IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 20, Issue 1, Ver. I (Jan.- Feb. 2018), PP 94-98
www.iosrjournals.org

Unified Testing and Recruitment with customizable shortlisting parameters

Kuntal Biswas
Corresponding Author: Kuntal Biswas

Abstract: *While industrialization began as a movement in 1800s, we are still basking in the effects of the same. Jobs are strewn everywhere if a person is qualified enough. It was fulfilling up to a few decades ago, providing jobs and uplifting men from hereditary obligation, to deserving positions in the society, based on their education. While this should be a merit of the modern society, it is startlingly inefficient, as of today, when population is ever on the increase and the competition is cut-throat, almost literally. Taking India as an example, the 2017-18 report by a leading news organization, namely 'The Times of India', projects the unemployment rate at 3.4% for the year 2017, which remains stagnant at the same number for 2018. While this might not seem a lot in a single glance, the actual number rises to a staggering height of 17.8 million for 2017, which goes on to 18 million in 2018. This becomes even more alarming when delving into the details, where a simple Google search shows that 1.5 million engineering students are graduating every year, and over 60 percent of them remain unemployed. The situation gets dire for the general studies students namely B.Sc., B.A etc. where campus selections are never offered. The fresh graduates, without a hint of experience have to compete with resigned employees, unsatisfied workers with years of experience, for the same job position. The primary requirement of any job interview clearly states an experience of at least 2-3 years, in the concerned subject. A fresh graduate can never hope the get to the interview round to showcase their expertise, when their application gets rejected simply for not making the cut. Without job, they cannot get any experience. It is the chicken-egg scenario, applied in real life.*

While a complete change in hiring system is thoroughly illogical, an impartial system that recommends candidates based on their practical merit, to overcome the minimum requirement standards, could bring a hopeful change in this hopeless struggle to earn a job. The gist of the proposition lies in the fact that an employer expects the employee to do the job efficiently, honestly; not trusting by just the brand of degree possessed by the employee. For example, a company designing android phones needs employees for their software team, who can tailor the software according to the need that the parent company decides. They require Android coders, efficient in JAVA, web design, and highly efficient microprocessor coders, not a student of completely unrelated branch just because he/she may be from a highly reputed institution. That is just to state one such occurrence. Numerous such examples exist, where students of renowned universities, but of unrelated streams are offered jobs, instead of a more competent student from a lesser known university. What this results in, is inefficiency on the clock. The company will need to organize crash courses aiming to cram 4 years' worth of knowledge and hands-on experience in 4 weeks or less time. These funds could be used to provide better equipment to fresh students of same stream from lesser institutes for whom, these courses will be nothing more than refresher review.

We are proposing an idea of such a system, which could allow a similar shortlisting that to a contemporary employee selection algorithm, while being impartial to the minimal requirements stated, as long as the candidate is capable of being suited for the job. We also propose an examination system that allows a better way of selecting the perfect application from a prospective employee pool. The examination system allows anyone with a bachelor's degree, at the bare minimum, to appear for a subject test, regardless of the graduation background, institution or location. This rules out the possibility of a bright student getting neglected due to trivial matters like, being a student of a lesser known institution or branch, since this test is unified and not confined to the campus of a particular university/college.

Date of Submission: 15-01-2018 Date of acceptance: 05-02-2018

I. Introduction

A selection process is inevitably dependent on the system by which it deems the eligibility of the candidate. While the marks and GPA are a good measure of knowing the academic strength of a student, it leaves a lot to desire on knowledge about their professional competence. As a result, most companies assessing for a campus selection, organize multiple rounds of tests before shortlisting students for their interview round. Although this process is tried and tested to work for an individual company, it should be noted that a student is also appearing for a number of exams set by other companies, at the same time. It exerts an incredible amount of pressure on the student, to manage their time and energy. It should also be noted, that such tests are redundant and costly for every single company involved in the campus selection process.

