

Capstone Project Report on

# Machine interviewing for more efficient applicant screening

*Submitted to*

**Praxis Business School, Kolkata**

*in fulfillment of the requirements for the award of the degree*

**Post Graduate Program**

in

**Data Science**

by

Anupam Misra C21001

Radha Rathore C21008

SHAHRUKH GOUHAR C21013

Mansoor Ali Shaik C21014

Under the guidance of

Dr. Subhasis Dasgupta



Department of Data Science  
Spring 2021

## Acknowledgments

We are profoundly grateful to Dr. Subhasis Dasgupta for his expert guidance throughout the project and for providing valuable suggestions at crucial junctures. We would like to express our deepest appreciation towards Dr. Prithwis Mukerjee and Prof. Charanpreet Singh, Founders & Directors, Praxis Business School, Kolkata. and Dr. Sourav Saha, Dean of Academics. Lastly, we would like to express our sincere heartfelt gratitude to all the members of the PGP Data Science Department who helped us directly or indirectly during this course of work.

Anupam Misra

Radha Rathore

Mansoor Ali Shaik

SHAHRUKH GOUHAR

Date: 10 November 2021

## Abstract

There is a large gap between the supply and demand of capable data scientists in India. The hiring managers need to put in a lot of effort to select the right candidates. Automated testing and phone interviews exist to initially screen the candidates. However, these techniques are inflexible to unique candidate requirements and prone to human bias. We have built an end-to-end solution for personalized automated interviewing of candidates. These candidates are scored using machine learning to prevent human bias and fatigue. Our solution will allow hiring managers to focus on more important tasks while making hiring more efficient.

# Contents

## Chapters

01   Introduction	1
02   Creating the dataset	3
03   Testing existing approaches	4
04   Final scoring architecture	7
05   Client web application	13
06   Candidate web application	16
07   Score generator	19
08   Conclusion and further work	22

## *Bibliography*

## List of figures

Figure 3.1 Comparison of MSE among available Sentence Transformers	4
Figure 3.2 Comparison among models across expected scores	6
Figure 4.1 BERT	7
Figure 4.2 New neural network	8
Figure 4.3 Neural network for creating custom embeddings	10
Figure 4.4 Architecture used in the scorer	12
Figure 4.5 Output score after adjusting the scorer	12
Figure 5 Client web app structure	13
Figure 6 Candidate web app structure	16
Figure 7.1 Batch scoring program structure	19
Figure 7.2 Scorer module in detail	21

## List of tables

Table 3 Comparison of MSE among top 3 models for the two types of approaches	6
Table 4.1 Performance on the dataset with different embedding dimensions	10
Table 4.2 MSE performance of different ML models on our test dataset	11
Table 4.3 Top three models considered for final scoring	11

# Chapter 1

## Introduction

### 1.1. Motivation and background

The IT industry is hiring in huge numbers presently. On average, for every open position, at least ten candidates are applying. Automated resume screening applications already exist. We wanted to automate the next stage of the hiring process: interviews. We read about chatbots and conversational AI to understand how to conduct interviews. Since we are studying and interviewing with data science companies, we decided it would be interesting to make an interview bot to interview candidates on data science questions.

### 1.2 Objective

There is a large gap between the supply and demand in the data science sector in the country[1][2]. Many companies use automated resume screening. Even then the hiring managers are faced with the problem of inviting a large no. of prospective applicants to interviews.

Some companies do telephonic interviews as the first line of screening. More often than not, human emotions take precedence over logic which makes the whole process biased. We have designed an interview system aimed to replace this stage of screening for a more fair evaluation.

While automated proctored examination platforms exist, they are not flexible to accommodate hiring requirements. Our application provides tailor-made assessments for hiring each unique applicant and filling each unique position. The client can select the different topics for the aspirants and email them a test link.

The aspirant would register in the candidate portal, log in and take the interview. The results would be computed on a daily basis and report(s) would be generated.

## 1.3 Business implications

Our application is designed to help businesses redefine their interview process by enhancing and optimizing it. The aim is to reduce time, cost, and effort by making the interview screening process smarter, faster, and fair.

The first stage of any interview process begins with preparing a proper interview schedule for numerous candidates, which is a long, tedious, and error-prone process. It is a hectic task of informing each candidate and making a slot available for them based on the availability of an interviewer. This is solved smartly by our application by parallelizing the process. As the candidates are evaluated by a machine, there is no scope of bias. This will result in better screening of candidates. The evaluation process is based on advanced machine learning logic which compares users' answers with standard answers and marks them based on the level of similarity.

Detailed reports after the interview would be helpful in better analysis of candidates' performance and making the hiring decision.



# Chapter 2

## Creating the dataset

### 2.1 Gathering and structuring the data

Data is the main requirement for any data science project. As we were going to conduct interviews, we built a repository of question-answer pairs. We created a databank of questions from Statistics, Linear regression, Logistic Regression, KNN, SVM, KMeans, Decision Tree, and Naive Bayes. We kept questions of three difficulty levels from the aforementioned topics. In total, we created around 250 questions. The question-answer pairs were either web-scraped or entered manually. The answers were scrutinized to remove non-contextual words and include specific relevant keywords.

### 2.2 The need for multiple answers

During several rounds of testing, we realized that some questions could be answered with multiple keywords. Hence, up to three answers were written for those questions wherever applicable. During scoring, the user's answer closest to either of the three answers was considered for scoring.

### 2.3 Creating a separate test dataset

We tried different approaches like sentence-transformers, LSA, and neural networks for scoring the user answers. As we are working with a very niche text vocabulary we had to separately create a test dataset comprising question-answer pairs from the original dataset. However, in this dataset, we manually added a new attribute called user-answer to indicate user answers of different levels of correctness against each question. Expected scores were also allotted against each user-answer variation to train and evaluate the different scoring models.

# Chapter 3

## Testing existing approaches

### 3.1 Introduction

Machine learning lies at the heart of our interview bot. The interview bot evaluates how close the user's answer is to the expected answer and allots a score accordingly.

Several pretrained transformer models were available, which had been pre-trained on a huge corpus of data to capture a global perspective. However, as we are working on a niche segment of text comprising technical vocabulary, we are not going to solely rely on pretrained models.

To evaluate the several sentence transformers we used MSE instead of MAPE as many expected scores were zero. We used the test dataset on the 21 pretrained sentence transformers available at HuggingFace [3]. The MSE scores obtained are shown in figure 3.1.

