

Interview Performance Evaluator

**CONTENTS**

1. Project objective……………………………………
2. Data Collection and Storage………………………………………
3. Data understanding…….………………………………………………….……….
4. Text Preprocessing…………….…………………………………
5. Model Building...………………………………….………………
6. Conclusion……….…………………………………………………………………
7. Deployment…………………………………………………………………………………………

**Project Objective**

This is a project based on automating the interviewer process. As Large number of Applicant attends Interviews. So, for evaluation of such large number of Applicant manually required hard efforts. Sometimes quality of evaluation may change according to mood of Interviewer. So, it will be very helpful if the process of evaluation of applicant is automated.

## Business Objective

**Minimize:**

* Minimize Human Intervention (Results in reducing biased decision making)
* Minimize work load of Hiring Managers
* Reduce time to hire
* Minimize the Evaluation Time for Descriptive Answers and reduce Human Intervention.
* minimizing recruitment process time and reduce the hr management cost
* minimize the problems faced by evaluation of subjective answers
* reducing time means minimizing the time consumption of interview event will get the result on spot
* minimize the problems faced by evaluation of subjective answers

**Maximize:**

* Easy evaluation of hiring process
* Maximize the Quality and Performance
* Maximize the productivity
* Maximize the Accuracy of Evaluation of the Descriptive Answers and its affiances to choose right Candidate
* Interview process smoother way and less time, Higher number of validations
* The efficiency for organizations by making the process seamless, secured, fast and convenient for candidates
* Classification of Profiles by Rank based on their performance
* Maximize the model accuracy to get suitable candidates
* How long a Task would take to complete. Estimating effort for the test is one of the major and important tasks in Test Management

## Constraints:

## Automated criteria can hurt diversity and good candidates may be thrown out of the race simply because they don't include certain keywords in their applications.

## Good for preliminary screening but not enough to make a final decision on hiring a particular candidate.

## SUCCESS CRITERIA

* Business success criteria: To reducing of cost and human interaction organization will take a process of hiring
* ML success criteria: To classify the applicants with basis on their performance on interview process with Bert model
* Economic success criteria: how many applicants are possible for the conducting of interviews will be easily to make decisions

**FEASIBILITY**

* On applicability of the ML technology, we faced the problems on automated evaluation process
* Automated criteria can hurt diversity and good candidates may be thrown out of the race simply because they don't include certain keywords in their applications.

Legal constraints are:

* Good for preliminary screening but not enough to make a final decision on hiring a particular candidate.
* Accurate data is difficult to collate and cannot fully rely on Machine learning model solutions.

## Data Collection and Storage

* For data storage we are using mongo dB database MongoDB is a document-oriented database which stores data in JSON-like documents with dynamic schema. It means you can store your records without worrying about the data structure such as the number of fields or types of fields to store values. It provides high performance, high availability, and easy scalability.
* MongoDB works on concept of collection and document. We are storing and extracting the data by using Pymongo .
* We can build the corpus according to the domain for which the interview is being conducted and that corpus can be stored and extract by using pymongo library in mongo dB database. In the same way we will store and extract Interview questions and applicant answer.

#EXTRACTING THE CORPUS FROM MONGO DB

import pymongo

client = pymongo.MongoClient("mongodb://localhost:27017/")

# Database Name

db = client["test\_database"]

# Collection Name

col = db["corpus"]

x = col.find()

for string in x:

    print(string)

string=string['contents']

## Document Organization

This document is organized into the following section:

|  |  |
| --- | --- |
| Introduction | Provides information related to the document |
| Data Understanding | Data understanding is the knowledge that you have about the data, the needs that the data will satisfy, its content and location. |
| Text Preprocessing | The various text preprocessing steps are: Tokenization. Lower casing, stemming, remove special character, Stop words removal. |
| Model Building | Build a model to Automate the interview process |
| Performance evaluation and Deployment | A major purpose of a performance evaluation is to acknowledge and encourage high quality performance. Deployment is to form an app to Automate the interview process |

# Data Understanding

## Context

Here we have used data science domain for building a corpus for training the model, Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from noisy, structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains. Data science is related to data mining, machine learning and big data.