A better idea could be a unified testing system that students can apply for. The specialty of such a system relies on the fact that students apply for very concise topics of their choice, instead of a subject that encompasses a very broad field. Doing so, reduces the redundancy of testing for the same subjects, since they already have their GPAs to show their general intelligence in academics. They should be able to showcase their skills in topics which are usually not covered in colleges. For example, Swift programming language as a regular subject, is not taught in any university in India, as are languages like ruby, pearl, genie etc. As a result, the software companies looking for employees, have to rely on the general subjects that are already being taught, to gauge their competency in an unfamiliar topic. Here, we store a table of questions that are marked with tags of various topics. Moreover, these are such questions which, instead of bookish knowledge, arise on a frequent basis while working on a block of code, or such. This table also contains the level of such questions, and an increasing score based on the level of question.

II. Methodology

The whole system relies on a series of tables in a database, and a very simple interface which allows seamless access of the stored data.

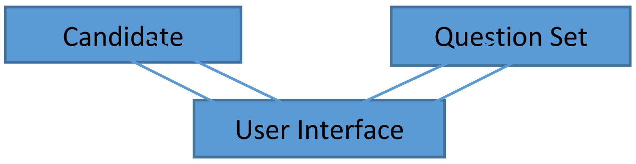


Fig: Structure of the test system

Two distinct tables are maintained, one with the candidate details, and the other with the question set. The user interface allows seamless access to both tables, while in two different modes: 1. Student, 2. Employer. The student mode allows the student to appear for a particular subject test, one at a time. The results are stored along with the student data, in the candidate table. Only one entry per candidate is allowed, and subsequent appearances for different subjects is simply appended as a string to the earlier score. The Employer mode allows the selectors to shortlist the candidates, based on their requirement. The operations are simple fetch and store, which can be run on even modest of computers.

An example an entry in the Question_set table is:

Question number	Question	Level of question	Marks	Subject Tags
xx123	How will you evaluate a string in a given format, into its original data type?	Level 1	3	Python, Algorithm

What this does, is allows this single question to be used for two distinct subjects of Python and Data Structure/Algorithm. Benefit of such a system allows the panel of selectors to know that a student who can answer such questions, is equally proficient in both the topics. Even if one of the subjects mentioned in the tag is not taught officially, then the employer will not have to worry about the efficiency of the candidate by guessing. They can simply see for themselves if the candidate is skilled enough for their line of work. Numerous such examples can be stated.

Next step is inevitably the storage of the results in such a way, that it is accessible and decipherable by the selectors, according to their needs. Again, we maintain a table, which stores the results in the following manner:

Serial Number	Name	Age	Experience	Qualification	Score
xy123	Tom	25	2yrs	B.Tech	Python 85 Java 70 Math 35

Do note the storage of subject and marks as a single string. Since a student will appear for more than a few subjects, it becomes easier to store the scores as appends to an existing string, which can be accessed easily using simple string manipulation functions, for example:

1. Locate existence of subject using y=string.find(x) function, where the location of the subject is stored as an integer in y.
2. Using (y + length of subject name + 1) as the index, store the integer value, separated by a space after the subject name, in z.

3. Then we type cast z as int(z), and use that value to sort the candidates accordingly.

Then we arrive at the point where shortlisting is done. For this, we consider an example, where we manually evaluate a few entries stored in the application pool:

Ab1 | Tom | 25 | 2 | B.Sc. | Python 75 Java 70 Math 35
Ab2 | Ram | 22 | 0 | B.Tech | Python 85 Java 75
Ab3 | Shyam | 23 | 1 | B.Tech | Java 63 Microprocessor 85 Math 45
Ab4 | Bob | 25 | 1 | B.Sc. | Python 60 Hindi 70 Math 50

Suppose, a company wants one employee to work on a project in python programming language. The applicable employees are: Ab1, Ab2 and Ab4. Now, how do we arrive at a sorting process, which is equally efficient in eliminating the obvious experience difference among the candidates? We start by analyzing, what a year of experience in a particular topic does to an employee. One year is sufficient to learn the comparative strength of a subject, its idiosyncrasies and nuances, while leaving a profound gap in the ability to improvise and think on the spot for a workaround. This is because, the workarounds are ideally pored over by the team head, a position which a candidate needs at least 3 years to attain. The team head has to consider the effect of the workaround, which is imposed on different groups working on the fragments of the same project. Hence, we set some similar rules while comparing two candidates:

1. If experience1 - experience2 >= 3:

Candidate1 is preferred, i.e. if the experience difference is more than 3 years, then the more experienced candidate is preferred, even if the score difference states otherwise.

