions. One such example is shown in Figure 3.3.4 for an instance from question 2.7:

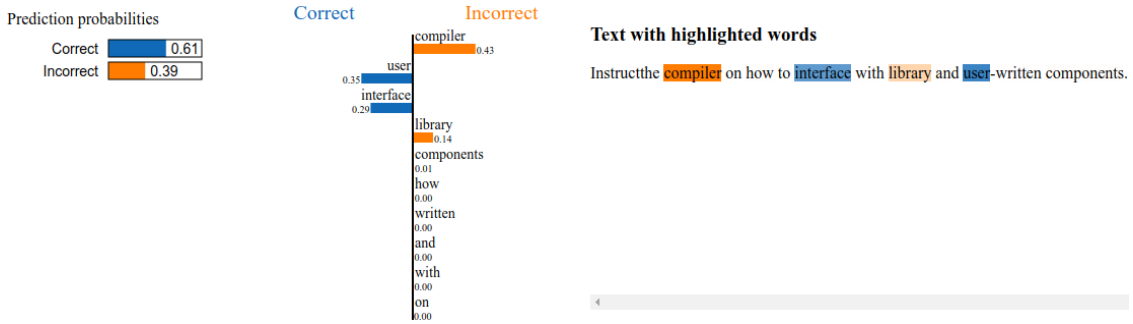

Figure 3.3.4: An example instance with a wrong prediction and wrong explanation.

- Since questions belonging to the second category can only have a narrow range of variations in the answers, the subsequent features extracted from such answers and then the resulting interpretation (by which we mean the highlighted unigrams) is sufficient to allow a human end user to get a good understanding of the cause of the decision.

- For answers belonging to the third category of questions, the answers tend to be more diverse and the size is at a minimum of two sentences or greater in length. In this case, even though the answer grades may be predicted accurately, the explanation from LIME which is limited to highlighting unigrams becomes inconceivable in majority of the cases. For example consider the explanation shown in Figure 3.3.5 for an instance from question 12.4 from category 3:

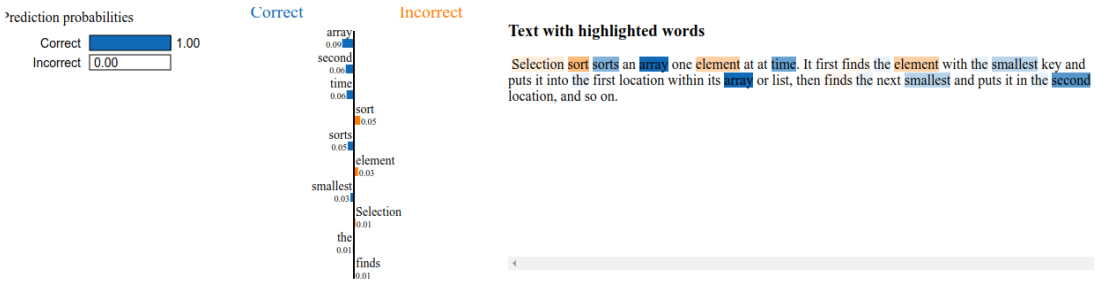

Figure 3.3.5: Sample explanation for comparatively large and diverse student answers.

Similar results are observed with the Neural Network dataset. Consider an example case study where the following question needs to be answered: *"Explain the Bias Variance Dilemma!"*. The penned down sample answers give the following hints to the human graders for "Correct" answers (i.e, for a score of 2 points):

- A supervised learning problem.
- 2 sources of errors: variance of data / bias of model.
- If high bias=> good fit, but high variance.
- If low bias=>low variance but also bad fit.
- Makes generalization difficult.

The Figure 3.3.6 illustrates explanations for a sample "Correct" and "Incorrect" answer.

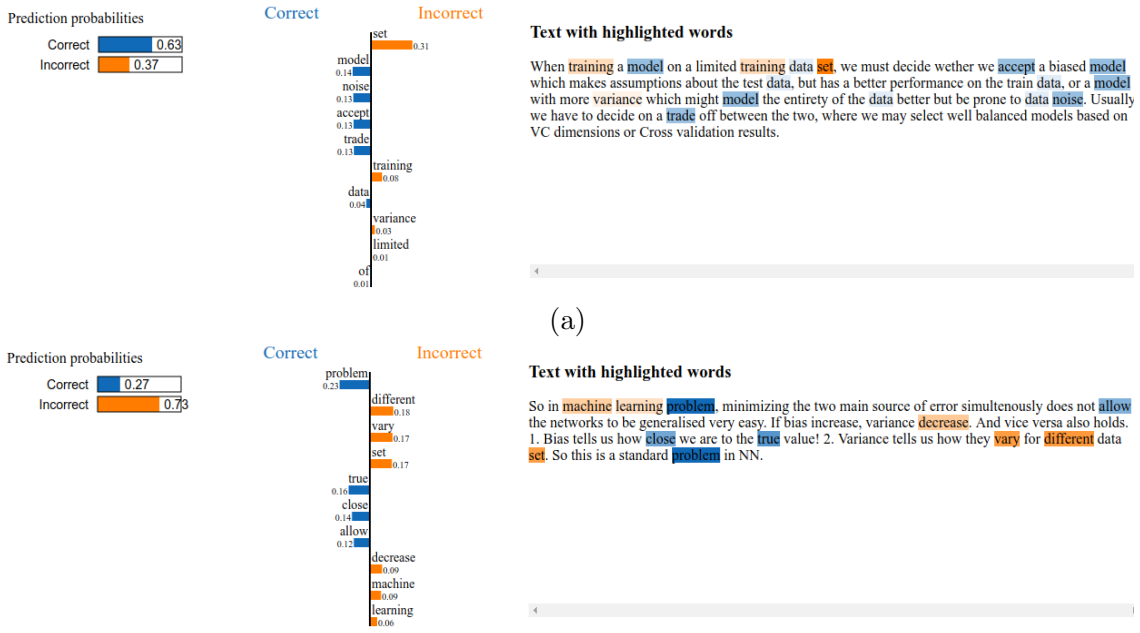

Figure 3.3.6: Sample explanations generated for (a) 'Correct' and (b) 'Incorrect' student response from the Neural Network dataset.

Explanations such as the one generated by LIME are thus not suitable for questions where the student responses show a large diversity, making it difficult to pinpoint specific features in such answers and does not provide a convincing interpretation of a model's decision.

Since neither of the two feature extractors significantly outperform the other, we choose to use TF-IDF as the feature extractor in the further experiments simply due to its ability to discount frequent and noisy words.

3.3.2 Experiment 2 - Influence of machine learning models on LIME explanations

Using the the sample question 1.1 from the category 1 as a case study, we get a validation accuracy of 66.7% on training the model for 100 epochs. Model accuracy and the subsequent interpretation are closely interlinked, meaning that if the underlying model performs badly in predicting the real outcome, then the

following interpretation is also affected and becomes irrelevant. Due to the low accuracy, the resulting predictions and also the explanations generated by LIME are poor when compared with those of experiment 1. For example consider the following explanation generated for a test instance:

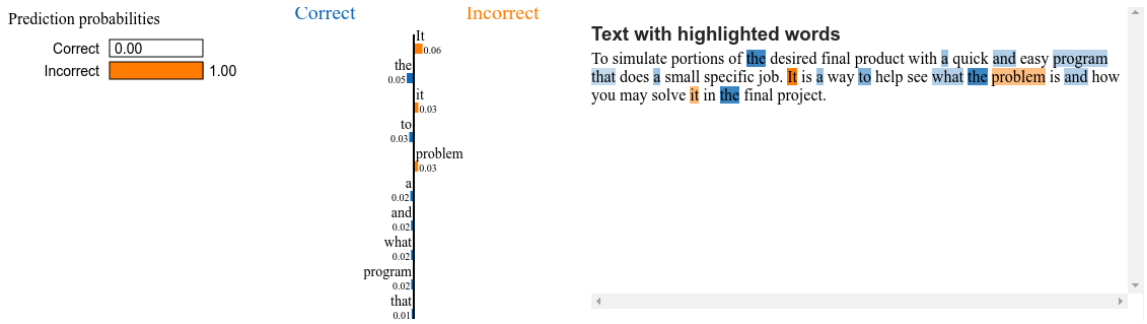

Figure 3.3.7: An example student answer that is wrongly predicted with an unconvincing explanation.

The above example shows one instance where the student answer was wrongly predicted as 'Incorrect' as well as an explanation where the key parts of the answer are not highlighted leading to an inconceivable explanation. Some test instances were seen to have redundant text highlighted as the main contributing features, while a few instances showed mismatch between the generated explanation and the prediction. One such instance is shown below:

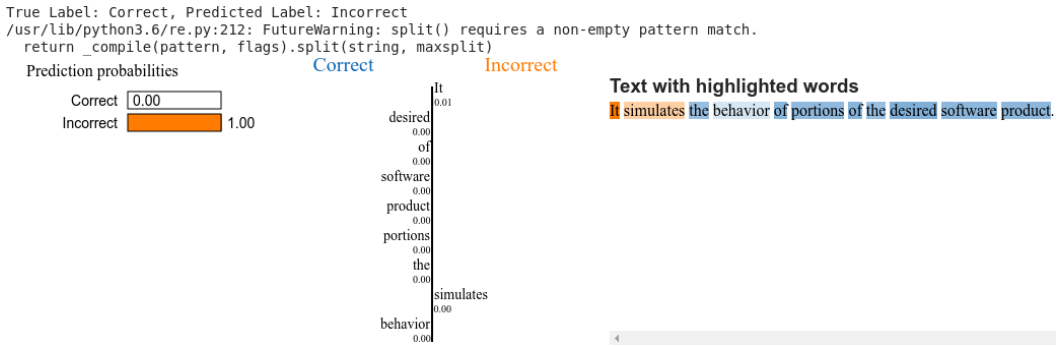

Figure 3.3.8: A test instance with a prediction as 'Incorrect' but without enough features highlighted in the text to support this prediction.

A low validation accuracy was consistently observed with the CNN model for all the questions in each category and the resulting highlighted keywords in the

generated explanations also deviated from human expectations.In contrast, using a logistic regression model shows a much better result by achieving an accuracy of 81.8% when tested on the same sample 1.1 used in experiment 1. While the accuracy is better than the CNN model, it is lower compared to the Naive Bayes classifier and hence shows a comparatively higher number of incorrect predictions. However the LIME generated explanations themselves were observed to have no major differences when compared with the results of experiment 1. Table 3.4 shows a comparison of the validation accuracy achieved by each of our models for every question in each category.

Q.No.	**Validation Accuracy**		
	Naive Bayes	CNN	Logistic regression
1.1.	90.90%	66.67%	81.81%
1.5.	90.90%	66.67%	81.81%
2.7.	36.36%	50.00%	27.27%
4.5.	54.54%	66.67%	63.63%
2.1.	81.81%	33.33%	90.90%
2.5.	72.72%	16.67%	81.81%
7.4.	72.72%	50.00%	81.81%
11.1.	63.63%	66.67%	63.63%
12.10.	81.81%	83.33%	81.81%
3.3.	100%	66.67%	100%
12.4.	54.54%	66.67%	63.63%
12.8.	72.72%	83.33%	81.81%
12.9.	81.81%	16.67%	63.63%

Table 3.4: Validation accuracies of different models.

As seen from Table 3.4, in majority of the cases, Naive Bayes gives comparable or better results than either of the other two models used in the experiment. We

thus use Naive Bayes classifier in our further experiments, as well as in our proposed solution approach which is discussed in the following chapter.

3.3.3 Experiment 3 - Comparison of SHAP and LIME's interpretability

We use the observations for the sample question 1.1 to highlight the main points from the observations of the visualization obtained from SHAP. Similar to results of experiment 1, we get a model accuracy of 90% on using a combination of TFIDFVectorizer and Naive Bayes classifier for the sample 1.1 in the first category of questions. We obtain a global interpretation for each class (i.e, the 'Correct' and 'Incorrect' class) by making a summary plot as shown in Figure 3.3.9. This figure gives an overview of the top 20 most important features in the model for question 1.1 by plotting the SHAP value for every feature for every sample.