## Feature selection

Feature selection methods to shorten the long text and then use the shortened text as the input to the Natural Language Processing model. For this we remove stop words, we do tokenization and embedding (token, interval segmentation, position)

**Text Preprocessing**

* To Improve Accuracy the Noise Removal should be done which can be done by Text Normalization by Stemming, Lemmatization and also removal of Special characters, tokenization etc. based on the requirement.
* Here we are using Regular Expression library for text preprocessing (removal of special character, tokenization etc.)

 import re

            context=string.lower().strip()

            context = re.sub('[^a-zA-Z0-9\s\.]', ' ', string)

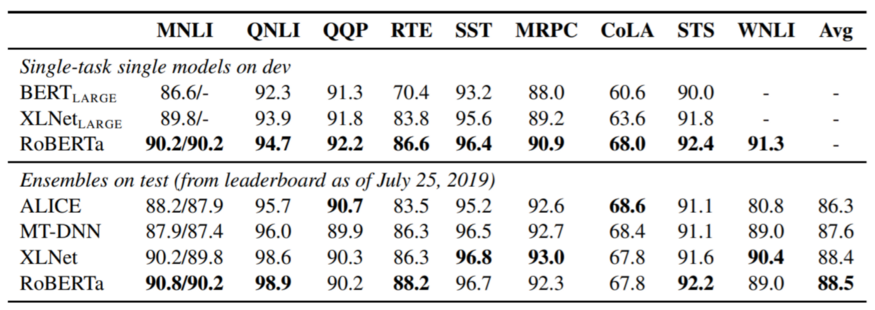
            context = ' '.join(context.split())

**Model Building**

* Here we are using Roberta-large pretrained model for answering evaluator question and sentence Bert pretrained model for word embedding of model answer and applicant answer and lastly used cosine similarity for checking similarity between evaluator question and applicant answer.
* Roberta-large pretrained model which can be easily downloaded from hugging face transformers has 12- Transformer layer, 768-hidden layers, 12-multiheads attention, 340M parameters.
* Importantly, Roberta uses 160 GB of text for pre-training, including 16GB of Books Corpus and English Wikipedia used in BERT. The additional data included [Common Crawl News dataset](http://web.archive.org/web/20190904092526/http:/commoncrawl.org/2016/10/news-dataset-available/)(63 million articles, 76 GB), Web text corpus (38 GB) and Stories from Common Crawl (31 GB). Robert is just a BERT acquainted with dynamic masking, and without next sentence prediction.
* modelname ='deepset/xlm-roberta-large-squad2'
* from transformers import AutoTokenizer, AutoModelForQuestionAnswering
* model =AutoModelForQuestionAnswering.from\_pretrained(modelname)
* tokenizer = AutoTokenizer.from\_pretrained(modelname)
* Then we use sentence transformer for converting the model answer and Applicant answer into a vector form, Sentence Transformers is a Python framework for state-of-the-art sentence, text and image embeddings. This can be useful for semantic textual similar, semantic search, or paraphrase mining. The framework is based on Torch and Transformers and offers a large collection of pre-trained models tuned for various tasks.
* Lastly, we are using cosine similarity to check the similarity between model answer and Applicant answer. Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis.
* from sentence\_transformers import SentenceTransformer
* from sklearn.metrics.pairwise import cosine\_similarity
* model = SentenceTransformer('paraphrase-distilroberta-base-v2')
* result=[]
* score=[]
* for i in range(num):
* sentences=[model\_answer[i],applicant\_answer[i]]
* sentence\_embeddings = model.encode(sentences)
* scores=cosine\_similarity([sentence\_embeddings[0]],sentence\_embeddings[1:])\*100
* if scores>35:
* results='correct answer'
* else:
* results='incorrect answer'
* result.append(results)
* score.append(scores)
* result=pd.Series(score)
* scored=result.loc[:].mean()
* If the score is greater than 30% (threshold value) then the applicant has cleared the interview else not cleared, this threshold value can be determined according to the client requirement