2. If experience1 == experience2 AND score1 >= score2:

Candidate1 is preferable to candidate2, i.e. if experience remains same, then the candidate with higher score is preferred.

3. If (experience1 - experience2 <=2) AND (score2 – score1 >= 10 percent):

Candidate2 is preferred, i.e. if the score difference is greater than 10 percent, while experience difference lies between 0 and 2, the candidate with higher score is preferred.

4. If (experience1 - experience2 <=2) AND (score1 – score2 >= 10 percent)

Candidate1 is preferred, i.e. if the score difference is greater than 10 percent, while experience difference lies between 0 and 2, the candidate with higher score is preferred.

According to the given set of rules, the candidates should be sorted in the following manner:

Ab2 | Ram | 22 | 0 | B.Tech | Python 85
Ab1 | Tom | 25 | 2 | B.Sc. | Python 75
Ab4 | Bob | 25 | 1 | B.Sc. | Python 60

Which is justified, since Ab2 with no experience scores equal to or more than 10 percent than Ab1, which means that Ab2 is better suited to a python job than Ab1. Similarly, Ab4 scores considerably less than Ab2, and 15% less than Ab1, which means that Ab4 is the least suited for the job, despite having more experience than Ab2.

III. Results

The prototype program is designed in python. A simple interface has been designed, that shows results and prompts in the interpreter window itself.