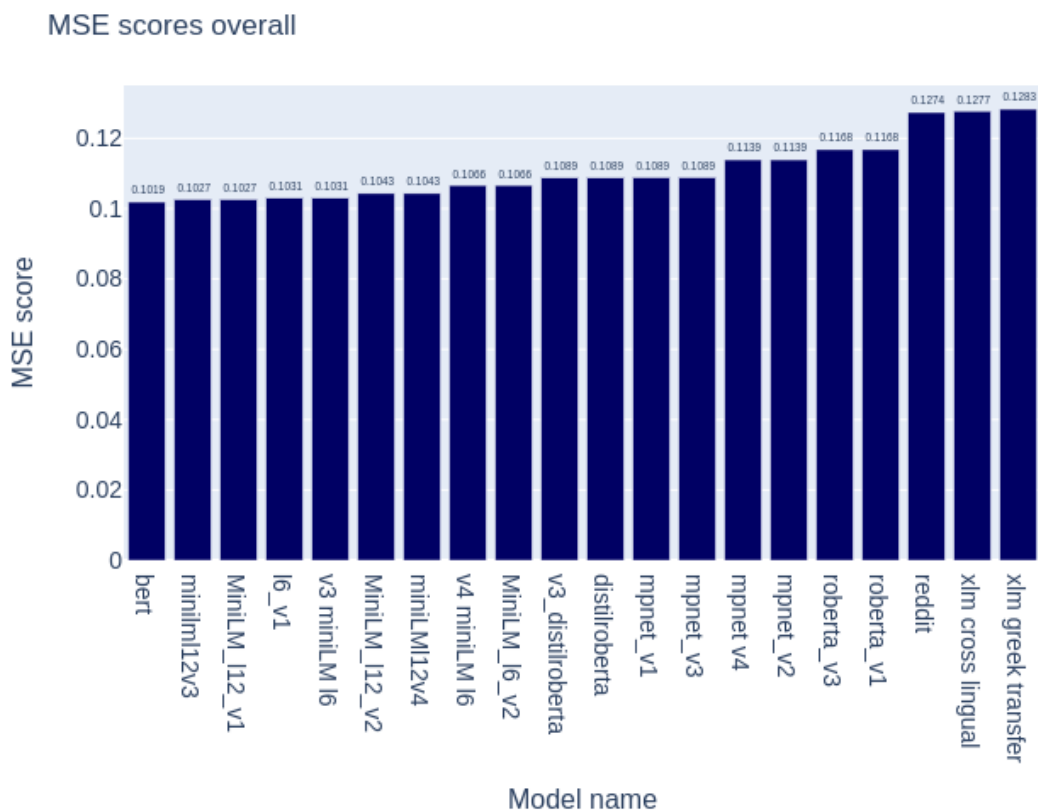


Fig. 3.1 Comparison of MSE among available Sentence Transformers

BERT was also pre-trained on our dataset. There was no appreciable reduction in MSE. Hence we decided to test different approaches by training other models on our test data.

The different local approaches tried:

1. Doc2Vec

The Doc2Vec model was trained on our test dataset for 500 epochs with a window size of 2. The embedding dimension was 500. With the dataset being very small, as expected, the Doc2Vec performed very poorly as measured by the MSE between the expected score and the scores predicted by the Doc2Vec model.

2. LDA

Latent Dirichlet Allocation can detect latent topics in the dataset. In our case, each question is a topic. So this approach would not work.

3. LSA - Count and TFIDF Vectorizer

Count Vectorizer with `n_gram` size as 10, filter as English words, and no. of SVD components as 500 was fit on our dataset. TFIDF Vectorizer with `n_gram` size as 10, filter as English words, and no. of SVD components as 500 was also fit on our dataset. In both these approaches, the MSE did not drop below that of BERT and there was always a chance of overfitting.

4. New neural network

We decided to train an embedding layer specifically on freely available data science textbooks, lectures, and our databank. The correct answers and the answer variations from the test dataset were fed through this into two GRU networks. Cosine similarity was taken between the outputs from the GRU units to generate the score. This yielded the lowest MSE. More about this network is discussed in the next chapter.

Model	Overall MSE
<i>Locally trained models</i>	
New neural network	0.043
LSA CountVectorizer	0.117
LSA TfidfVectorizer	0.134
<i>Pretrained models</i>	
BERT	0.102
MiniLM L12 v3	0.103
Mini LM L6 v3	0.103

Table 3 Comparison of MSE among top 3 models for the two types of approaches  
MSE for final model is 0.0059. It is discussed in detail in the next chapter.