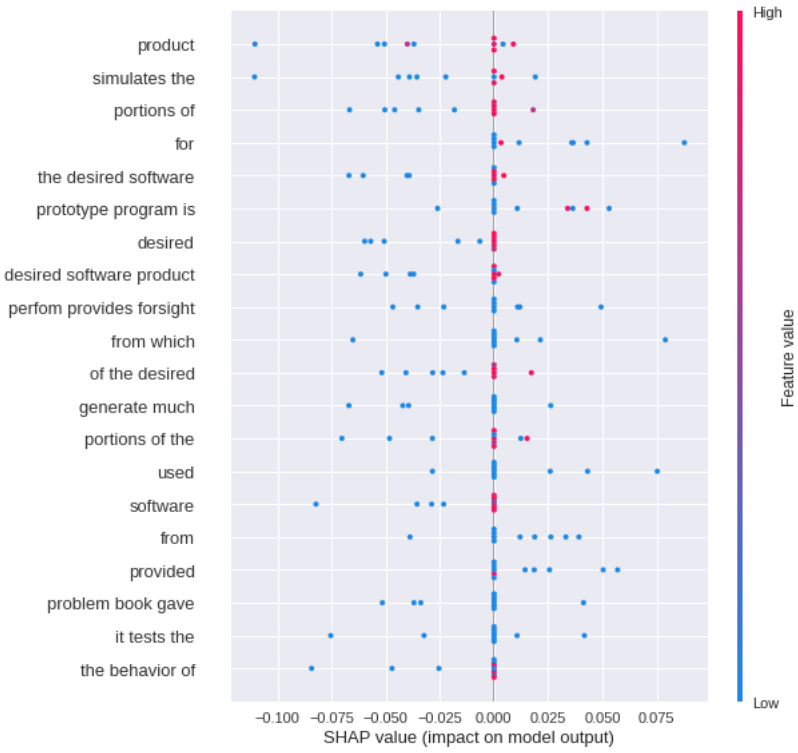

Figure 3.3.9: SHAP summary plot of top 20 features for the class "Correct".

The plot reveals that low values (i.e, the samples highlighted in blue) of features such as "*simulates the*", "*portions of*", "*desired*" and "*software*" for samples (low here indicating the absence of said feature) will have negative influence on model output. This indicates that these features are important for a student answer to be classified as "Correct". Presence of these features in any sample (highlighted in red) will not influence the classification of the answer as "Correct" in a negative way. This perfectly matches our intuition that these features are the most important predictors towards this class and that this relationship has been learned by the model.

Now consider the following plot showing the influence of the features on the class 'Incorrect':

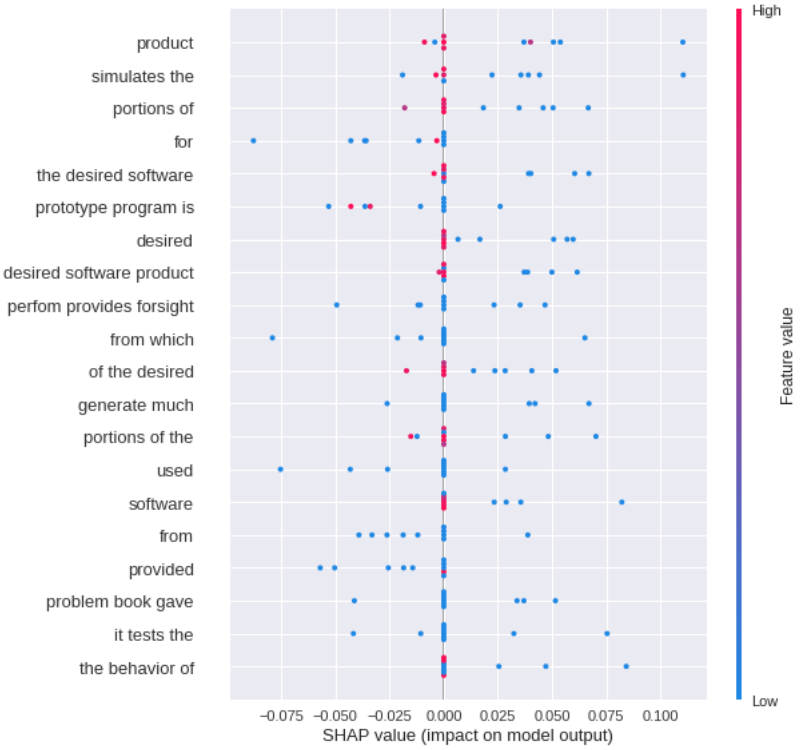

Figure 3.3.10: SHAP summary plot of top 20 features for the class "Incorrect".

It can be seen that this is a flipped version of Figure 3.3.9. Taking the feature "*simulates the*" as example, low value of this feature (indicated by the blue dots) has a positive impact on the classification towards this class. The values of the important features and their influence on the model output for the second class can thus be

deduced to be exact opposite of their influence on the first class.

3.3.3.1 Quantitative Benchmarking

As mentioned before there is no clear consensus on how to evaluate and measure interpretability. As such we make an attempt to assign a quantitative score to the interpretability achieved through LIME and SHAP by using an application level evaluation of the explanations. This evaluation is carried out by the experimenter and concrete procedure is explained in the following text.

The evaluation is carried out on the models built from the example questions mentioned in each category. When using LIME to get explanations for a student answer, we assign a score to the displayed explanation to indicate the degree to which we agree with the explanation. To accomplish this, we use a Likert scale that can take three possible values:

- **1** => indicates that the human user **disagrees** with the explanation indicating very low interpretability and trust.
- **2** => indicates that the human user **partially agrees** with the explanation.
- **3** => indicates that the human user **agrees** with the explanation indicating a high degree of interpretability.

A point to highlight here is the fact that some bias might inadvertently occur in the data since only one user, i.e the experimenter is recording the data. The measure of interpretability given by a different user might show significant differences in the values. Thus a further quantitative analysis through a user survey needs to be carried out to verify the comparison performed here.
The data captured in the manner described above constitutes a set of ordinal data that is qualitative in nature and thus cannot be manipulated through operations as we do for numeric (i.e, discrete or continuous) data. However, in order to have a final score that can be used to quantitatively compare LIME and SHAP's interpretation levels, we came up with a simplified process as described below:

1. From the data recorded for each question, we compute the ratio of the occurrences for each type of ordinal score.

2. Using SciPy library's interpolation function we perform the following mappings:
    - Ratios computed for ordinal value **1** is mapped from $[0, 1]$ => $[0, 0.33]$ range.
    - Ratios computed for ordinal value **2** is mapped from $[0, 1]$ => $[0.34, 0.66]$ range.
    - Ratios computed for ordinal value **3** is mapped from $[0, 1]$ => $[0.67, 1]$ range.
3. The mapping obtained for the ordinal value with the largest ratio in the data for a given question is concluded to be the *Interpretation Level (IL)* for that question.

In case we have a situation where more than one type of ordinal score has the same maximum ratio of occurrences (for e.g ordinal scores 1 and 2 both having a ratio 6 out of a total of 15 samples), in such a case we simply take the average of the these ordinal scores, and the average value that lies between 1 and 3 is mapped to [0,1] range directly. The above procedure is suitable for LIME where local instances can be quickly explained and is in a form easily understood by non-human experts. In case of SHAP where global explanations from summary plots are a more suitable choice, we follow a more simplified process of determining the interpretation level. Since the plots highlight the top globally important features necessary for every test sample for a question as well as their impacts, we directly score the explanation based on how accurately we feel the features and their impacts have been captured in the visualization. In case of SHAP, we found that a 3-point Likert scale was insufficient to give a proper measure of the interpretability, and thus used a wider 5-point scale. This ordinal data was directly mapped to the $[0, 1]$ range which corresponded to the *Interpretation Level (IL)* for SHAP for that question.

Figure 3.3.11, Figure 3.3.12 and Figure 3.3.13 respectively show the distribution of scores for each sample answer for questions from category 1, category 2 and category 3 when using LIME for generating explanations.

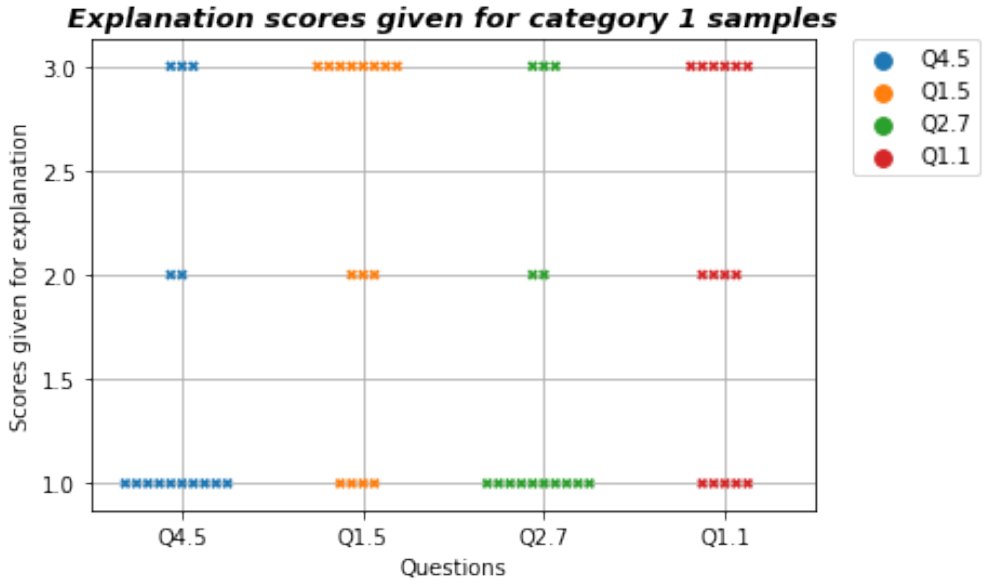

Figure 3.3.11: Plot of distribution of scores given for test samples for questions from category 1.

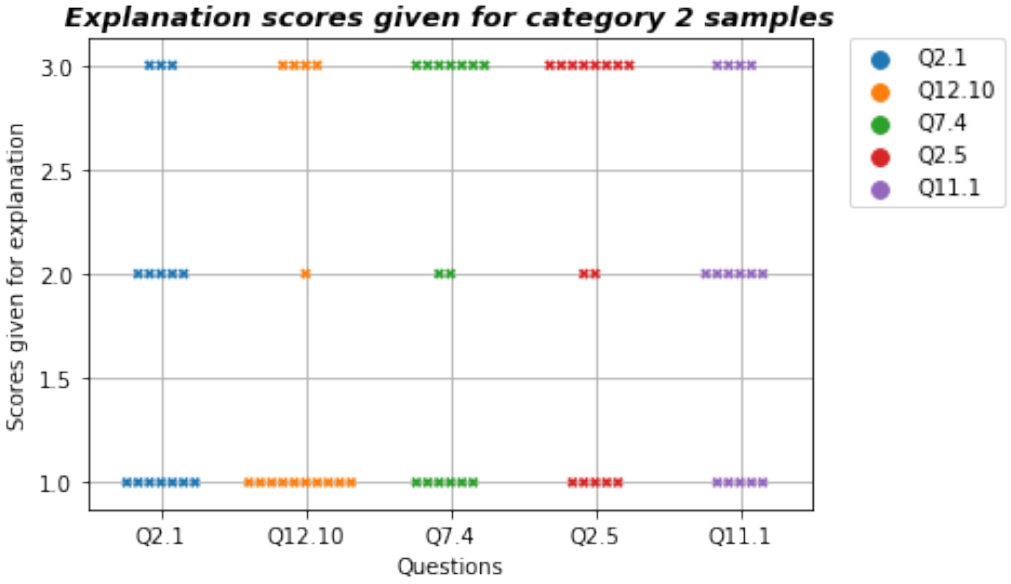

Figure 3.3.12: Plot of distribution of scores given for test samples for questions from category 2.