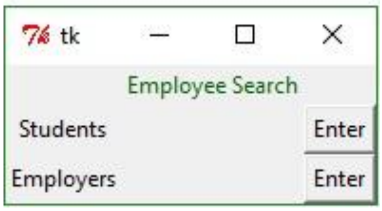


Fig: The main interface

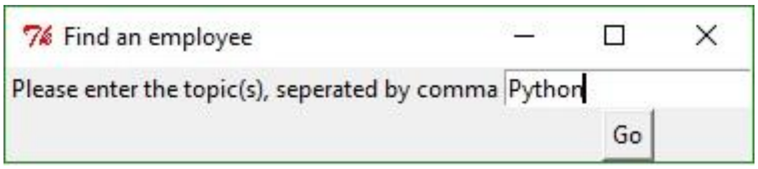
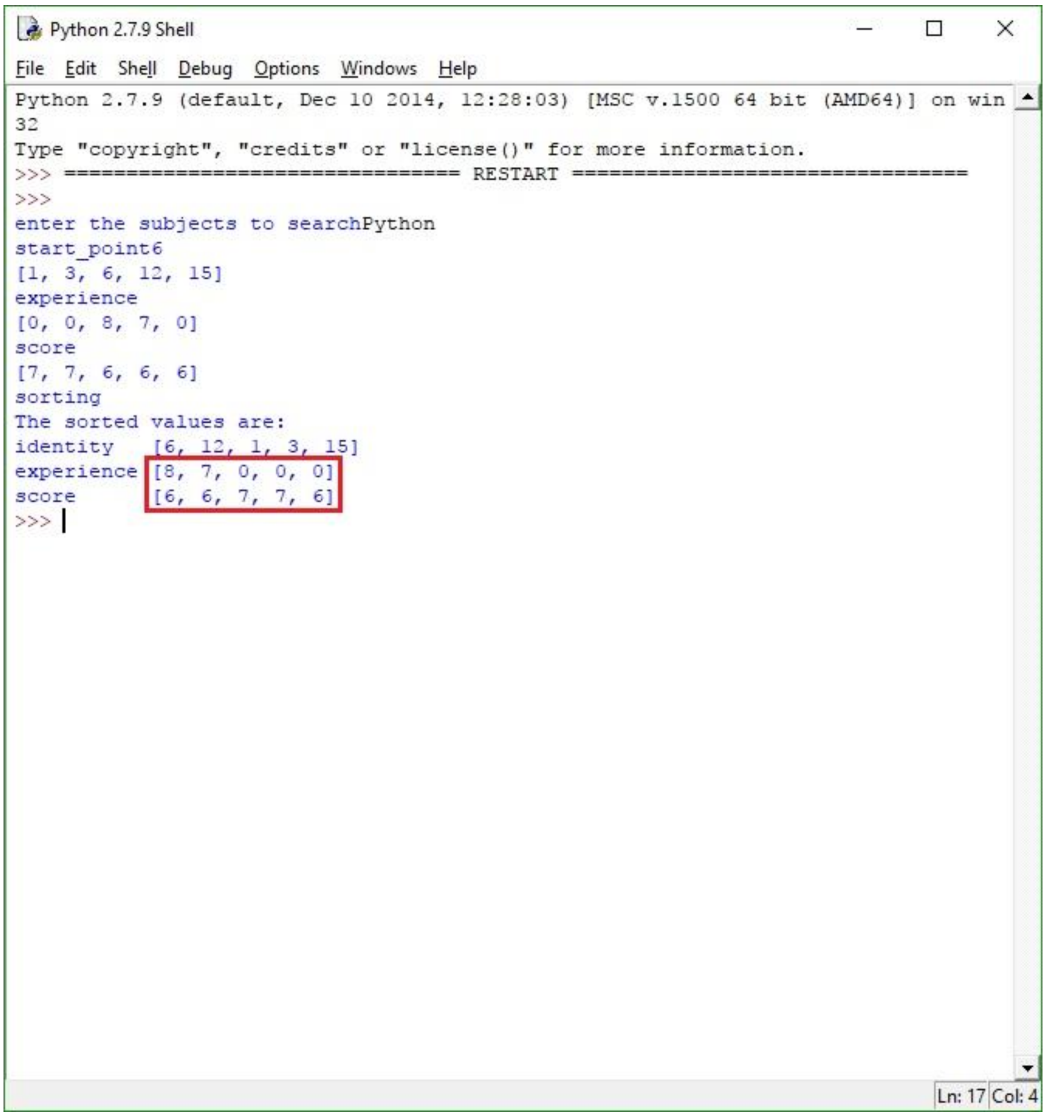


Fig: To search for students with specific skill-set. Multiple subjects may be entered



```
Python 2.7.9 (default, Dec 10 2014, 12:28:03) [MSC v.1500 64 bit (AMD64)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
enter the subjects to searchPython
start_point6
[1, 3, 6, 12, 15]
experience
[0, 0, 8, 7, 0]
score
[7, 7, 6, 6, 6]
sorting
The sorted values are:
identity    [6, 12, 1, 3, 15]
experience [8, 7, 0, 0, 0]
score       [6, 6, 7, 7, 6]
>>>
```

Fig: The sorted student list, according to the set rules

IV. Conclusion and Future Work

The coding for the prototype software has been kept to the bare minimum, to emphasize on the main algorithms namely, the tag structure and sorting methods. A good security measure can be implemented to prevent tampering with the database. The students should be encouraged to pursue subjects beyond their streams. To do so, a provision could be implemented, which averages the subjects and streams offering the most number of jobs, to serve as an encouragement to those who feel their knowledge is inadequate and are trying to get noticed among the employers. Lastly, I hope this idea helps the students as well as the employers alike, and ultimately, makes our life easier.

Kuntal Biswas "Unified Testing and Recruitment with customizable shortlisting parameters." IOSR Journal of Computer Engineering (IOSR-JCE) 20.1 (2018): 94-98.