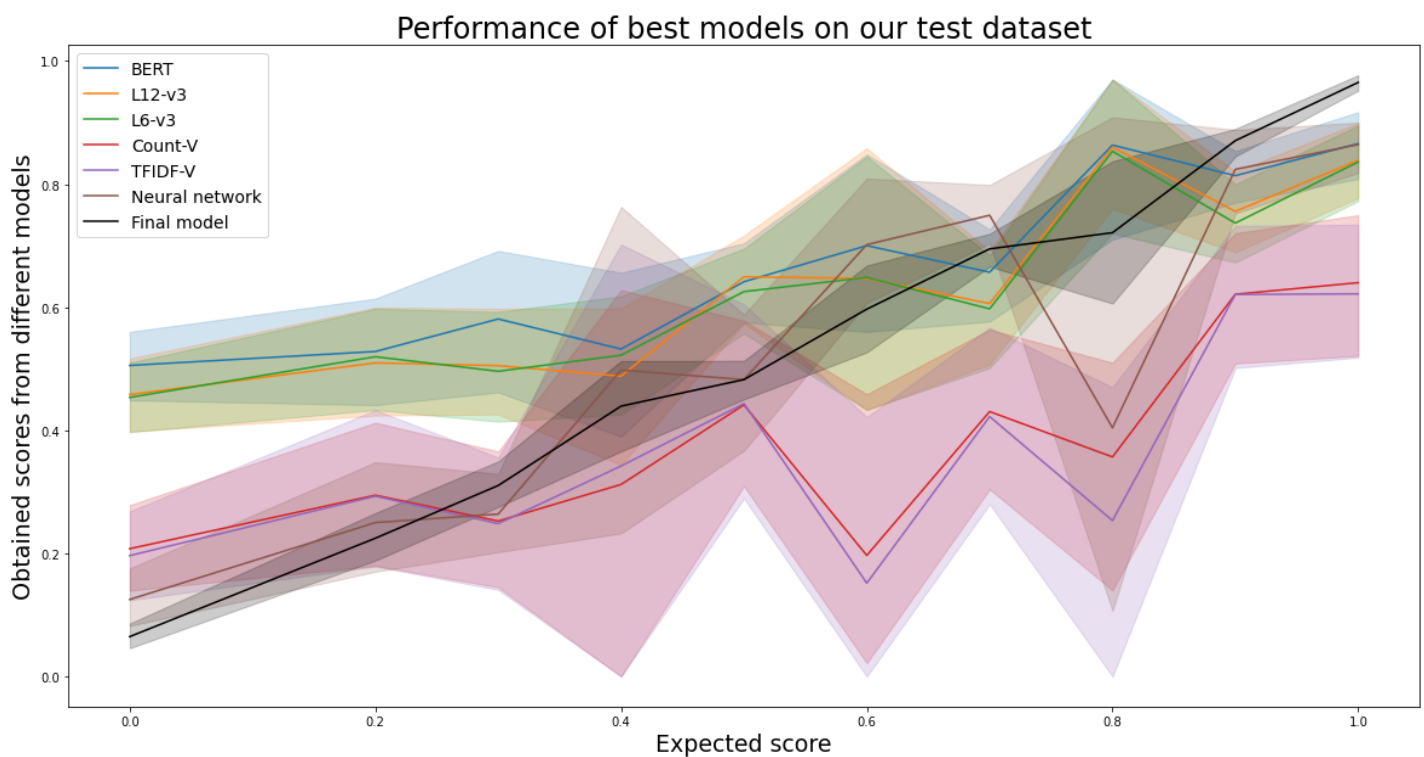


Fig. 3.2 Comparison among models across expected scores

# Chapter 4

## Final scoring architecture

### 4.1 Introduction

As seen from the previous chapter, our new neural network performs exceptionally well on the test dataset. However, there is a chance that it has been overfitted. Hence we will use this neural network in conjunction with BERT which was pretrained on a huge corpus of data.

The locally trained model on our corpus will be able to capture the technical keywords required in the interview. The global model will be able to capture nuances in the English language while scoring the answers.

### 4.2 Global approach

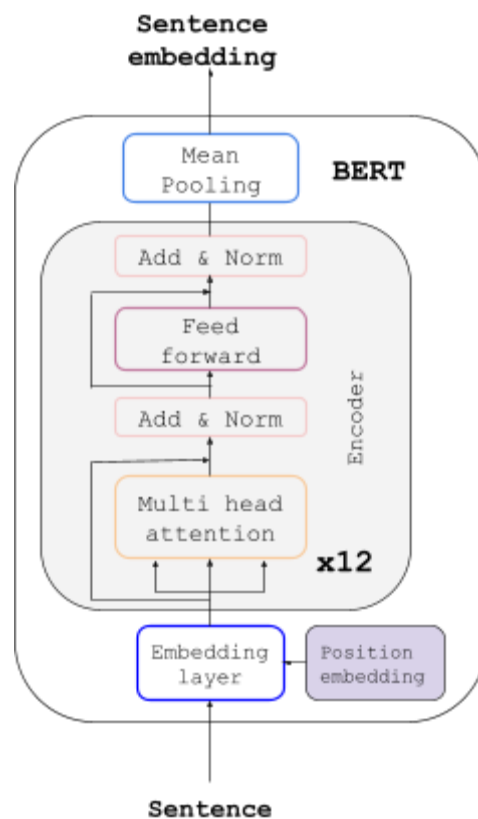


Fig. 4.1 BERT

BERT:

BERT [4][5][6] is a language model trained bidirectionally on a huge text corpus. The bidirectional nature enables better context understanding and sense of language.

We are using BERT as 12 encoders followed by mean-pooling to generate the sentence embeddings. The input max length is limited to 128 words and the embedding dimension for each word in the sequence is 1024 dimensional. The structure of BERT is shown in figure 4.1.

## 4.3 Local approach

### 4.3.1 New architecture

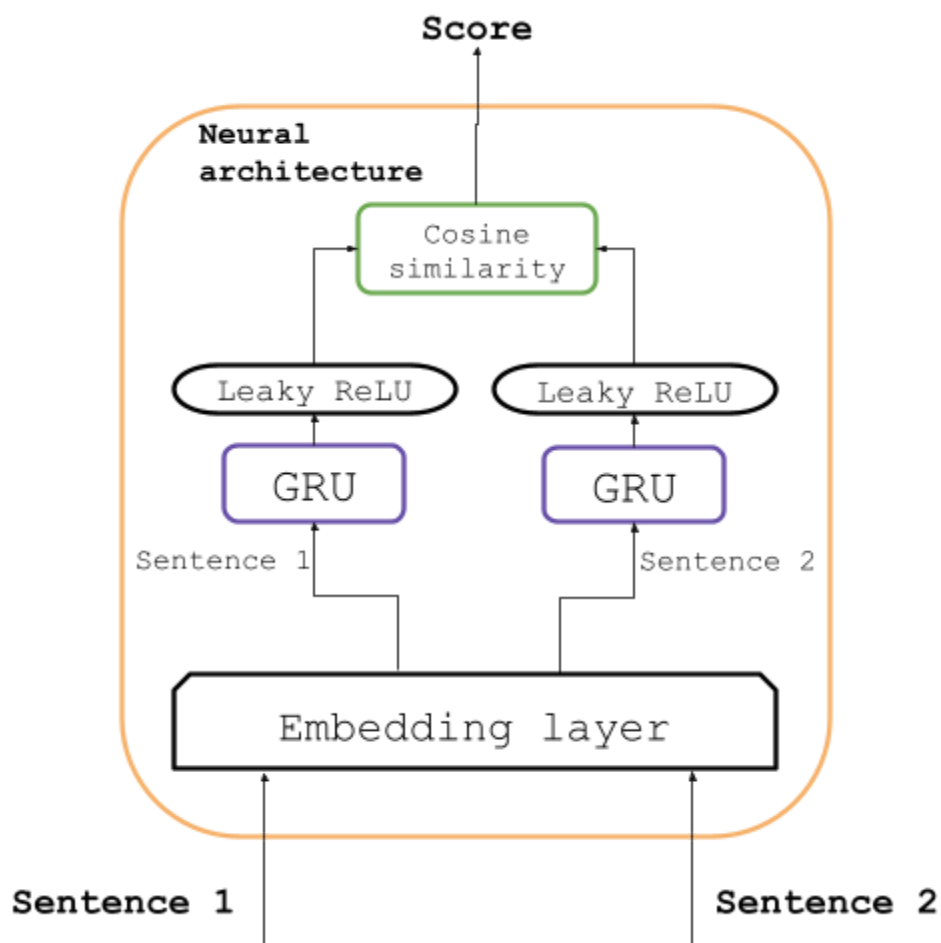


Fig. 4.2 New neural architecture[7]

Figure 4.2 shows the components of our new neural network. The different parts of the neural network are discussed below.