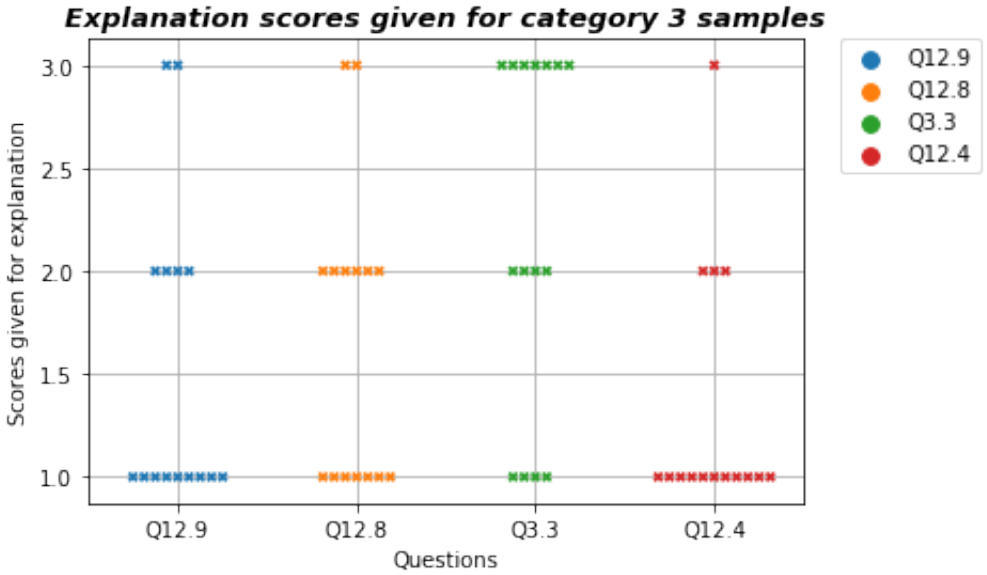

Figure 3.3.13: Plot of distribution of scores given for test samples for questions from category 3.

Figure 3.3.14, Figure 3.3.15 and Figure 3.3.16 show the *Interpretation Level (IL)* value of LIME and SHAP for each question under categories 1, 2 and 3 respectively.

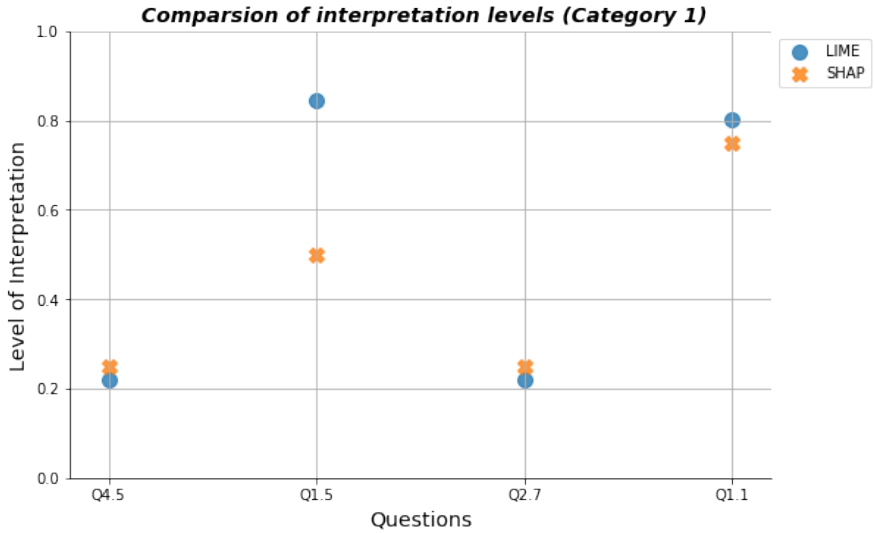

Figure 3.3.14: Quantitative Interpretation Level (IL) value comparison of LIME and SHAP for category 1.

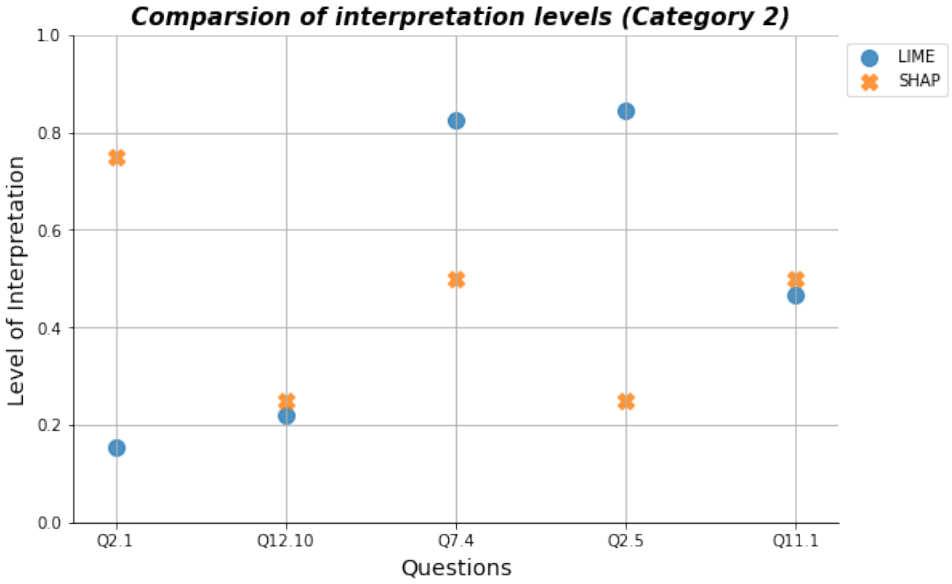

Figure 3.3.15: Quantitative Interpretation Level (IL) value comparison of LIME and SHAP for category 2.

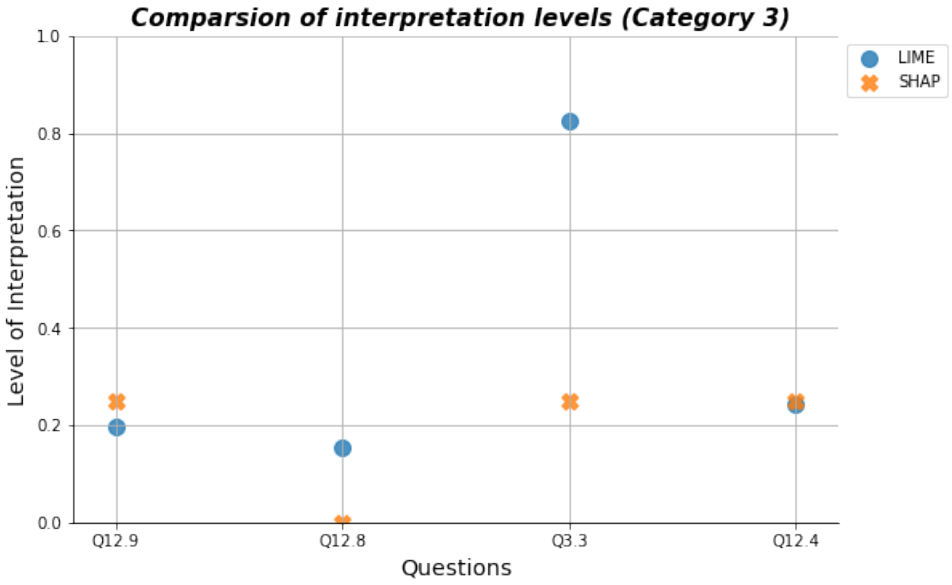

Figure 3.3.16: Quantitative Interpretation Level (IL) value comparison of LIME and SHAP for category 3.

3.4 Discussion

The main observations from the experiments in this chapter are summarized below:

- The validation accuracy was found to heavily influence the finally explanation. The higher the accuracy of the predictions, the closer the explanations were to a human's intuition.
- The explanations and interpretations obtained cannot be said to be notably better for any specific category of questions. From the experiments, we find that a categorization based on answer length is much more suitable in narrowing down the scenarios where LIME and SHAP are most suitable.
- Based on the observations of the explanations from LIME, we found that the most accurate explanations that matches human expectations is obtained from LIME (irrespective of the category to which the questions belong) when student answers satisfy two criteria:
  - The answers do not exceed one sentence in length.
  - The diversity in the unseen student answers are minimal.
- The features and feature attributions determined by SHAP is similarly affected by the increase in the linguistic diversity of student answers. In such cases redundant features were often judged to be among the top features affecting a decision for all samples, and the impacts on model outputs shown by the summary plots often deviated from human expectations.
- With no ground truth available to compare different interpretations techniques, an initial qualitative assessment followed by a strategy to map them to a value within a fixed range had to be developed to get an idea of how the two interpretation techniques measure against each other for the different cases studies. However it was indisputable that changing the scale used for the initial collection of the qualitative data, or changing the user used to record this data can easily influence the final quantitative measurement values.

- Global interpretation such as the one described in the third experiment is useful in understanding and getting a sense of the magnitude and direction of influence of each feature on the prediction. SHAP's strength lies in providing a globally consistent and accurate explanations as it involves an exhaustive approach by considering predictions for all possible coalitions for each feature. Compared to LIME which is fast and where explanations are obtained instantaneously, computing the Shapley values for the features takes time in the order of 5-10 minutes (observed in this experiment). On the other hand, while LIME does not have the accuracy and consistency properties of Shapley values, LIMEs output provides a more "human friendly" explanation, and is the preferable choice especially when exposing the explanations to lay-persons.

4

Solution Design

4.1 Proposed Algorithm

The algorithm developed in this work was aimed with end goal of predicting and presenting student textual responses such that a human end user's understanding of the relation between the components of the current instance and the model's prediction of the score is enhanced in a way similar to the explanations given in LIME, but at the same time also attempting to address some of the limitations of LIME.

One of the important issues that we address in this algorithm is the highlighting of text beyond unigrams as part of the explanation. Even if our model uses ngrams, LIME is restricted in its explanations by indexing and highlighting unigrams. However as observed from our experiments, explanations of this level is not sufficient enough to promote understanding of the human users given the diversity of student answers. Thus including words, phrases of a set length is included in the explanation generated by our algorithm.

The second main goal that we aimed to achieve was to identify globally important features that can also be used to explain the local instance being predicted. While former research as well as the results of our prior experiments has shown that establishing globally faithful explanations that also maintains its importance in the local context is still a challenge[21], we attempt to strike a balance between the two by actively involving the human end users in the grading process. The strategy

employed by the algorithm is mentioned in the following text:

- Starting from an unlabeled dataset, a small subset of instances are selected, and are then annotated by a human oracle. Unlike in active learning strategies where the selection of data points to be annotated is done based on the predictive uncertainty of a trained model, our selection is completely random and the annotation proceeds until the initially selected subset is exhausted.

- The labeled training set is used to learn our model (using the Naive Bayes classifier) and the ngram range of (1,3) is used to learn features for each class of decision (i.e, grades).

- Features learned from the trained model are used as baseline hints to generate explanations for the unseen data. This is done in the following way:

    1. Ngrams in the range of (1,3) is first extracted and compared against the global vocabulary of features for each class. Matching features found are set to be included as part of the explanation.

    2. Trigram phrases are given the highest priority to be highlighted in the explanation followed by bigrams. In other words, if there are features present in the form of trigrams, any possible duplications in the form of bigram and/or unigrams is eliminated. The same process is also applied for bigram features.

    3. Any trigram or bigram phrases in the unseen data that begin with the same words as the features that are a part of the global vocabulary are suggested as belonging to the same class of vocabulary and are included as part of the final explanation. In case a user disagrees with the suggestion, the feedback section of the algorithm can be used to correct and update the global vocabulary accordingly.

- Features recognized as part of an explanation are highlighted and the human oracle is allowed to suggest changes in the score as well as the explanation.

- Features suggested by the human oracle is used to update the current vocabulary of explanation features, and the change is highlighted to the human user.

- The new instance and its label is used to update the model and recompute the global feature sets for each class of grades. The global vocabulary is then updated with pool of feedbacks received upto the present instance.