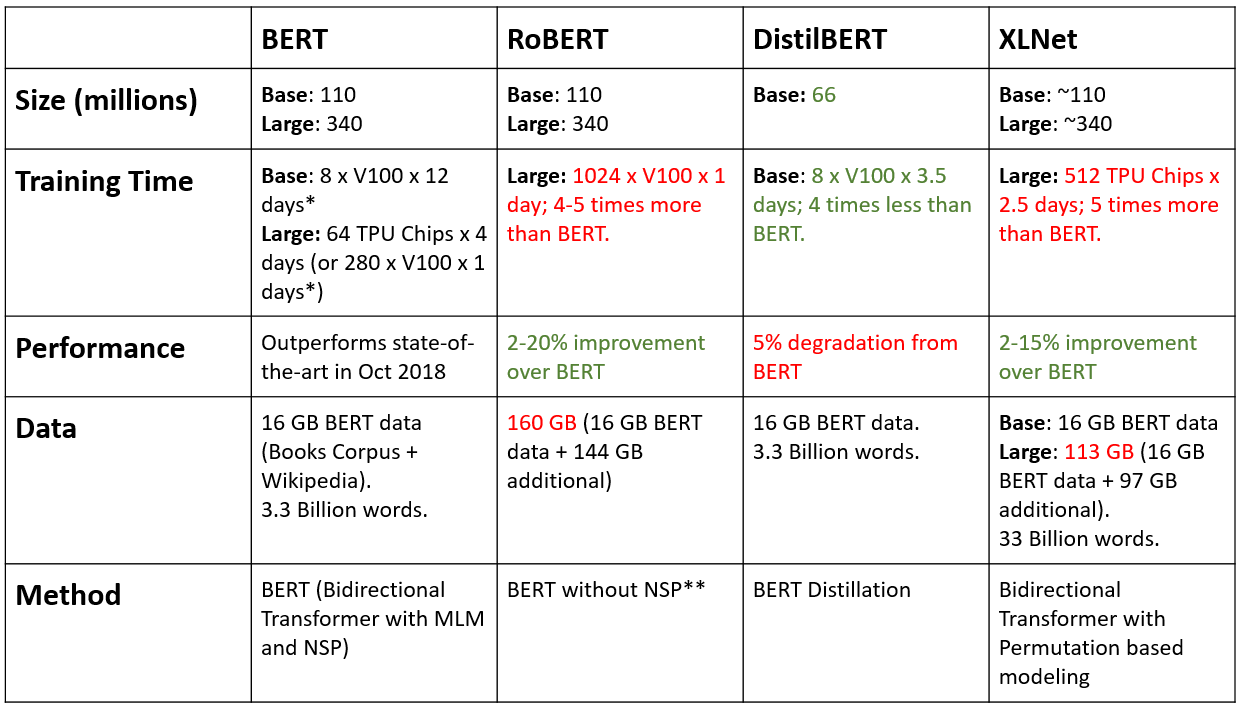
**Model Performance and Evaluation**

* To evaluate performance, Roberta Large pretrained has been compared to other state-of-the-art NLP systems. Importantly, Roberta Large achieved all of its results with almost no task-specific changes to the neural network architecture.
* On the GLUE benchmark NLP tasks, the model achieves a score of 88.5 on the public leader board and achieve the state-of-the-art score on 4 of GLUE tasks: Multi Natural Language Inference (MNLI), QuestionNLI, Semantic Textual Similarity Benchmark (STS-B), and Recognizing Textual Entailments (RTE) at the time of its release. As a result, Roberta outperforms all BERT model on GLUE benchmark results:



* The General Language Understanding Evaluation (GLUE) benchmark is a collection of resources for training, evaluating, and analyzing natural language understanding systems. ... A public leaderboard for tracking performance on the benchmark and a dashboard for visualizing the performance of models on the diagnostic set.

**Comparison of Model Performance**

****

**Conclusion**

* Roberta is undoubtedly a breakthrough in the use of Machine Learning for Natural Language Processing. It can be used for a wide range of practical applications and downstream tasks. Here we Have used Roberta and sentence Bert model to find the Performance of an Applicant in an interview.
* This Can really help the companies also to find the best suited candidate for the given job role with reduced time and manpower which will indirectly help in the growth of the company.