### Embedding layer

The embedding layer is of 64 dimensions with a vocabulary size of 7709 words. The embedding layer receives sentences after the words are converted into dictionary indices from the text corpus and padded to ensure the sequence is of the same length.

### GRU:

We are using a 2-layer unidirectional GRU with a hidden size the same as the max sequence length of 50. If the answer is greater than 50 words, only the first 50 words available in the vocabulary are used.

### Leaky ReLU:

During different iterations and configurations we found leaky ReLU to perform best on both the training and the testing datasets. It alleviates the problem of dying activation when embedding values are negative.

### Cosine similarity:

We are checking the cosine similarity between the two sequences after activating the GRU network output. This step will evaluate how close the user's answer is to the correct answer.

## 4.3.2 Custom embeddings

We have used dump from open-source books to create the unique words in the embedding layer. The preprocessing done on the text:

- Removing non-alphabetical characters
- Lowering of the text
- Tokenization
- Creating word dictionary

These words from the text corpus were used to create skip-gram pairs using `keras.preprocessing.sequence.skipgrams` [8]  
The embedding layer was trained on these skip-gram pairs using a neural network[9] as shown in fig. 4.3 below.

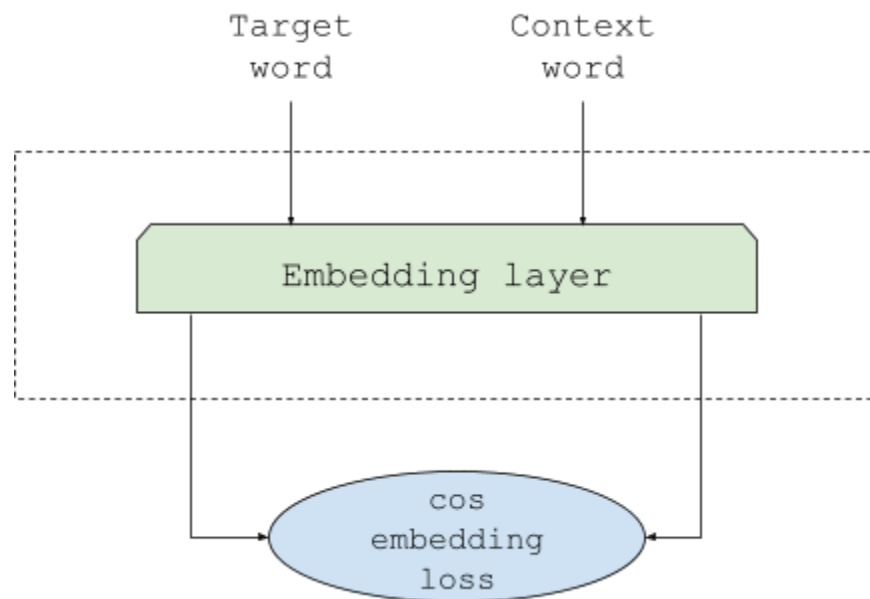


Fig. 4.3 Neural network for creating custom embeddings

The vocab size was 7709 and the embedding dimension is 64. Window size is taken as 4. The model was run for 40 epochs to achieve a cos-embedding loss of 0.296 on the test set.

Embedding dimension	Training loss	Testing loss
16	0.6418	0.3165
32	0.6106	0.2997
64	0.6031	0.2961
128	0.6096	0.2999
256	0.6324	0.3108

Table 4.1 Performance on the dataset with different embedding dimensions



## 4.4 Final model

The scores generated by the two models are fed into a machine learning model which predicts the final score.

We trained different machine learning models on our data. The MSE of the ML models on the test dataset without any hyperparameter tuning are shown below:

Model	MSE( test dataset)
Random Forest	0.0375
SVR	0.0396
Linear Regression	0.0397
AdaBoost	0.0463
XGB Regressor	0.0479
GB Regressor	0.0511
Decision Tree	0.0721

Table 4.2 MSE performance of different ML models on our test dataset

We tried hyperparameter tuning of Random Forest and SVR to obtain improved results as below:

Model	Hyperparameters	MSE( test dataset)
Random Forest	Max depth = 12, max features = 1	0.0059
SVR	Kernel = 'linear', C=0.5, gamma=0.2, epsilon=0.06	0.0363

Table 4.3 Top models considered for final scoring

We used Random Forest as our final model.

The final scoring architecture is as shown below:

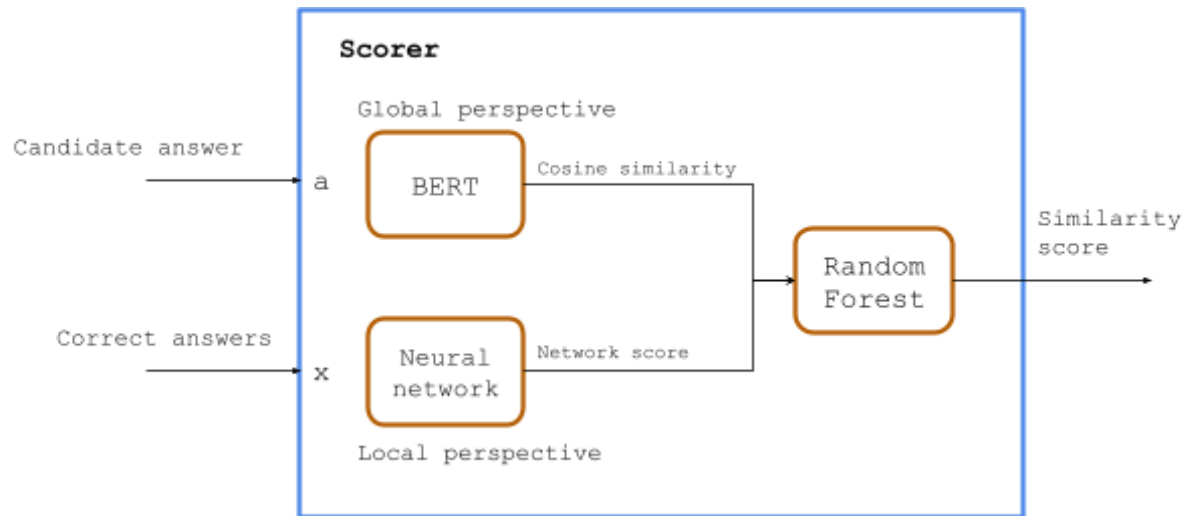


Fig. 4.4 Architecture used in the scorer

### Adjusting the similarity score for nuances in language:

Generally a similarity score of 0.5 is achieved if similar words but an incorrect answer is entered. Hence we give a score of zero upto the similarity score of 0.5. Also, Above 0.9, the answer is correct but may have used slightly different language. Hence we give a similarity score of 1. In the range (0.5,0.9] we scale the score as below:

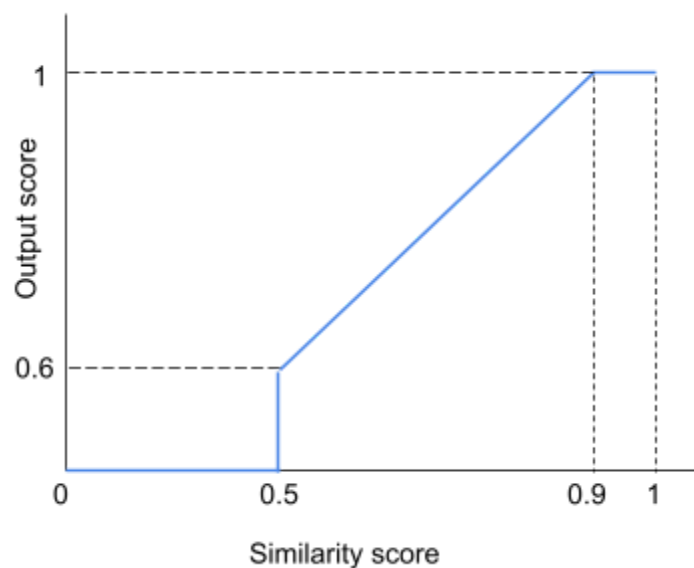


Fig. 4.5 Output score after adjusting the scorer

# Chapter 5

## Client web application

### 5.1 Introduction

We plan to generate personalized interviews for each candidate. The client web application is built to cater to the needs of the hiring manager.

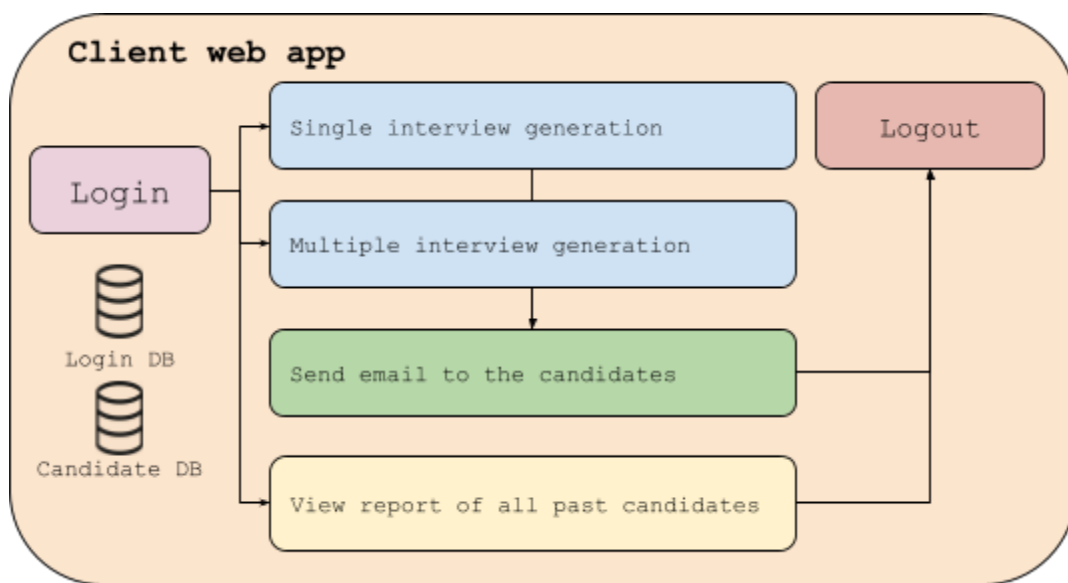


Fig. 5 Client web app structure

### 5.2 Secure login

The hiring manager can log into the web app using his credentials. The login credentials are verified from the Login DB which is stored locally in the machine running the client web application server. The data exchange between the client application and the server is encrypted using hashing. The user session is managed using a Login Manager.

## 5.3 Generating the interview

### 5.3.1 Single interview generation

Sometimes the hiring manager may require to fill only one position. In this case, downloading and filling a template is cumbersome. The authorized person would select the topics to test a candidate on and enter his email ID. He would also mention the deadline for taking the interview.

### 5.3.2 Multiple interview generation

Generally the hiring manager would collate the manpower requirements and send the emails to multiple candidates at once. The authorized person would download the Excel template for sending the interview emails. He would then fill the template with the topics against each candidate's email ID and upload it. He would also mention the deadline for taking all the interviews.

## 5.4 Sending email and storing aspirant data

The application sends an email to each candidate with the topics to be tested in, the deadline, and the interview link. The candidate email IDs and the topics are stored in a remote interviewee DB for access during candidate registration. When the candidate registers in the candidate interview portal, the portal checks whether an email had been sent to the candidate, otherwise, it prevents the candidate from registering for the interview.

## 5.5 Viewing reports

Interviews are taken to make hiring decisions. After the candidates take their interviews, their performance is stored in a database and their overall performance is available in the client application.

## 5.6 Hosting on AWS

The interview generation application is expected to be used by multiple hiring managers in the client organization. Hence it was hosted on an AWS EC2 instance for use within the organization. The security group was set up to only allow my machine for client application.

Here are some snapshots of the client web application.



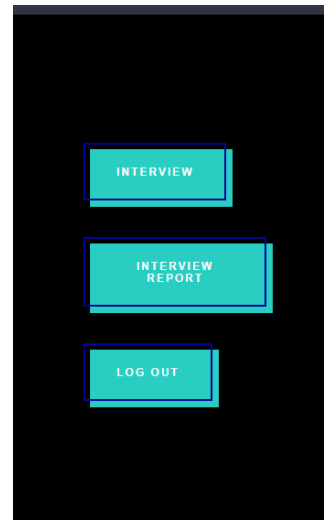
The client side of the awesome data science company

