Descriptive Answer Evaluation

Prayag Singh[1], Saurabh Sheorain[2], Shivam Tomar[3], Shubham Sharma[4], N.K. Bansode[5]

[1,2,3,4] Department of Computer Engineering, Army Institute of Technology, Pune, Maharashtra
[5]Professor, Department of Computer Engineering, Army Institute of Technology, Pune, Maharashtra.

---***---

Abstract – *With the growth in population, and necessity of education, it is proving hard for assessors to check the relevance and accuracy of the answers written by students. In this work we present a new approach of calculating the score of each answer (entered by student) based on the dataset on which the machine is trained. For every answer being entered it was important to reward points based on the usage of words and their importance. We evaluate our approach using Kaggle Short Answer dataset (ASAP-SAS, 2012). With this approach we can save the cost of checking the answers manually and can devote the same time for human welfare.*

***Key Words*:** Machine Learning, Gradient Boosting Machine, Random Forest, Boruta.

1. INTRODUCTION

In last few years there has been a significant growth in the number of solutions used for automatic grading of answers. Some of them included grammar checking, syntax and lexical breakdown of student's answer (Landauer et al., 2003). Answers are brief and evoke very specific responses from students. Regular expressions, text templates or patterns have been used to determine whether a student answer matches a specific word or a phrase present in the rubric text. Manually generated regular expressions have been used as features in generating models that score short answers in Kaggle Short Answer Scoring Competition (ASAP-SAS, 2012)[1]. Tandalla (2012)'s approach is best performing and achieved Quadratic Weighted (QW) Kappa of 0.70 using regular expressions as features [2][3][4]. But regular expression generation could be a time consuming process.

One of the main contributions of this paper is the use of an automated approach to generate patterns that can be used to grade short answers effectively, while spending less time an effort. Boruta is used for features selection [5]. Whereas Gradient Boosting Machine and Random Forest are used to further enhance the accuracy of the results [6][7]. While the implementation of not 100% accurate as it can't be used for small training dataset. If used there will result in less accuracy and will be similar to normal string matching. This paper aims to address the exact quantity of data required for high accuracy of scores. Each Set of question corresponds to at least 1600 (approx.) semantic different answers as training dataset.

Traditional grading systems are failing to meet the desired accuracy in scoring. The Grading techniques used are pushing the accuracy till a certain limit. Moreover, the issue with the traditional grading system is use of attributes. The attributes to be selected for scoring are invalid or irrelevant sometimes hence result in wrong scoring. The proposed model aims to provide proper relevant labels or attributes from the dataset which will enhance the precision of scoring. Also the proposed system provides User Interface (UI) for simple user interaction and understanding.

2. ANSWER GRADING SYSTEM

The project sets an automated technique to grade student answer based on the training dataset. In this we have an answer reviewed and scored by two human graders. The use of more than one grader will enhance the accuracy and answer checking performance. The answer grading system is divided into 5 stages as shown id Fig-1.

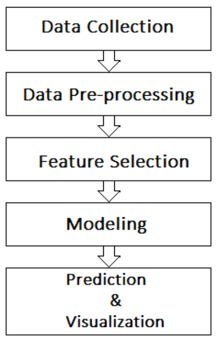

Fig -1: Flowchart of proposed work

2.1 Data Collection

Dataset will be responsible for converting answers into their corresponding score based on its relevance as per data. Kaggle provided a training data of approximately 17,000 answers with two scores graded by two different people [1]. While talking about number of questions, our training dataset has 10 questions and every question is graded based on the set of labels and features described important by the human grader.

Name	Type	Description
Id	Numeric	A unique identifier for each individual student answer.
EssaySet	Numeric	1-10, an id for each set of essays.
Score1	Numeric	The human rater's score for the answer. This is the final score for the answer and the score that you are trying to predict.

Score2	Numeric	A second human rater's score for the answer. This is provided as a measure of reliability, but had no bearing on the score the essay received.
EssayText	String	The ascii text of a student's response.