**Deployment**

We have used flask and pywebio for deployment. PyWebIO provides a series of imperative functions to obtain user input and output on the browser, turning the browser into a "rich text terminal", and can be used to build simple web applications or browser-based GUI applications without the need to have knowledge of HTML and JS. PyWebIO can also be easily integrated into existing Web services.

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

**Installations**

Stable version:

pip install pywebio

When using the Flask as PyWebIO backend server, you need to install Flask by yourself and make sure the version is not less than 0.10. You can install it with the following command:

pip install -U flask >=0.10

**Deployment Implementation**

* we will import the webio\_view library to Get the view function for running PyWebIO applications in Flask. The view communicates with the browser by HTTP protocol.

from pywebio.platform.flask import webio\_view

* The path of the static file of PyWebIO is stored in pywebio.STATIC\_PATH

from pywebio import STATIC\_PATH

* Send a file from a given directory with send\_file(). This is a secure way to quickly expose static files from an upload folder or something similar. And for rendering html files we are using render\_template

from flask import Flask, send\_from\_directory,render\_template

* For storing and extracting data from mongo dB we are using the pymongo library.

from pymongo import MongoClient

* This module provides functions to get all kinds of input of user from the browser

from pywebio.input import \*

* This module provides functions to output all kinds of content to the user’s browser, and supply flexible output control.

from pywebio.output import \*

* configure the environment of the current session.

from pywebio.session import set\_env

* Flask constructor takes the name of the current module (\_\_name\_\_) as argument.

app=Flask(\_\_name\_\_)

* The route() function of the Flask class is a decorator, which tells the application which URL should call the associated function.

@app.route("/")

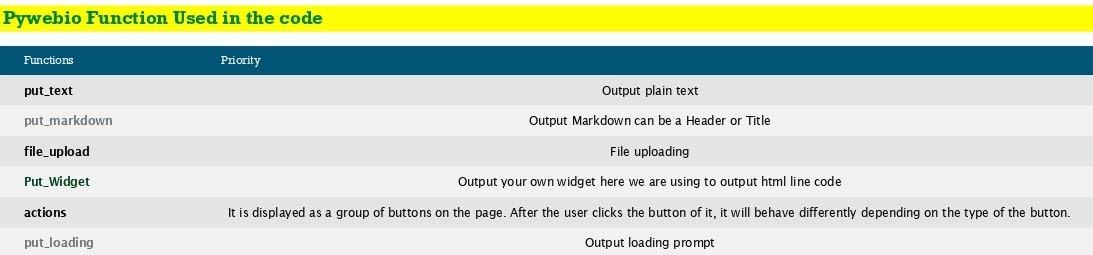
* ‘/’ URL is bound with hello\_world() function which returns the rendered Html template for making a homepage for the app.

For Rendering Html template we need to have a folder called templates in the same directory from where the python deployment code is saved  and inside that an Html file to render .

def hello\_world():

    return render\_template('inner.html')

* Then we will define a function called model() where we will give different inputs and file upload function and lastly, we will run our transformer model to generate an output



* we call our function for this we need to add a rule so in app we have a function called as add\_url\_rule so here I am going to use a url like ‘/tool’ I have to give second parameter which is ‘webio\_view’ and this web\_view which I have imported previously in this I just need to pass our function called model and this will do entire execution

app.add\_url\_rule('/tool','webio\_view',webio\_view(model),

methods=['GET','POST','OPTIONS'])

* Then we will use app.run to run our app on a local server you have to give the port number.

app.run(host='127.0.0.1',port=5000)

* Lastly our app is Running on <http://127.0.0.1:5000/>

**Conclusion**

Pywebio is a beautiful library which allows us to showcase our code on web as if we are just writing on terminal and its getting displayed on to the web For more understanding you can go to the below link:

[PyWebIO — PyWebIO 1.4.0 documentation](https://pywebio.readthedocs.io/en/latest/)