Algorithm 1 Generate explanation with active feedback

Input: unlabeled dataset $\mathcal{D}$

1: Select n random samples $\mathcal{X} \leftarrow \{x_1, x_2...x_n\}$ from $\mathcal{D}$ as *training* and the rest as *test*.
2: **for** Each x_i in $\mathcal{X}$ **do**
3: Obtain annotation y_i for x_i from human oracle.
4: **end for**
5: Preprocess the data and train model ϕ on annotated dataset.
6: Obtain feature sets $\mathcal{F}^{\mathcal{C}}$ and $\mathcal{F}^{\mathcal{I}}$ corresponding to each class where each feature f_i is an ngram in range (1,3).
7: **for** Each instance x_i in *test* **do**
8: Get model's decision on the grade.
9: Derive ngram features in range (1,3) from instance x_i into $\mathcal{N}$.
10: Obtain explanations $\mathcal{E}^{\mathcal{C}}$ and $\mathcal{E}^{\mathcal{I}}$ such that a feature $e_i \in \mathcal{E}^j$ iff $e_i \in \mathcal{F}^j$ for the set of classes $j = \{C, I\}$.
11: **for** Each n_i in $\mathcal{N}$ **do**
12: $\mathcal{E}^j \leftarrow n_i \cup \mathcal{E}^j$ if n_i begins with the same word as one of the f_k in $\mathcal{F}^j$ where $j = \{C, I\}$.
13: **end for**
14: **for** Each explanation set $\mathcal{E}^j$ **do**
15: **for** Each feature e_i in $\mathcal{E}^j$ **do**
16: **if** Feature e_i in $\mathcal{E}^j$ also occurs as part of a larger ngram in $\mathcal{E}^j$ **then**
17: Remove ngram e_i from $\mathcal{E}^j$.
18: **end if**
19: **end for**
20: **end for**
21: Highlight model's decision and explanation.

Algorithm 2 Generate explanation with active feedback (continued)

Get user feedback on correctness of model prediction and update model's decision for current instance.
Obtain user feedback $\mathcal{V}^{\mathcal{C}}$ and $\mathcal{V}^{\mathcal{I}}$ on correctness of highlighted explanations.
Update annotated dataset with $\mathcal{X} \leftarrow \mathcal{X} \cup x_i$ and using corrected decision.
$\mathcal{E}^{\mathcal{C}} \leftarrow \mathcal{E}^{\mathcal{C}} \cap \mathcal{V}^{\mathcal{C}}$ and $\mathcal{E}^{\mathcal{I}} \leftarrow \mathcal{E}^{\mathcal{I}} \cap \mathcal{V}^{\mathcal{I}}$.
Display corrected explanation and update model ϕ.
Recompute feature sets $\mathcal{F}^{\mathcal{C}}$ and $\mathcal{F}^{\mathcal{I}}$ followed by $\mathcal{F}^{\mathcal{C}} \leftarrow \mathcal{F}^{\mathcal{C}} \cap \mathcal{V}^{\mathcal{C}}$ and $\mathcal{F}^{\mathcal{I}} \leftarrow \mathcal{F}^{\mathcal{I}} \cap \mathcal{V}^{\mathcal{I}}$.
end for

It should be noted that the model learned is on a "per-question" basis,i.e, a new model is learned after all the student responses given for the current question is exhausted.
The above described algorithm is implemented and integrated with a graphical user interface which serves as a viable proof of concept of the theory discussed in this chapter.

4.2 Interface Design

The development of the graphical user interface (GUI) started with an aim of being an iterative design process where a version of the interface is developed, undergoes user testing, and the results are used to improve the current version. In this section, the software architecture and the concepts behind the design of the first version of the GUI is discussed. The designer's observations and evaluation of the first prototype GUI will be mentioned in the next chapter. which also sets the stage for the next version of GUI.

4.2.1 Software Architecture and Design

The architecture for the developed GUI is shown in Figure 4.2.1. The whole dataset is first converted into stored as comma separated value (CSV) files as this makes it easier to convert the data into a Pandas dataframe. Depending upon the specific question selected at the front end, the corresponding CSV file is read and converted into a Pandas dataframe. Tasks such as grading the initial answers, and

providing feedback for the later predictions are done through the front end. The backend is responsible for preprocessing and extracting the features, learning a model, predicting and generating explanations, updating model from received feedback and finally storing the final grades and explanations.

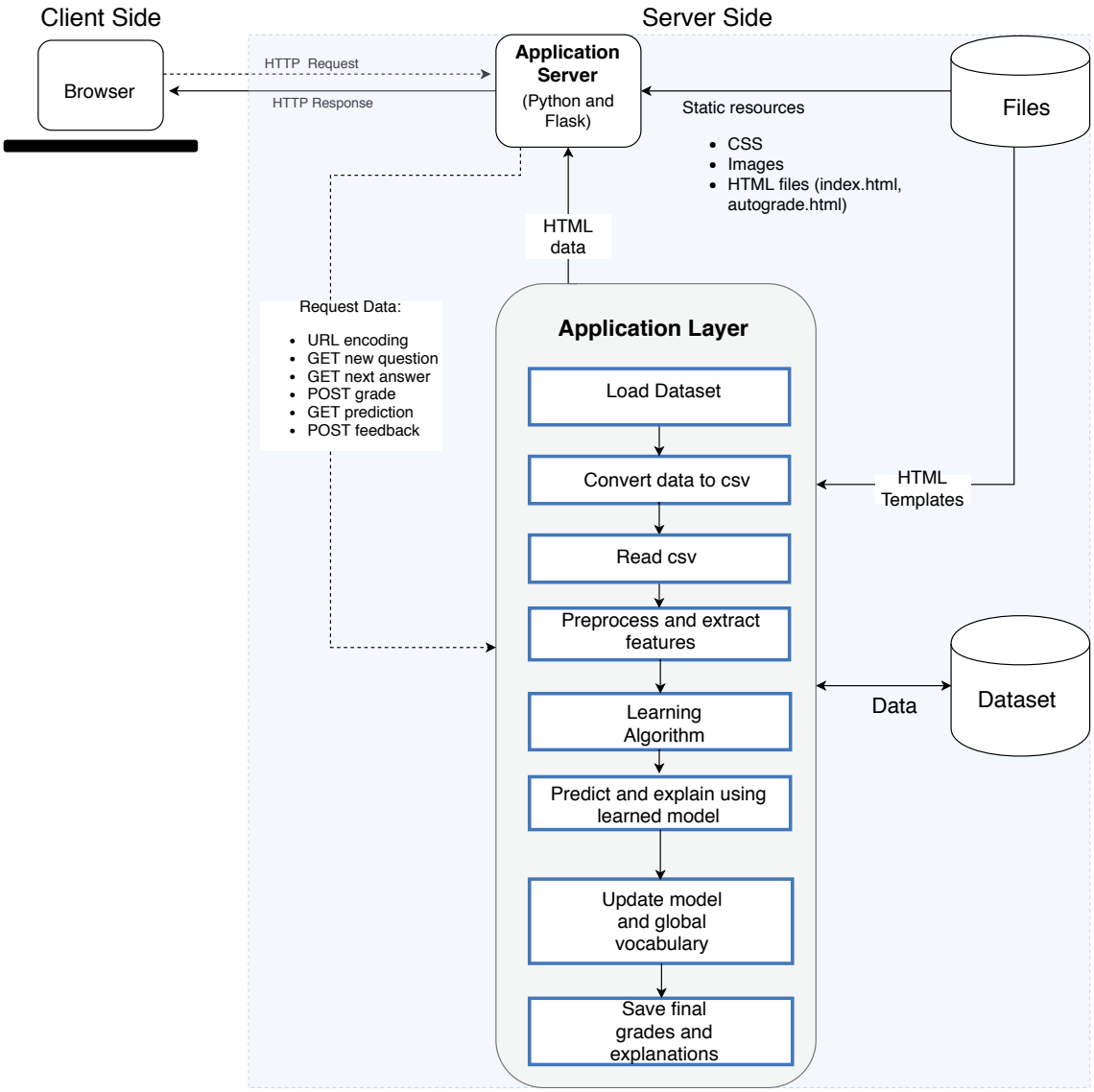

Figure 4.2.1: Software Architecture of first version GUI

With the algorithm mentioned in the previous chapter being implemented via python scripts, HTML5 and CSS was used to develop the frontend elements of the

grading assisting tool. The GUI was designed to be hosted and deployed as a web application. Thus, in order to have a python driven web application, the micro web framework Flask was used to route requests and responses between the client and the python web server. In addition, as a consequence of our pre investigations, in our application layer, we use TF-IDF as our feature extractor and the Naive Bayes classifier to learn our model.

The intention behind the development of the GUI was for it to serve as an assisting tool for human graders rather than a fully automated grading system. We will detail some of the requirements that influenced the features incorporated into the initial version of the tool:

1. Being a grading assisting tool, one of the primary requirements was the need for a feature to enable human raters to annotate a subset of the student answers so that the tool is able collect enough data to aid the human users in auto grading the subsequent set of student answers as well as provide supporting explanation for the system suggested grade.

2. A method to accept the human grader's feedback was the next main requirement integrated into the GUI in order to allow the tool to improve the model's accuracy on the subsequent grades from the updated knowledge pool.

3. The visual look of the explanations were developed following the ideas from LIME (i.e, assigning different colors to features belonging to different classes).

4. A reference to the model answer for the current answer being graded was added as a secondary requirement to assist the human users during the grading process.

With the goal of having a minimalist and easy to use design for the interface, the GUI frontend was set with one radio button for each possible grade. Since we treat the grading process as a binary classification task with only two possible grades for an answer, we thus include two radio buttons for 'Correct' and 'Incorrect' answers.

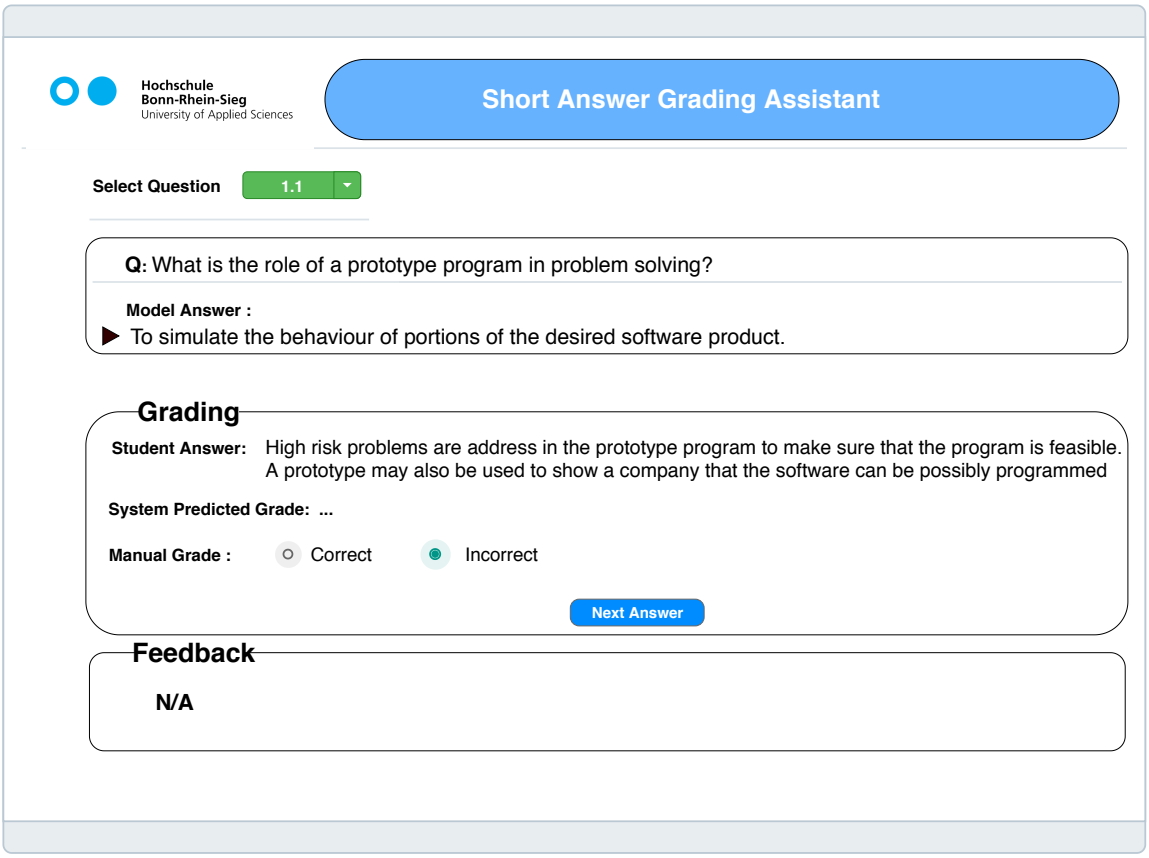

Figure 4.2.2: GUI design for the display during the initial grading stage.

