# Automatic Test generation via Machine Learning

**Proposal for a design project for the School of Electrical and Computer Engineering, Cornell University**

**By**

**Name:** ***Arjun Jauhari***

**Project Advisor:** ***Prof. Christoph*** **Studer**

**Signature:**

**Date:** ***December 16, 2015***

# 1 Abstract

This project is part of Machine Learning and Education Research aimed to build an automatic system for generation of test/exams. The project uses data dumps from 130+ stack-exchange websites to learn various measures like difficulty of question, quality of answers, ability of users in a particular specialization. The goal is to combine these learned parameters to generate a test on a particular topic with user defined difficulty. Data on these 130+ stack-exchange websites is huge, therefore a careful review is required to understand how data is arranged, what different attributes mean, so that it can be processed to extract out the information of interest. The next phase of the project aims to build upon this data a statistical model which will map the interaction between underlying features of data. The final phase will use this model to build a full system whose intended application will be to generate a test but it can be easily extended to applications like ranking users based on their answers, generating summary of most difficult questions, etc.

# 2 Introduction

Recently, use of Internet has become ubiquitous in Education. There is a rapid growth in volume of academic texts available online – lecture excerpts from courses, books chapters, Scholarpedia pages, and sources like technical blogs, Stack Exchange posts, Wikipedia. All these sources available today help a learner(student) to enhance their knowledge without depending much on other human but there are not sufficient ways available to assess one's knowledge in a particular discipline without using an expert(teacher) generated test. On the other hand, there is no scarcity of quality questions and corresponding answers on the Internet. Goal of this project is to use this huge pool of questions and develop a novel system which can create tests automatically solving the problem of expert dependent assessment of a learner.

# 3 Background

This project aims to use the data available on various Stack Exchange websites to generate tests automatically. But the scope of project is not limited to Stack Exchange only it can easily be extended to other similar websites like Quora. Stack Exchange provides the dump of their data in a well formatted xml files. Each data dump comprises of 8 files namely -

- Badges.xml
- **Comments.xml**
- PostHistory.xml
- PostLinks.xml
- **Posts.xml**
- Tags.xml
- **Users.xml**
- Votes.xml

Although all of these files contains important information but the one's which are in bold are most relevant for the implementation of this project. Each line in these files corresponds to a unique Post, User, etc defined by a row tag. All information is stored under attributes of each row. For example a sample row looks like -

< *row***Id** = ”1”**PostTypeId** = ”1”**CreationDate** = ”2015 − 02 − 03*T*16 : 40 : 26.487”**Score** = ”22”**ViewCount** = ”307”**Body** = ”*SampleQuestion*”**OwnerUserId** = ”2”**LastEditorUserId** = ”2”**LastEditDate** = ”2015 − 02 − 03*T*17 : 51 : 07.583”**LastActivityDate** = ”2015 − 02 − 03*T*21 : 05 : 27.990”**Title** = ”*SampleTitle*”**Tags** = ”*line − numbers*”**AnswerCount** = ”2”**CommentCount** = ”0”**FavoriteCount** = ”3”/ >

# 4 Issues

To generate a test which can be compared to a typical expert generated test, following issues must be addressed -

1. Classify the big pool of questions into various bins of varying difficulty. This classification will provide a provision to control the overall difficulty level of test.
2. Quality of each StackExchange user needs to be modeled. How good is a particular user, will govern the quality of his contribution which will help in doing #1 above.
3. Generated test needs to be diverse i.e. it should have a good mix of question's difficulty level, should cover all sub-topics, etc.
4. Provide multiple choices of answers with a balanced differentiation between right and wrong answers. The answers should be such that they don't make it obvious for test taker to guess the right one.
5. Design a metric which can compare the performance of automatic generated test with that of expert(teacher) generated test. This is essential to measure the performance of algorithm.

# 5 Approach

To address above mentioned issues a statistical approach is considered where first step is to identify the features which are relevant for the model. Secondly, model needs to learn the parameters which govern the interaction between these features. Below I describe the model.

## 5.1 Subscripts

- $i$ is subscript for question.
- $j$ is subscript for answer.
- $k$ is subscript for user.

## 5.2 Notations

1. $u_k$: Quality measure of the $k$th user.
2. $q_i$: Quality measure of the $i$th question.
3. $va_{ij}$: Normalized votes corresponding to $j$th answer of $i$th question.
   Calculated as: $va_{ij} = \frac{|sa_{ij}|}{\sum_j |sa_{ij}|}$
   where $sa_{ij}$ is the actual votes(score) read from data dump.
4. $a_{ijk}$: Quality measure of $a_{ij}$th answer given by the $k$th user.