Sign In

Username

Password

Login page



Dashboard



Switch between single and multiple interview

### Generate a single interview

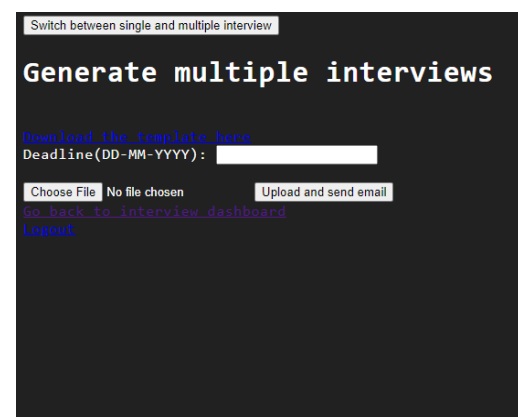
- ☐ Statistics
- ☐ Linear Regression
- ☐ Logistic Regression
- ☐ KNN
- ☐ SVM
- ☐ KMeans
- ☐ Decision Tree
- ☐ Naive Bayes

Enter the interviewee email:

Deadline(DD-MM-YYYY):

[Go back to interview dashboard](#)

[Logout](#)



Switch between single and multiple interview

### Generate multiple interviews

[Download the template here](#)

Deadline(DD-MM-YYYY):

No file chosen

[Go back to interview dashboard](#)

[Logout](#)

Generating interviews

## Report of people who have already taken the interview

Interview date	User Login	Topics	Percentage (%)	Passed	Score	Total
2021-10-21	sharukh.gouhar@praxis.ac.in	KNN, SVM, Statistics	82	Yes	123	150
2021-10-22	radha.rathore@praxis.ac.in	SVM, Decision Tree, Naive Bayes	61	Yes	92	150
2021-10-23	mansoor.ali.shaikh@praxis.ac.in	Linear Regression, KNN, KMeans	51	Yes	77	150
2021-10-23	geetha.joseph@praxis.ac.in	KMeans, Logistic Regression, Decision Tree, Statistics	88	Yes	176	200
2021-10-22	ashwin.kumar@praxis.ac.in	KNN, Naive Bayes	68	Yes	68	100
2021-10-25	yash.aswani@praxis.ac.in	Statistics, Logistic Regression, KNN, SVM	56	Yes	112	200
2021-10-20	anupam.misra@praxis.ac.in	Linear Regression, Logistic Regression, Statistics	66	Yes	100	150

[Dashboard](#)  
[Log Out](#)

Overall report of candidates

# Chapter 6

## Candidate web application

### 6.1 Introduction

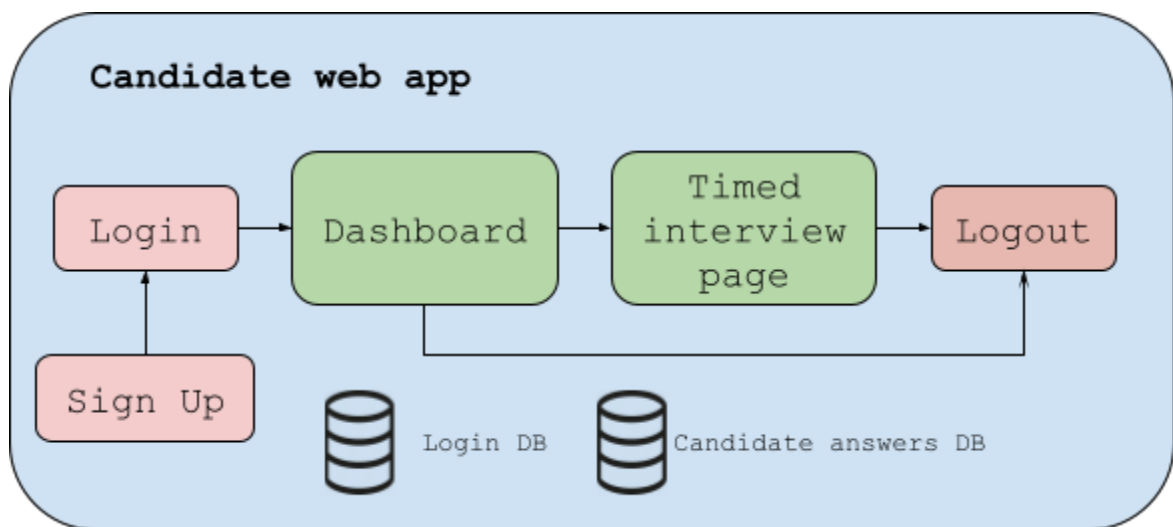


Fig.6 Candidate web app structure

### 6.2 New user registration

Candidates who received the email for an interview register for the test on the sign-up page. The details entered are verified from the data saved in the interviewee database. Upon successful verification, credentials are saved in the login database.

### 6.3 Secure login

After registration, the candidate is allowed to log in for an interview from the login page using his/her credentials. The data exchange between the client application and the server is encrypted using hashing. The user session is managed using a Login Manager. If the user has already logged in (even if he has skipped the test) he is disabled from taking the test again.

## 6.4 Dashboard

This page displays the instructions regarding the interview. The instructions would include the no. of sections, no. of questions, and the time limit. Other information as deemed necessary by the client and those required as per standard examination practices can also be included here.

## 6.5 Taking the interview

This is the page where questions from selected topics with answering area boxes are displayed. The whole page is divided into different sections according to topics. An equal number of questions are displayed for each topic. A candidate can choose to answer questions from any section at any time. Each topic contains questions of varying difficulty ranging from 1 to 3. Difficulty level 1 question holds 5 marks, level 2 holds 10 marks and level 3 holds 15 marks respectively.

The candidate has the option to answer a question as many times as he/she wants and save it. On clicking the save button Question, Answer, date & time of answering, and candidate id are saved in the candidate answers DB. There is a timer set for the interview and the interview automatically ends after the specified time.

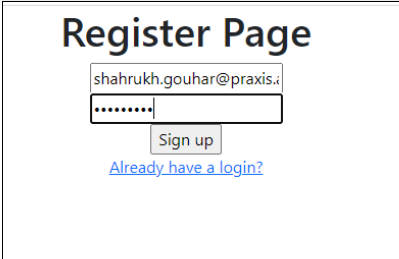
## 6.6 Storing the answers

The different answers against each question answered by the candidate are saved in a answers DB. This is fetched by the scoring program to score the candidate.

## 6.7 Hosting on AWS

The interview application needs to be publicly accessible to candidates. We deployed our web application on an AWS EC2 instance for public access.

Some snapshots of the candidate web application:



**Register Page**

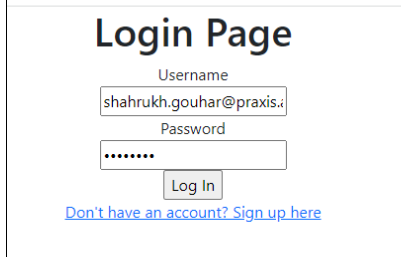
shahrukh.gouhar@praxis.ac.in

.....