Figure 4.2.2 shows the concept design for the interface as seen by the human users when they start grading an initial subset of the student answers. For each question, the model answer is displayed at the top for the user's reference. The feedback section during this stage is left empty and comes into effect once the system starts assigning the grades and explanations for the new answers.

After a fixed set of answers are graded by the users, the tool starts assigning grades to the remaining unseen answers. Figure 4.2.3 shows envisioned design for this stage where the text in the student answers is highlighted in two colors as part of the explanation to the human users. The highlighted phrases would include the necessary bigrams and trigrams as part of the explanation unlike LIME. Phrases highlighted in green indicate features contributing towards 'Correct' answers while red has been used for features contributing towards 'Incorrect' answers.

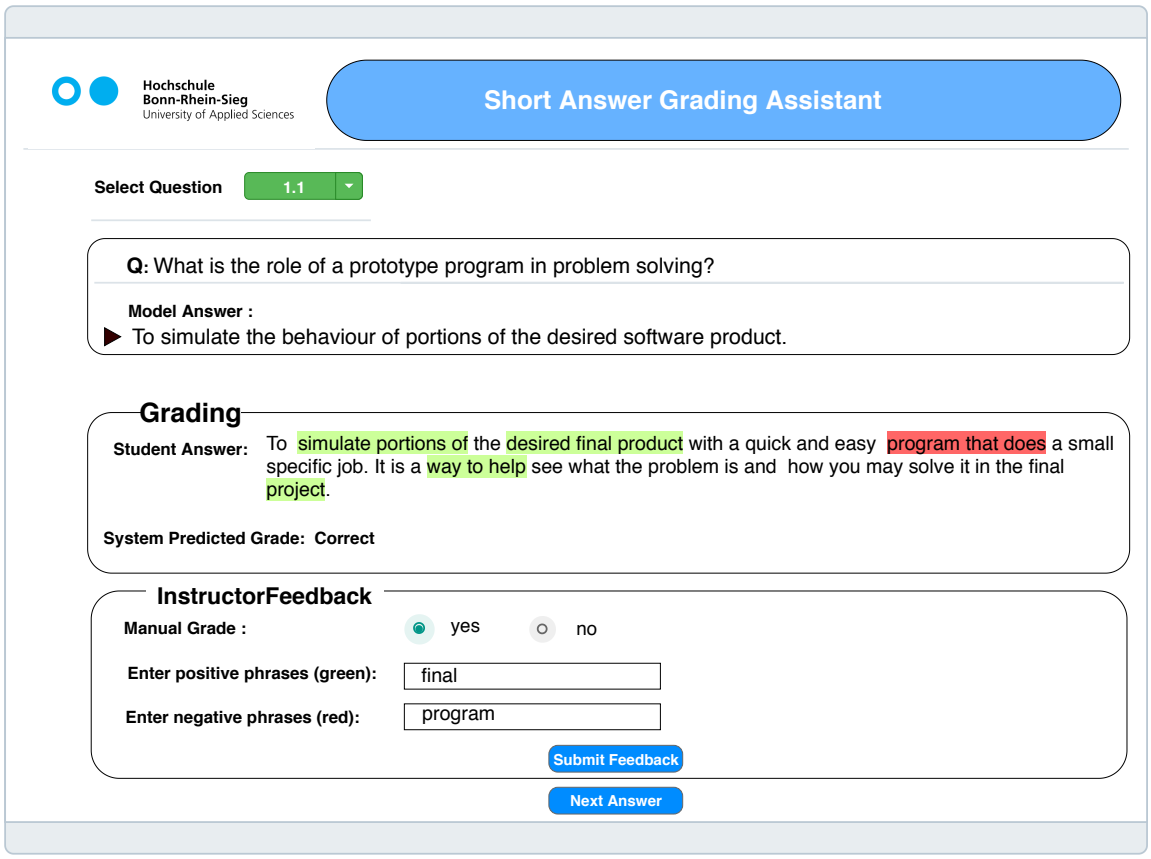

Figure 4.2.3: GUI design during autograding, explanation and feedback stage.

The feedback section now allows the users to indicate whether they agree with the system assigned grade, and if any changes should be made to the highlighted phrases. The feedback is optional and if included will allow the tool to improve the model accuracy as well as update its internal vocabulary of features for each category of grade. Figure 4.2.4 shows the change that should be seen once feedback is submitted.

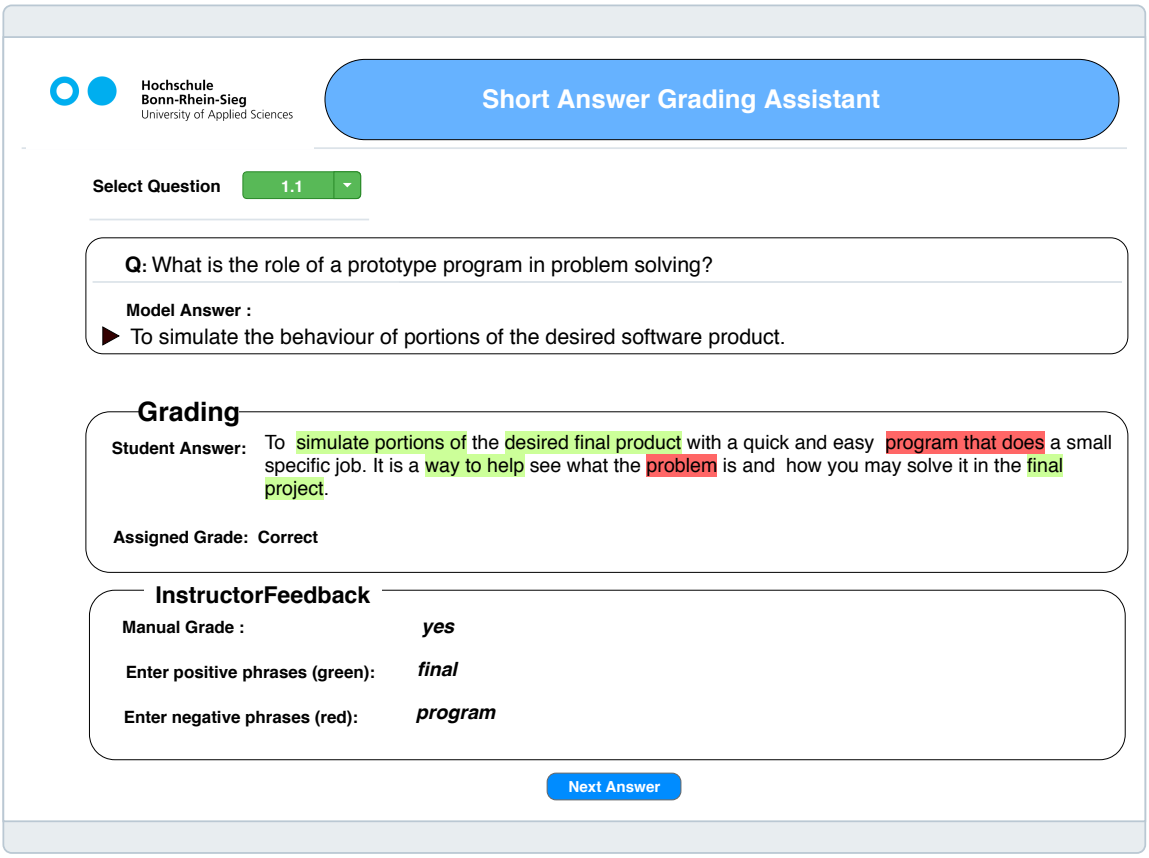

Figure 4.2.4: GUI design after feedback submission.

The results of the actual interface implemented will be discussed in the following chapter.

5

Results

This chapter starts with the example visualizations of the interface for a sample case study in order to show the working of the implemented tool. This is followed a quantitative evaluation of the tool in order to compare the developed software's interpretation level against LIME and SHAP and the experimenter's observations from the tests run using the interface. The quantitative assessment is performed by selecting the same questions as those used in the experiments described in Chapter 3. Finally the envisioned concept for the next prototype will be briefly described at the end of this chapter.

The answers for question 1.1 from Mohler's dataset is used as the case study to demonstrate the working of the GUI. Figure 5.0.1 shows the display seen during the start of grading of the initial answers by the grader, while the display seen when the tool starts to autograde the answers with highlighted explanations is illustrated in Figure 5.0.2. As can be seen from Figure 5.0.2, highlighted phrases now includes trigram phrases as part of the explanation as well unlike LIME.

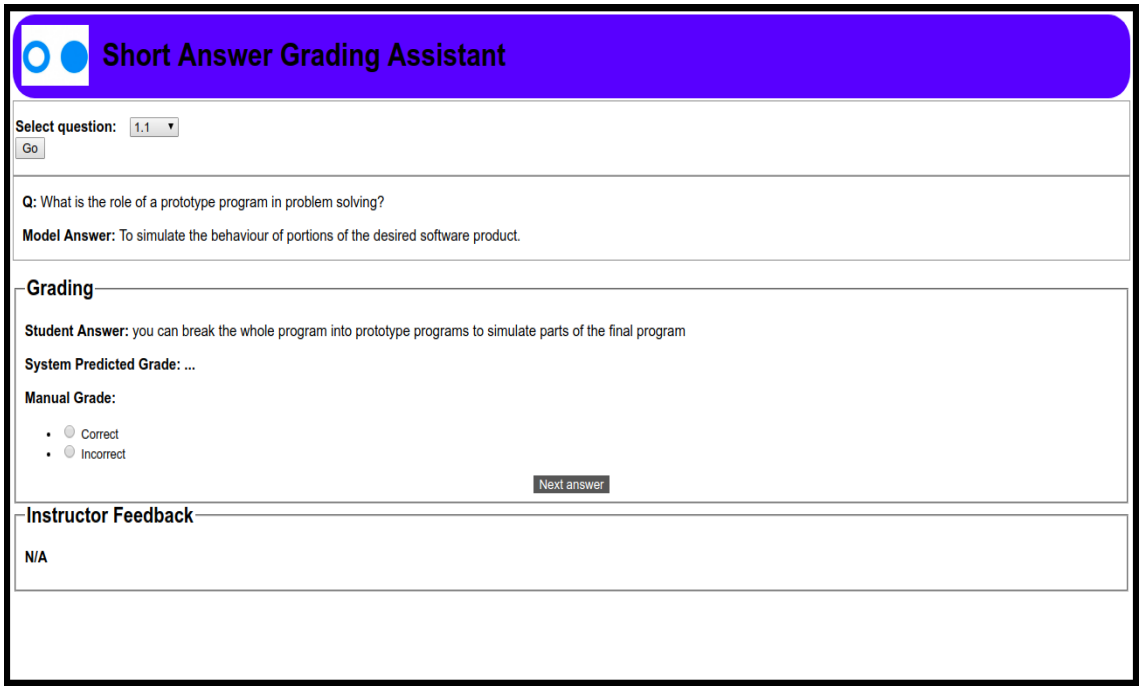

Figure 5.0.1: Visualization of the GUI during the annotation phase of the grading process.

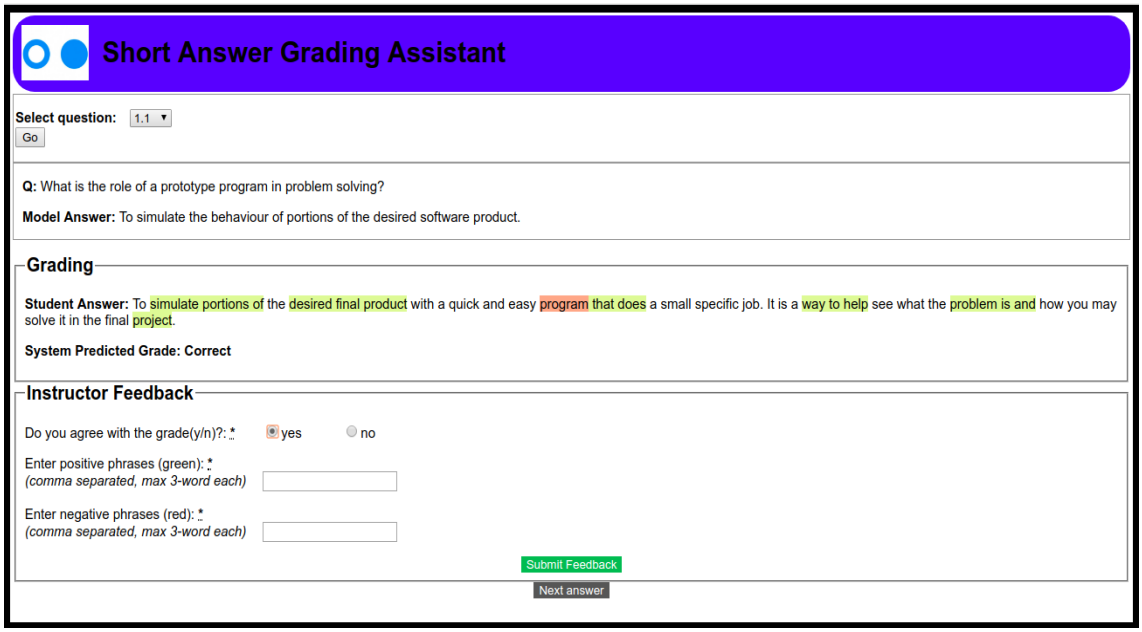

Figure 5.0.2: Visualization of the GUI during the autograding phase of the grading process.