5. $f_{acc}^{ij}$: Boolean flag telling if this answer was Accepted, read from data dump.
6. $r_k$: Reputation of the $k$th user, read from data dump.
7. $N_a^i$: Number of answer to $i$th question, read from data dump.
8. $vq_i$: Number of votes to $i$th question, read from data dump.

## 5.3 Equations

Below equations model the relation/dependence between the above defined parameters.

1. $a_{ijk} = f_a(u_k, q_i, va_{ij}, f_{acc}^{ij})$
2. $u_k = f_u(\{a_{ijk}\}_{ij}, q_i, r_k)$ , where $\{a_{ijk}\}_{ij}$ is set of all answers by user k
3. $q_i = f_q(u_k, N_a^i, vq_i, \{a_{ijk}\}_{jk})$, where $\{a_{ijk}\}_{jk}$ is set of all answers to $i$th question

## 5.4 Visualization

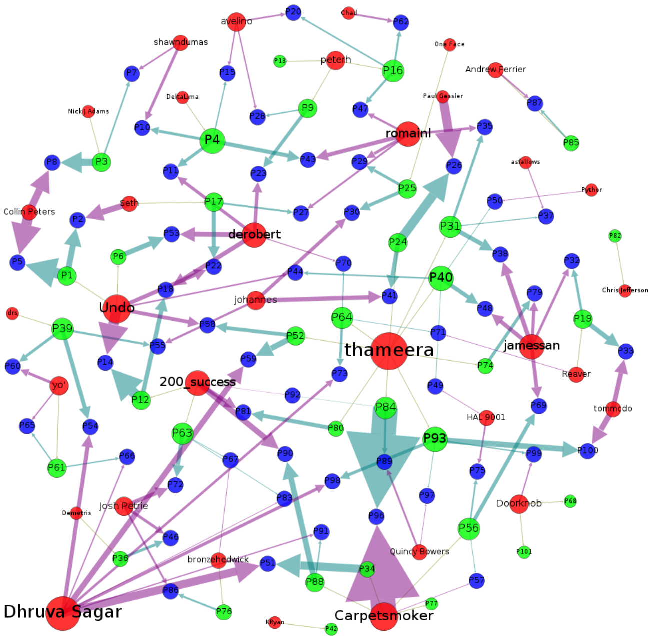

Figure 1: Sample Graph

To visualise this model I generated graphs using Gephi, figure 1 above shows one graph on a very small subset of data. Legend for the graph-

- Red Nodes: User
- Green Nodes: Questions
- Blue Nodes: Answers

- Green Edge: User2Question(u2q)
- Blue Edge: User2Answer(u2b)
- Grey Edge: Question2Answer(q2a)

The edges connect each user to their post. Post can be a question(green edge) or an answer(blue edge). Also, each question is connected to its answers through grey edge.

Important details about the graph -

- Node size represents, how active is that particular user. For example, user with name 'thameera' is the one of the most active user followed by user 'Dhruv Sagar'.
- The thickness of these q2a edges tells the quality of an answer relative to all the other answers for same parent question. For example, user 'Carpet Smoker' has give one of the best answer.

## 6 Status

Currently, the model is very simple with $va_{ij} = \frac{|sa_{ij}|}{\sum_j |sa_{ij}|}$, where $sa_{ij}$ is the actual votes(score) read from data dump.
and $a_{ijk} = w_1 * u_k + w_2 * q_i + w_3 * va_{ij} + w_4 * f_{acc}^{ij}$

As of now weights $w_1, w_2, w_4$ are 0 and $w_3$ is 1, so $a_{ijk} = va_{ij}$

User quality is being modeled as: $u_k \sim \mathcal{N}(mean(\{a_{ijk}\}_{ij}), Var(\{a_{ijk}\}_{ij}))$

Question quality is still not modeled.

## 7 Timeline and Deliverables

| **Date** | **Action Items** | **Deliverables** |
|---|---|---|
| October 2015 | 1. Explore Stack Exchange websites<br>2. Understand their data dumps | Summary of findings |
| December 2015 | 1. Extract the relevant features from data and visualize them through graphs<br>2. Implement the basic model in python | Basic implementation of model |
| March 2016 | 1. Extend the model to cover all features<br>2. Implement Machine Learning algorithm to learn all parameters | Extended implementation of model |
| May 2016 | 1. Use the learned model to write an algorithm which generates automatic test<br>2. Evaluate and refine the algorithm | Final working system |

## 8 Summary

This project targets to develop a system which automatically generates a test using the huge pool of questions and answers available online on Stack Exchange websites. Several important issues are being considered. I am taking a modular approach and improving incremently to take into account all the mentioned issues. It is not yet clear how well the above mentioned model approach will work. Therefore, I am trying to keep the model flexible. So that it generalize well and can be tweaked easily without much change.