Sign up

[Already have a login?](#)

Registration



**Login Page**

Username

shahrukh.gouhar@praxis.ac.in

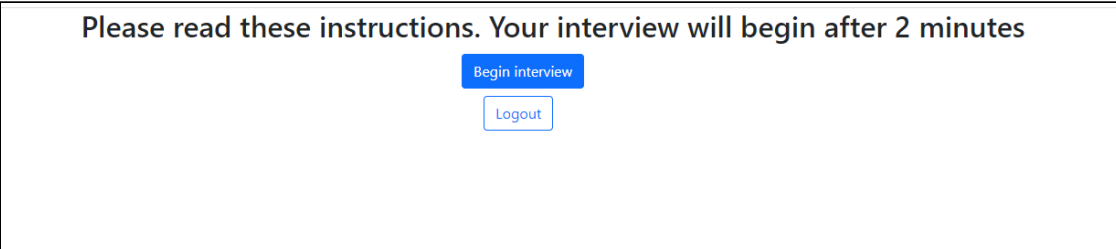
Password

.....

Log In

[Don't have an account? Sign up here](#)

Login

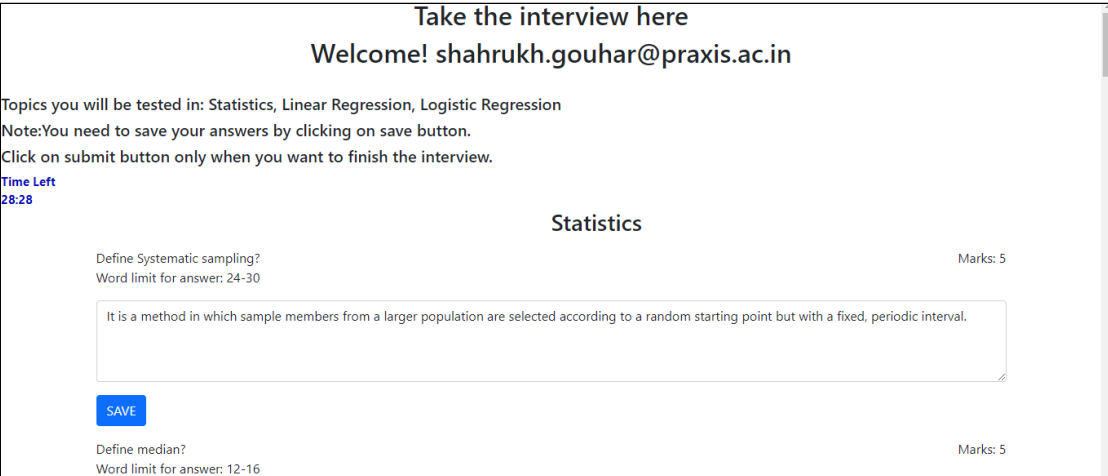


**Please read these instructions. Your interview will begin after 2 minutes**

Begin interview

Logout

### Dashboard



**Take the interview here**

**Welcome! shahrukh.gouhar@praxis.ac.in**

Topics you will be tested in: Statistics, Linear Regression, Logistic Regression

Note: You need to save your answers by clicking on save button.

Click on submit button only when you want to finish the interview.

**Time Left**  
28:28

**Statistics**

Define Systematic sampling? Marks: 5

Word limit for answer: 24-30

It is a method in which sample members from a larger population are selected according to a random starting point but with a fixed, periodic interval.

SAVE

Define median? Marks: 5

Word limit for answer: 12-16

Interview in progress



**Success**

Your responses have been captured

[Logout](#)

After the interview is submitted or it's timed out



# Chapter 7

## Score generator

### 7.1 Introduction

After the candidate has submitted the test, all the answers are saved in the DB. We are not comparing and scoring each saved answer instantaneously because the candidate can change their answers as many times as they wish during the test. Only the last saved answer for any question will be used for evaluation. Therefore, we have developed the Score generator module separately, which is scheduled to run daily. All the tests taken in a day will be evaluated and scored at the end of the day. Keeping the scoring module separately has enabled us to do the bulk operation at one go.

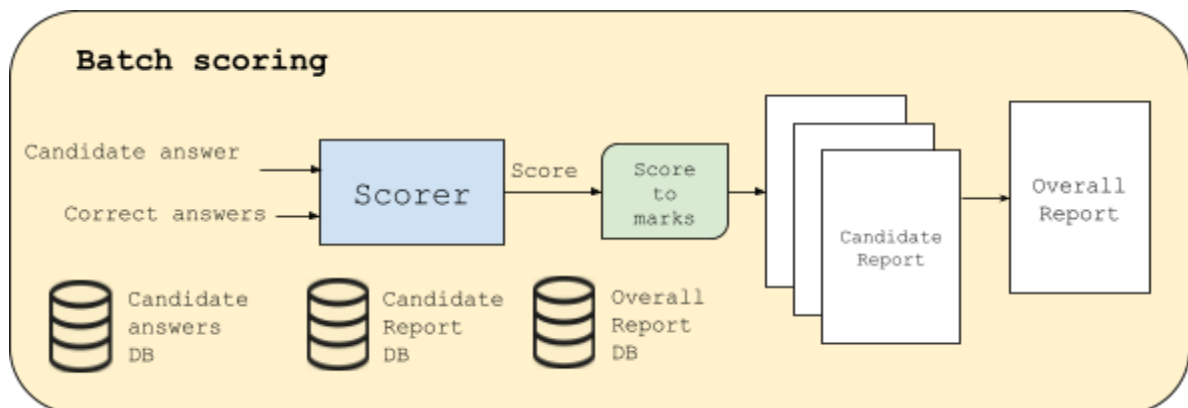


Fig. 7.1 Batch scoring program structure

## 7.2 Fetching the saved answers and preprocessing

The details from Candidate answers DB will be fetched to get questions, the user answers, saved date, user id. If a user has saved multiple answers for the same question, then only the latest answer will be fetched for evaluation. Scoring of these answers will be scheduled to run on a daily basis, preferably during night hours. Therefore, at the end of any given day, for all the candidates who took the test on that day, their scores will be generated and saved.

The data from Candidate answers DB will be merged with the master data. The resultant data frame will contain the question, user answer, saved date, user id, category of question, standard answer 1, standard answer 2, standard answer 3, and the maximum marks allotted for each question.

## 7.3 Scorer module

The Scorer function takes the user's answer and the standard answers to generate a score based on the level of similarity between the answers. If the user has not entered any answer or has entered random letters he is directly assigned a score of 0.