The Figure 5.0.3 shows the UI display seen after a feedback is provided. The explanation highlighted in the student answer changes to reflect the provided feedback while the feedback section shows what inputs have been provided for the current answer.

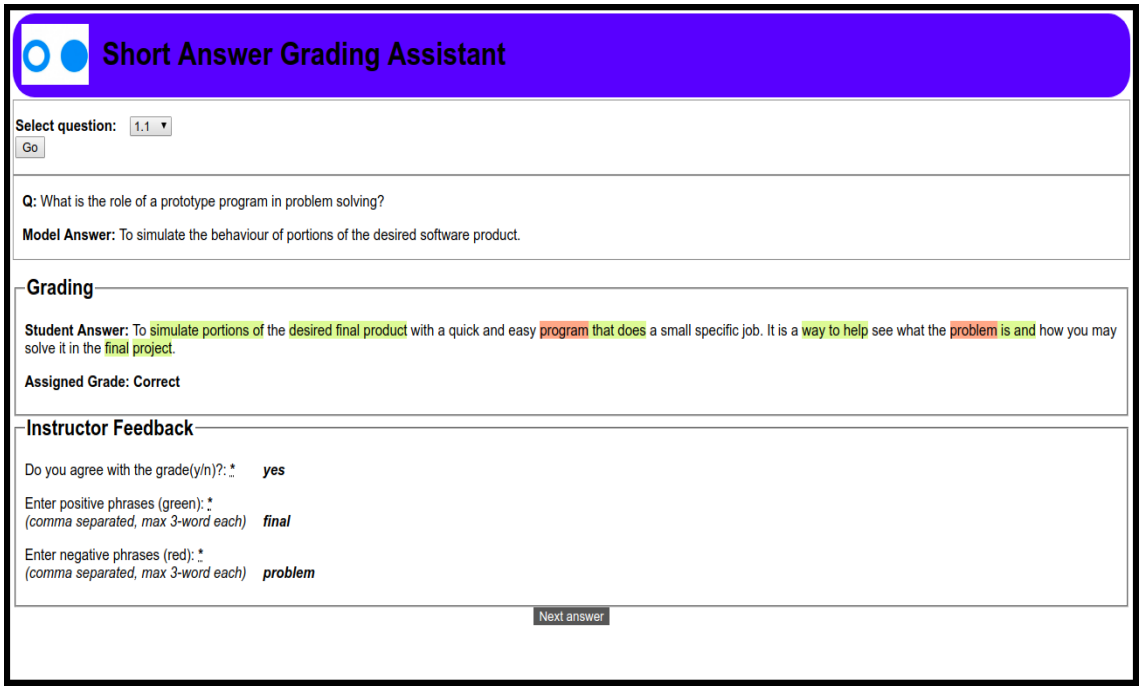

Figure 5.0.3: Visualization of the GUI showing the display following a submitted feedback.

The provided feedback updates the global vocabulary pool of features for each class of grades, and the following answers will now include and highlight these features as part of the explanation. Figure 5.0.4 shows an example new answer that marks the features specified in the feedback in Figure 5.0.3 as part of the explanation. The feature "*problem*" given in the earlier feedback for incorrect answers is now highlighted as part of the explanation for the "Incorrect" answer illustrated in Figure 5.0.4. Further, since we have an active learning process that updates its model as well as the global vocabulary from the new answers, features learned from this "Incorrect" answer will be used to explain the new answers better. Consider the prediction for the answer seen in Figure 5.0.5.

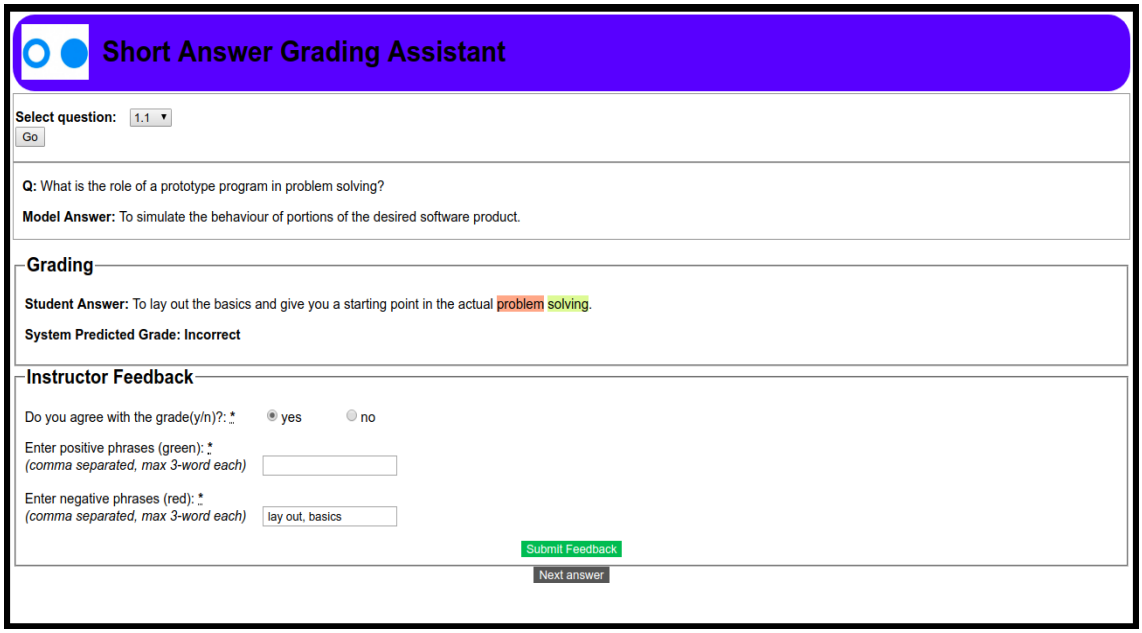

Figure 5.0.4: A new answer with the explanation containing features from a prior feedback.

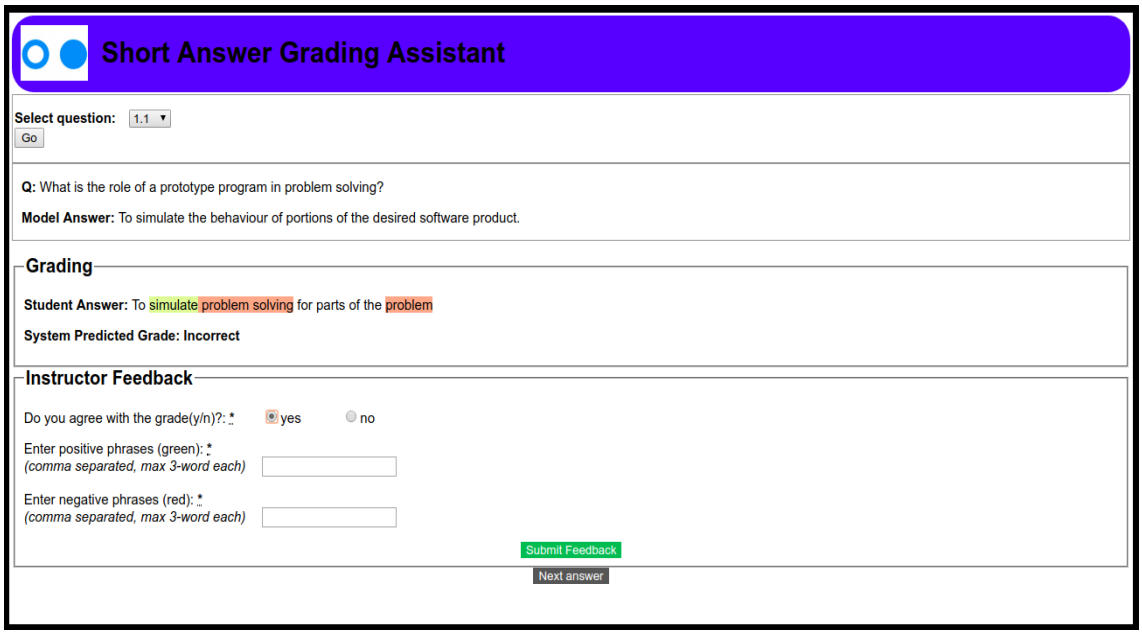

Figure 5.0.5: A new answer highlighting the new features learned from a prior answer.

The feature "*solving*" which was initially recognized as an explanation for "Correct"

answers (illustrated by the green highlight in Figure 5.0.4), was later understood to be an explanation for the "Incorrect" answers after the model was updated with the new answer. The bigram "*problem solving*" learned from this answer is used to explain the "Incorrect" answer seen in Figure 5.0.4. The explanations was thus observed to markedly adapt and improve with each new answer graded.

5.1 Evaluation

The quantitative evaluation is done by repeating the procedure carried out when evaluating the explanation from LIME, i.e, using a Likert scale of 3 values (1,2 and 3) to score the degree of the acceptance of the displayed explanation for every answer for all questions specified in each category mentioned in Chapter 3, and then SciPy library's interpolation function to map the maximally occurring score for a question into the ranges [0,0.33], [0.34,0.66] and [0.67,1] (for 1, 2 and 3 respectively). Figure 5.1.1, Figure 5.1.2 and Figure 5.1.3 respectively show the distribution of scores for each sample answer for questions from category 1, category 2 and category 3 when using our proposed solution for generating explanations.

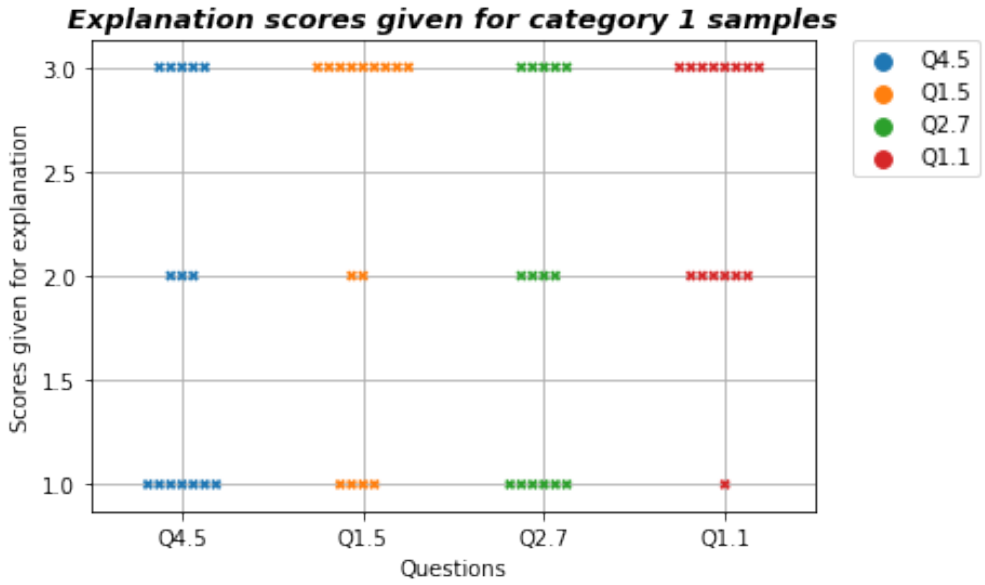

Figure 5.1.1: Distribution of scores for explanations when using the proposed solution for questions from category 1.