Table -1: Attributes in Training Dataset

They also provided two data sets as test data which consists of approximately 6,000 answers each, but these ones don't have their scores. Testing Dataset is similar to training dataset (as shown in Table-2).

Name	**Type**	**Description**
Id	Numeric	A unique identifier for each individual student answer.
EssaySet	Numeric	1-10, an id for each set of essays.
EssayText	String	The ascii text of a student's response.

Table -2: Attributes in Testing Dataset

2.2 Data Preprocessing

Dataset contain answers with non-alphabetic characters and need to be replaces by spaces. In case of Set-10 we will replace non-alphanumeric characters by space because Set-10 makes references to experimental numerical data and those numerical data may influence scoring. Later all capital letters are replaces by lower case letters.

The next step is for cleaning the data by correcting misspelled words. For this we have used predefined dictionary of words and also created some words as per requirement of the answers (marking them as special words).

The spelling corrector [2] checks if the word is the word is the list of 'right' words. If it is, the same word is returned. Else, there are specific rules for some words such as

If word == 'alot' : return 'a lot'

	kappa
Benchmark Bag of Words	0.64554
My Bag of Words	0.67863
Bag of Words and Bigrams	0.68789
Bag of Words, Bigrams and Trigrams	0.68514

Table -3: Usefulness of cleaned data [3]

Rather than using raw counts, Random Forest is used for training by changing the counts to term ferquency – inverse document frequencies.[4]

$$tfidf = (1 + log(tf)) \cdot log\left(\frac{N}{dfw}\right)$$

Where,

tf-idf : termfrequency – inverse document frequency

tf : term-frequency, raw counts of a word in an answer

idf : inverse document frequency

N : number of answers

dfw : document frequency of a term,

number of answers in which a word appers

	Kappa	
	With raw counts	With tf-idf weighting
Bag of Words	0.67863	0.65481
Bag of Words and Bigrams	0.68789	0.67005

Table -4: Comparing accuracy of raw counts and tf-idf

2.3 Feature Selection

Boruta Algorithm is a feature selection wrapper algorithm which is capable of working with any classification method that output variable importance measure (VIM)[5]. Boruta uses Random Forest [6]. It performs top-down search for relevant features by comparing original attributes with importance achievable at random. It also eliminates irrelevant features to stabilize test.

```
Boruta(x, ...)

# S3 method for default
Boruta(x, y, pValue = 0.01, mcAdj = TRUE, maxRuns = 100,
  doTrace = 0, holdHistory = TRUE, getImp = getImpRfZ, ...)

# S3 method for formula
Boruta(formula, data = .GlobalEnv, ...)
```

An object of class Boruta has finalDecision as a component which act most important in feature selection. It gives a factor a value out of 'Important', 'Confirmed' , 'Rejected' or ' Tentative'.

We trained Random Forests with these subset of words and obtained increase in performance and considerablee reduction of training time.

This predocure was repeated to obtain the 'impoertant bigrams'. These 'important words' and 'important bigrams' were only used for traning and development of models.

SET	**FEATURES ADDED**
1	1: 'how much vinegar'
	2: 'Size and type of container'

2	1: conclusion answer – ‘type of stretch’
	2: number of ‘ways to improve’ answers
3	1: ‘Alligator, Specialist, Generalist,’ have trouble adapting’ words in answer
	2: ‘Eat exclusively one type of food’
4	1: ‘major threat’
	2: ‘invade the everglade, area, property’
5	1: ‘RNA exists in nucleus’
	2: ‘Corresponding amino acid is added to RNA ’

Table -5: Features for different SET of questions(sample)

2.4 Modeling

The proposed system is using Gradient Boosting Machine (GBM) [6]. Boosting is a famous ensemble learning technique in which we are concerned with reducing the variance of learners.

Gradient boosting generates learners using the same general boosting learning process. It identifies hard examples by calculating large residuals (y_{actual} - y_{pred}) computed in previous iterations.