Otherwise, the user answer is compared with all three standard answers, and three similarity scores are generated respectively. The maximum of the three scores is taken as the final similarity score. Maximum of the three is taken to handle the cases when there is no standard answer 2 and 3, in which case, the 2nd, and 3rd similarity scores will be zero. Also, the answer needs to be very similar to only one answer. The similarity score is then scaled by the level of difficulty of the question to give the final score for that question.

## 7.4 Scoring single-word answers separately

While creating the dataset we realized that we have to take a different approach while handling Single-word answers. So the single word answer and multiple word answer questions are separated into two separate dataframes by using regex. For single word answer questions the user answer is compared with all the three answers present in the created dataset if they are matched then the score obtained is set as 1 else 0, the resultant dataframe is then appended with the multiple answer dataframe after calculating the score and final scorecard is generated at the end.

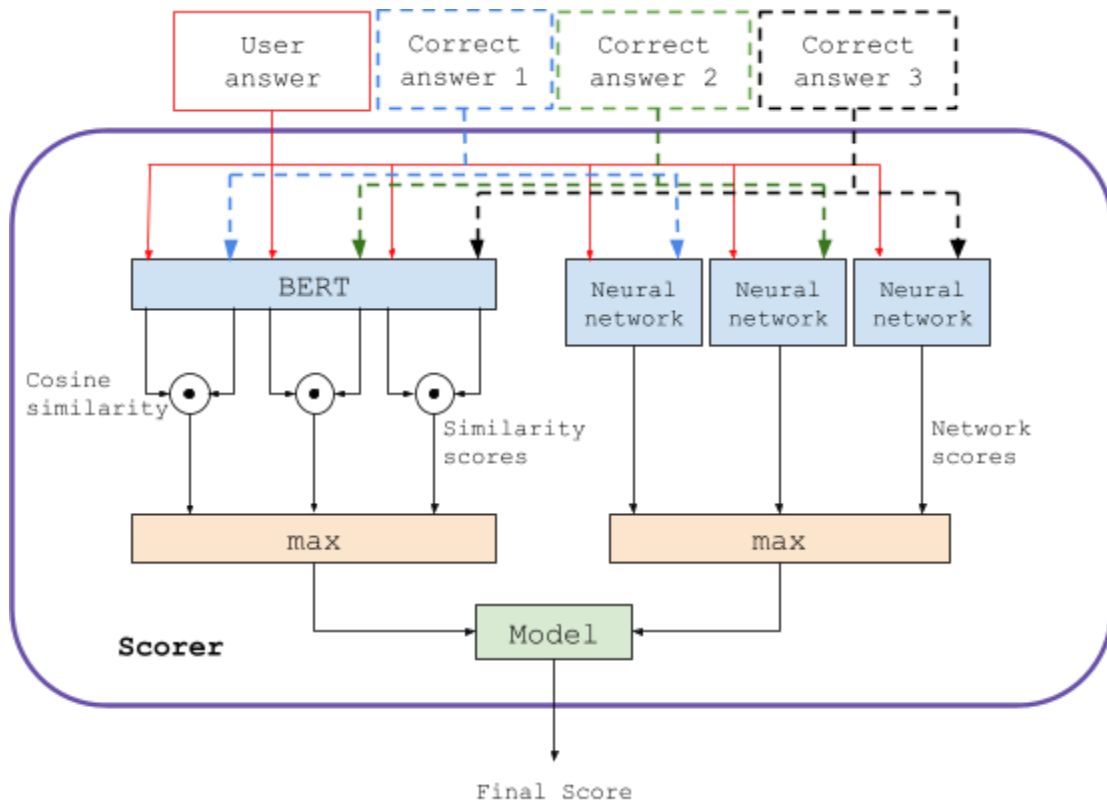


Fig. 7.2 Scorer module in detail

## 7.5 Generating the candidate report

The scores obtained for each question and data from Candidate answers DB is stored in Candidate Report DB. This table can be used to analyze a candidate's performance on various topics, different questions, and difficulty levels.

## 7.6 Generating the overall report

The table Overall Report DB is generated from the Candidate Report DB and contains user id, saved date, category, scores obtained, and maximum marks. It gives the total marks scored by the candidate in the test. Therefore, it can be used to compare and analyze the performance of different candidates.

# Chapter 8

## Conclusion and future work

### 8.1 Conclusion

Before starting this project, we had read extensively about the problems which could be solved using NLP and we had settled on this project. We tried many different techniques and learned a lot along the way about NLP. We have built a complete solution to automated interviewing using NLP. I am confident that enterprises who decide to use this can hire better and quicker.

### 8.2 Future work

- Use transformers instead of GRU for better attention
- Increase the topics in which questions are asked for more all-round interviewing
- Include sections for testing programming skills on SQL & Python
- Make the interview adaptive for more personalized interviewing experience
- Figure out ways to evaluate soft skills of the candidates
- Create environments to assess candidates on case studies
- Proctoring the exam
- Disabling the login if the user tries to login beyond the deadline
- Generate an interactive dashboard for the client to filter and select candidates
- Automatic scheduling the scoring program
- Downloadable reports for each candidate and overall results(within a certain time frame)

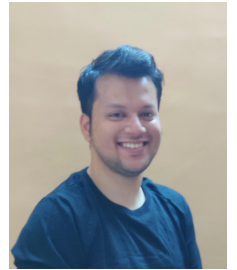
## *Bibliography*

- [1] [India TV - Massive supply gap in hiring Data Science talent in India](#)
- [2] [Neelanjana Mazumdar - 92% hiring managers face massive supply gap when hiring Data Science talent in India](#)
- [3] [HuggingFace - Models for Sentence Similarity](#)
- [4] [Google Research - BERT repository](#)
- [5] [sentence-transformers/SentenceTransformer](#)
- [6] [Jacob Devlin et. al :BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#)
- [7] [Junyoung Chung et. al :Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling in NIPS 2014](#)
- [8] [Keras: generating skipgram word pairs](#)
- [9] [Nauti Yogi: Word Embedding: Methods to generate them from scratch in Pytorch](#)

## *About the authors*

### [Anupam Misra](#)

Passionate about NLP and music



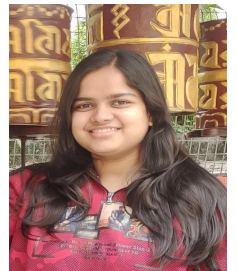
### [Shahrukh Gouhar](#)

Passionate about coding and mountaineering



### [Radha Rathore](#)

Passionate about solving real-world problems with data science knowledge and travelling



### [Mansoor Ali Shaik](#)

Passionate about world cinema and travelling