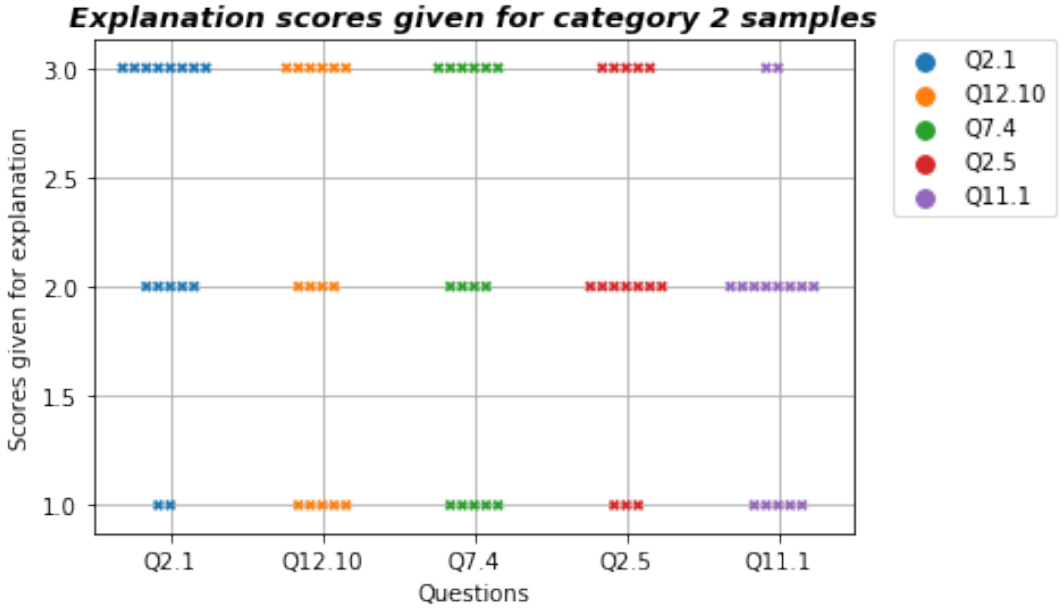

Figure 5.1.2: Distribution of scores for explanations when using the proposed solution for questions from category 2.

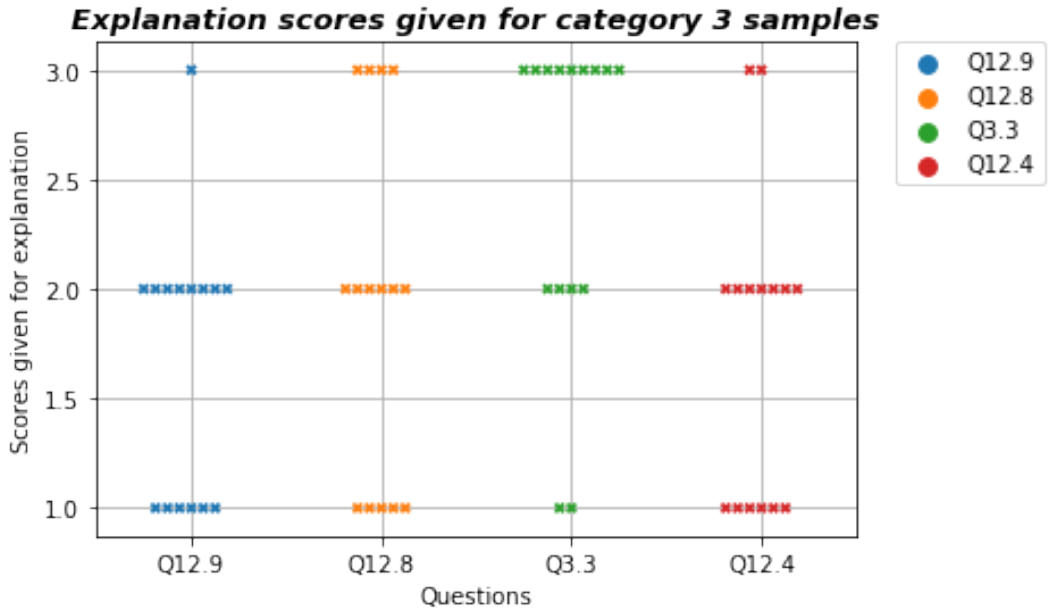

Figure 5.1.3: Distribution of scores for explanations when using the proposed solution for questions from category 3.

Figure 5.1.4, Figure 5.1.5 and Figure 5.1.6 show the *Interpretation Level (IL)* value of LIME, SHAP and the proposed solution for each question under categories 1, 2 and 3 respectively.

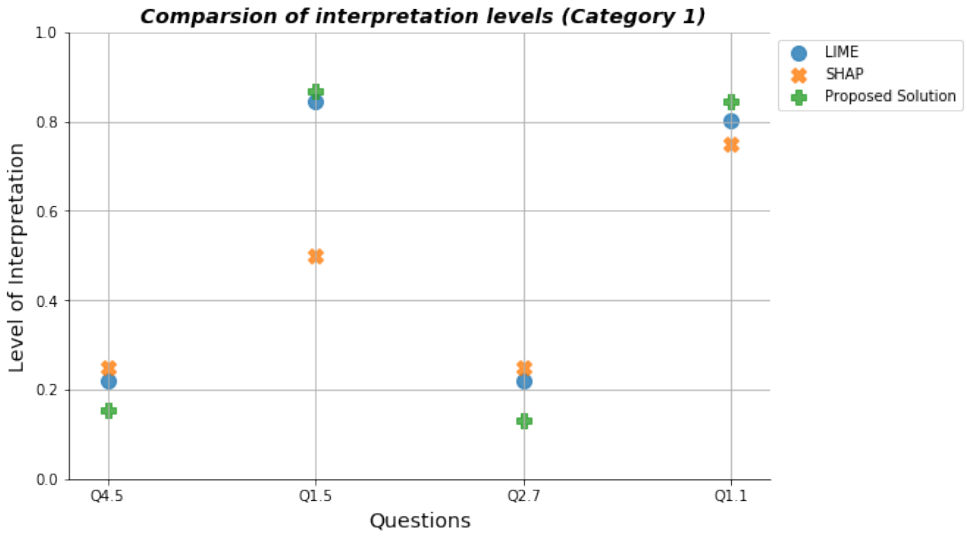

Figure 5.1.4: Quantitative Interpretation Level (IL) value comparison of LIME and SHAP for category 1.

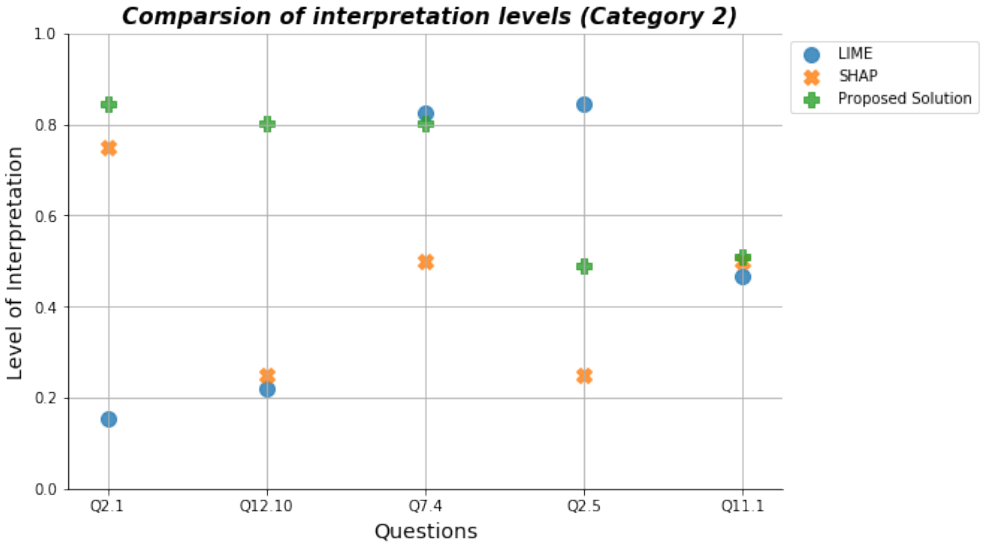

Figure 5.1.5: Quantitative Interpretation Level (IL) value comparison of LIME, SHAP and the proposed solution for category 2.

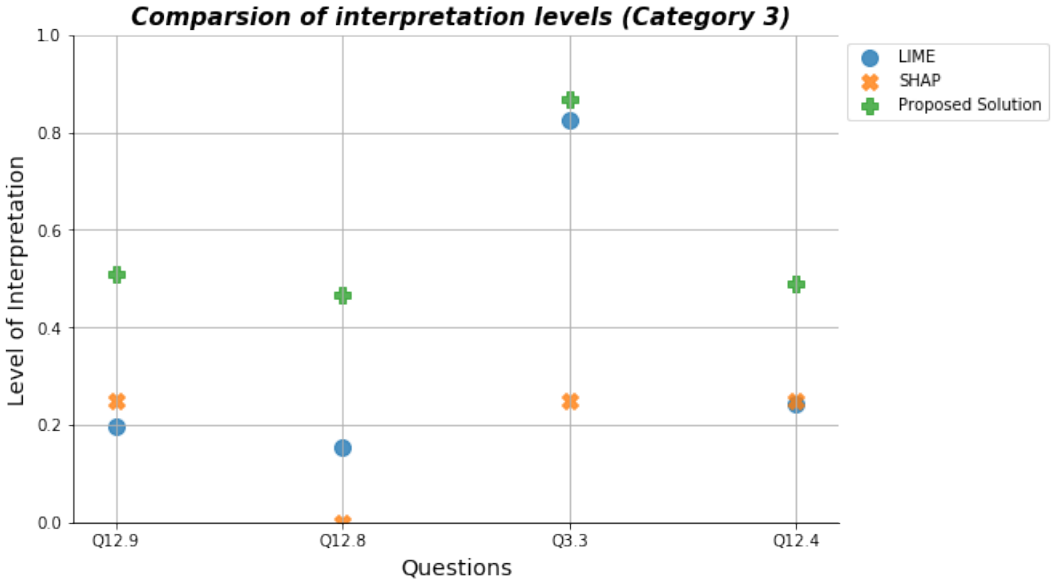

Figure 5.1.6: Quantitative Interpretation Level (IL) value comparison of LIME, SHAP and the proposed solution for category 3.

We see that the level of interpretation in all cases is at least as good as that obtained from LIME and SHAP or better in most cases when using the proposed solution. A few important details were noted down during the tests run using the GUI that is elaborated in the following section.

5.2 Observations

The first prototype GUI was designed to work as a proof of concept and show the feasibility of the algorithm described in the previous chapter. Tests run using the first prototype revealed the insights on aspects that require special attention when designing the next version of the interface. These are mentioned below:

- When providing feedback on the explanations, providing input in the form of comma separated phrases through text boxes was found to be a tedious and time consuming task. Thus an alternative means of providing input to the system was desirable.

- A conflict was observed in cases when handling certain features that were common in both classes. For instance when a unigram was present as an

explanation in one class but also occurred as part of a trigram in the second class, explanations usually would overlap, i.e, the feature would be highlighted in both colors.

- While the first GUI version was implemented with the intention of having a minimalist design, it was seen that repeating the process of reading the system's prediction and explanation and then providing feedback for each student answer expended too much time and effort.

5.3 Interface Version 2.0

The observations on the usability of the first prototype mentioned above propelled the interface evolution towards the next version. An envisioned concept design of the next version is shown in Figure 5.3.1.