ALGORITHM	PARAMETERS
Random Forest	Default
Gradient Boosting Machine	Interaction depth = 5
	Bag fraction = 0.5
	Shrinkage = 0.001
	Number of trees = 500

Table -6: Parameters used in Algorithm

2.5 Prediction and Visualization

The prediction of final score to an answer is through the cumulative calculation of probability and importance which is done using Gradient boosting Machine and Random Forest algorithm.

Final scores to answers are received in CSV file and need to be made User friendly. Hence, Shiny R is used to take input from various text files (.csv) and organize them to generate a User Interface (UI).

3. RESULTS

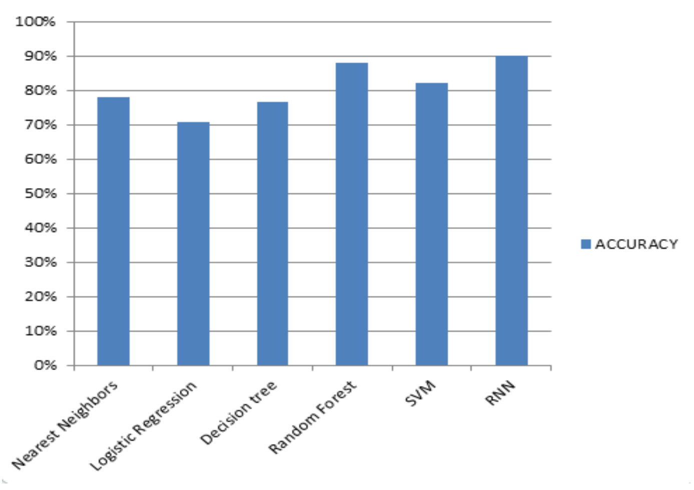

Chart -1: Performance Analysis Comparison Chart

4. CONCLUSIONS

After checking the compatibility and accuracy of various techniques we can state that RNN outcast Random Forest from a small margin but since Random Forest is helpful while dealing with attributes having different and variable importance value. Hence, we thought of using Random Forest and GBM to make system accurate and flexible as per dataset. Further if we have a huge training data build over multiple questions then we can extend our approach by introducing feature extraction mechanism which will automatically calculate the importance of a particular label by checking its frequency and similarities in other answers.

REFERENCES

[1] https://www.kaggle.com/c/asap-sas/data

[2] Norvig, Peter. "How to Write a Spelling Corrector." How to Write a Spelling Corrector. N.p.

[3] Porter, Martin. "Porter Stemming Algorithm." Porter Stemming Algorithm. N.p., Jan. 2006.

[4] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schutze. "Scoring, Term Weighting and the Vector Space Model." Introduction to Information Retrieval. N.p.: Cambridge UP, 2008.

[5] Miron B. Kursa, Witold R. Rudnicki (2010). Feature Selection with the Boruta Package. Journal of Statistical Software, 36(11), 1-13.

[6] A. Liaw and M. Wiener (2002). Classification and Regression by random Forest. R News 2(3), 18—22.

[7] Ridgeway, Greg. "Generalized Boosted Models: A Guide to the Gbm Package." N.p., 3 Aug. 2007.

[8] Brierley, Phil. "AusDM Analytic Challenge." AusDM Analytic Challenge. Tiberius Data Mining.

[9] Hassan, Samer and Mihalcea, Rada. "Semantic relatedness Using Salient Semantic Analysis", 2012.

[10] Valenti, S., Neri, F., & Cucchiarelli, A. (2003). An Overview of Current Research on Automated Essay Grading.

[11] Preston, D., & Goodman, D. (2012). Automated Essay Scoring and The Repair of Electronics, 1-18.

[12] Jones, Eric, Travis Oliphant, Pearu Peterson, and et al. "SciPy: Open source scientific tools for Python."

[13] Pedregosa, F., Weiss, R., & Brucher, M. (2011). Scikit-learn : Machine Learning in Python, 12, 2825-2830.