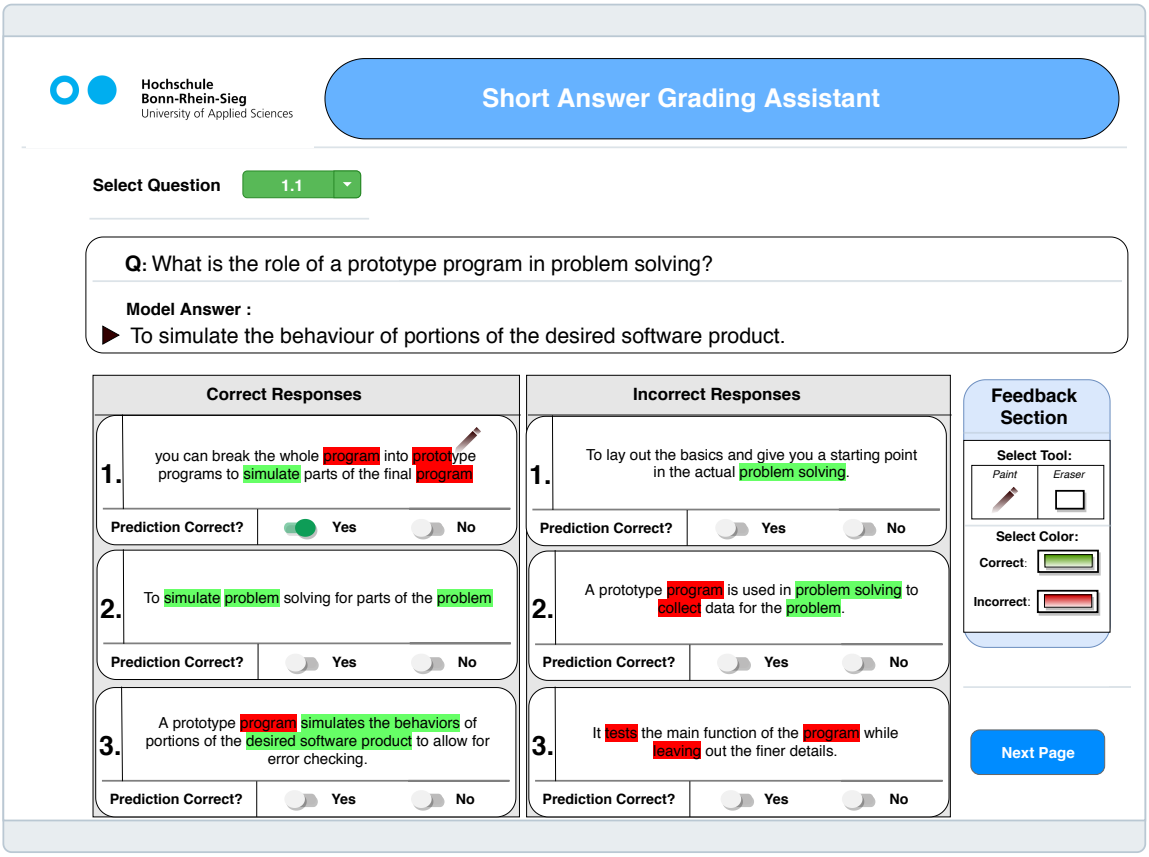

Figure 5.3.1: Concept design of interface version 2.0.

As seen in the envisioned design, the next version will have student responses clustered according to each class of grades in one display thereby ensuring a significant increase in the speed of the grading process. Textual input form of feedback will be removed and replaced with painting and erasing tool to edit the explanations highlighted in the interface.

6

User Evaluation and Survey

This chapter gives an outline of the planned user evaluation strategy to be carried out for verifying the proof of concept described in the previous chapter. Since the quantitative evaluation and comparison carried out previously solely depended upon the experimenter's subjective judgment of the given explanations, this evaluation aims at assessing the quality of the developed system through interacting with real users and thereby determining the requirements and limitations that were previously missed.

Since the development of the autograding assisting system is geared towards use in examinations and in-house assignments, the user evaluation will be carried out at Hochschule Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin. For the purpose of the evaluation, a class of 15-20 undergraduate students from the department of computer science ranging in age between 20 and 30 will be enrolled and a grading scenario will be simulated. The selected participants would have no prior experience in using the prototype grading system. The concrete steps in this evaluation process is as follows:

- An initial ten-minute introduction will be given to every participant to enable them to acquire a basic knowledge of the functionalities of the prototype GUI.
- Every participant is provided a desktop system with Ubuntu OS that has the autograding system pre-installed in it.

- Each subject will be asked to grade one set of answers for a question using the autograding tool. The question that each subject grades will be different. This is done to ensure that there are sufficient variations to better capture the limitations of the tool.

Following the end of the grading session, a questionnaire form will be administered to every participant. The questions in the form will consist of questions designed to capture degree of acceptance of the system's features as well as questions that properly determine a user's expectations from a similar system. Thus the included questions will consist of those that require the participants to provide a score on a tentative 7 point Likert scale (a wider scale compared to our prior experiments in order to gain a better understanding of the level of interpretation), as well as questions requiring the participants to provide descriptive statements on their difficulties as well as what they hope to see in the system.

An exploratory data analysis will be carried out on the acquired data in order to gain a statistical summary of the main characteristics of the acquired data. The summary statistics will be used to acquire as many patterns and insights from the data as possible which sets the direction for the next experiments and data collection.

7

Conclusions

7.1 Contributions

The contributions from this research are highlighted below:

- Two different model interpretability techniques (LIME and SHAP) were studied, compared and evaluated from the context of their applicability towards short answer grading by relying on two different datasets. This included the study of the influence of varying the feature extractors and learned models on the final explanations (more specifically the influence on the explanation from LIME). Furthermore, the two interpretability techniques were quantitatively assessed and compared to show the level of interpretability achieved with both for different case studies.

- An algorithm to generate explanations that is capable of addressing the requirement of highlighting ngram phrases in the range (1,3) was developed based on the insights gained from our pre investigations on LIME and SHAP. The solution developed is model agnostic and uses online feedback during autograding process to update its knowledge of vocabulary and thus improve the subsequent explanations. Being model-agnostic makes the proposed algorithm flexible in terms of the machine learning classifier used, and only small modifications are required to make the system work for multi-class classifications as well.

- A web application was developed to integrate the developed algorithm and provide a GUI that can assist human users in grading answers supported by explanations. While the GUI has been set to work with Mohler's dataset, only minor changes are required to allow it to be used with any dataset as long as the data is provided in CSV format.

7.2 Lessons learned

Several issues were encountered that had an impact on the direction of the progress in this research. The most significant of these are mentioned below:

- It was difficult to define what constituted as a "good" human friendly explanation especially from the context of explaining predictions of an autograding system for short answers. Furthermore, no single model interpretation technique existed that could act as a complete solution to our requirement.

- The solution proposed in this work is an approach that looks at only the syntactic properties of student answers. As a result, it was seen that the understanding gained from the explanations was limited in several cases where the student answers included words with more disambiguations. Properly taking the context of the words into account and having an approach that can exploit the semantics of a sentence was deemed to be necessary in order to evolve our solution and its resulting explanation to obtain more acceptable and "human-friendly" explanations.

- The lack of a concise evaluation methodology for measuring interpretability meant that we could not quantify the different interpretation methods in our experiments. We thus had to come up with a way to make a meaningful quantitative assessment from ordinal data to determine how the different methods compare against each other.

- Determining explanations that could achieve the property of being locally accurate while also maintaining the property being consistent with a human user's intuition was difficult even with a human in the loop as there were several conflicts arising due to features being present as part of a larger phrase or as part of both class of predictions.

7.3 Future work

During the course of this work, several domains remained unexplored due to the limits of the time involved in this research. Some areas that merits further investments are mentioned below:

- The user evaluation and survey mentioned in the previous chapter needs to be carried out a high priority in order to determine the functionalities required by real users. This combined with our suggestions previously mentioned for the version 2.0 of the GUI would allow us to simplify and optimize to the maximum extent.

- The current autograding tool is developed as the most basic standalone tool requiring the data to be separately converted into csv format before usage. This tool needs to be linked to students' answers given in Jupyter notebooks so that the process of conversion of the input data into the required format, followed by the grading, prediction of scores, and display of scores is more automated and ready to be deployed for usage in university environments.

- Latent Semantic Indexing can help determine relationships between words in latent space and is an unexplored avenue in the current work. This field can help us take a significant step in the direction of text comprehension if included as part of the algorithm and merits further research in this area.

References

[1] Mohd Juzaiddin Ab Aziz, Fatimah Dato'Ahmad, Abdul Azim Abdul Ghani, and Ramlan Mahmod. Automated marking system for short answer examination (ams-sae). In *Industrial Electronics & Applications, 2009. ISIEA 2009. IEEE Symposium on*, volume 1, pages 47–51. IEEE, 2009.

[2] Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. " what is relevant in a text document?": An interpretable machine learning approach. *PloS one*, 12(8):e0181142, 2017.

[3] Steven Burrows, Iryna Gurevych, and Benno Stein. The eras and trends of automatic short answer grading. *International Journal of Artificial Intelligence in Education*, 25(1):60–117, 2015.

[4] Laurie Cutrone, Maiga Chang, et al. Auto-assessor: Computerized assessment system for marking student. In *2011 IEEE International Conference on Technology for Education*, pages 81–88. IEEE, 2011.

[5] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[6] Wael H Gomaa and Aly A Fahmy. Short answer grading using string similarity and corpus-based similarity. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 3(11), 2012.

[7] Daniel Gutierrez. Layer-wise Relevance Propagation Means More Interpretable Deep Learning. *Open Data Science*, 2018. URL https://opendatascience.com/layer-wise-relevance-propagation-means-more-interpretable-deep-learning/.

[8] Debra T Haley, Pete Thomas, Anne De Roeck, and Marian Petre. Measuring improvement in latent semantic analysis-based marking systems: using a computer to mark questions about html. In *Proceedings of the ninth Australasian conference on Computing education-Volume 66*, pages 35–42. Australian Computer Society, Inc., 2007.

[9] Michael Heilman and Nitin Madnani. Ets: Domain adaptation and stacking for short answer scoring. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 275–279, 2013.

[10] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[11] Yanling Li and Yonghong Yan. New similarity measures for automatic short answer scoring in spontaneous non-native speech. 2012.

[12] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[13] Edward Ma. Interpreting your deep learning model by SHAP. https://towardsdatascience.com/interpreting-your-deep-learning-model-by-shap-e69be2b47893, 2018. [Online; accessed 01-Oct-2018].

[14] Ahmed Ezzat Magooda, Mohamed Zahran, Mohsen Rashwan, Hazem Raafat, and Magda Fayek. Vector based techniques for short answer grading. In *The Twenty-Ninth International Flairs Conference*, 2016.

[15] David Martens and Foster Provost. Explaining data-driven document classifications. 2013.

[16] Tim Miller. Explanation in artificial intelligence: insights from the social sciences. *arXiv preprint arXiv:1706.07269*, 2017.

[17] Michael Mohler, Razvan Bunescu, and Rada Mihalcea. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 752–762. Association for Computational Linguistics, 2011.

[18] Christoph Molnar. *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/, 2018. https://christophm.github.io/interpretable-ml-book/.

[19] SriSatish Ambati Patrick Hall, Wen Phan. Ideas on interpreting machine learning. https://www.oreilly.com/ideas/ideas-on-interpreting-machine-learning, 2017. [Online; accessed 01-Oct-2018].

[20] Stephen G Pulman and Jana Z Sukkarieh. Automatic short answer marking. In *Proceedings of the second workshop on Building Educational Applications Using NLP*, pages 9–16. Association for Computational Linguistics, 2005.

[21] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[22] Ando Saabas. Interpreting random forests. http://blog.datadive.net/interpreting-random-forests/, 2014. [Online; accessed 01-Sept-2018].

[23] P Selvi and AK Bnerjee. Automatic short-answer grading system (asags). *arXiv preprint arXiv:1011.1742*, 2010.

[24] Asterios Stergioudis. Model Interpretability with SHAP. *F1 Predictor*, 2018. URL http://www.f1-predictor.com/model-interpretability-with-shap/.

[25] Md Arafat Sultan, Cristobal Salazar, and Tamara Sumner. Fast and easy short answer grading with high accuracy. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1070–1075, 2016.

[26] Hao-Chuan Wang, Chun-Yen Chang, and Tsai-Yen Li. Assessing creative problem-solving with automated text grading. *Computers & Education*, 51(4): 1450–1466, 2008.

[27] Brendan Whitaker. Layer-wise relevance propagation and other neural network explanation methods. https://towardsdatascience.com/neural-network-explanation-methods-solutions-to-the-uber-incident-in-arizona-7660fc6759c2, 2018. [Online; accessed 01-Oct-2018].