**Trans Hire Documentation:**

In this document, we are providing some details about what we have achieved till now, how it is done and what can be the pending or might be implemented in future in order to enhance the accuracy.

As we Initially discussed we have divided the Trans hire project into different requirements which are given below:

1. Extracting Important keywords from Resume

2. Candidate Ranking based on extracted keywords from Resume and Identifying top candidates according to JD

3. AI Powered Chatbots for conducting psychometric or aptitude test

4. Voice Powered Bot for conducting Normal Screening with candidates

5. Communication skills checker during Interview calls

6. Emotion Analyser to check overall emotion of candidates in a call

7. Interview answers analyser during interview call

8. Social Networking Profile Matcher with candidate resume

**Requirements Progress Details:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No.** | **Name of Requirement** | **Approach Used** | **Status** | **Main codes Completed** | **API created** | **Testing done** | **Completed By** |
| **1** | Extracting Important keywords from Resume | **Dictionary and Rule based** | **Completed** | **Yes** | **Yes** | **Yes on 50 resume** | **Karthick** |
| **2** | Candidate Ranking based on extracted keywords from Resume and Identifying top candidates accroding to JD | **Dictionary and Rule based** | **Completed** | **Yes** | **Yes** | **Pending** | **Karthick** |
| **3** | AI Powered Chatbots for conducting pyschometric or apptitude test | **Not finalized** | **Not started yet** | **NA** | **NA** | **NA** | **NA** |
| **4** | Voice Powered Bot for conducting Normal Screening with candidates | **Not finalized** | **Not started yet** | **NA** | **NA** | **NA** | **NA** |
| **5** | Communication skills checker during Interview calls | **Not finalized** | **Not started yet** | **NA** | **NA** | **NA** | **NA** |
| **6** | Emotion Analyser to check overall emotion of candidates in a call | **Not finalized** | **Not started yet** | **NA** | **NA** | **NA** | **NA** |
| **7** | Interview answers analyser during interview call | **Not finalized** | **Not started yet** | **NA** | **NA** | **NA** | **NA** |
| **8** | Social Networking Profile Matcher with candidate resume | **Not finalized** | **Not started yet** | **NA** | **NA** | **NA** | **NA** |

**Details about Completed Requirements till date:**

1. Extracting Important keywords from Resume

2. Candidate Ranking based on extracted keywords from Resume and Identifying top candidates accroding to JD

**Approaches and steps used to complete the Requirements:**

**1. R1:** Extracting Important keywords from Resume

**Approach used:**

Rule Based and Dictionary based approaches

**Steps used to complete the R1:**

Step1: First we read the resume in .docx and .pdf format. For reading the resume in the prescribed format we have used the following libraries in python that can read pdf and docx files and convert into simple string format.

* pdfminer.six (pip install pdfminer.six) for reading pdf file
* doc2txt (pip install doc2txt==0.8) # For reading docx file

Step2: Then we pre-process the converted text and remove some unwanted noise such as punctuations, special symbols, emojis and all.

Step3: We have created some dictionary to identify the entities from the text such as dictionary to identify skills, education details, certifications etc. and we have set some rules based approach to identify the entities such as university name, company name, email id, contact no, work experience, external links etc.

Step4: we have also included the self learning module that can learn extra entities if found in resume and we have appended those in spate csv file to collect some data.

Step5: After that we have build some function that read the whole text from resume and use those rules and dictionary to extract relevant entities from the resume.

**Accuracy Acheived:**

We have tested our R1 part on 50 resume as well. and we have achieved overall 83% accuracy as an average for all entities.

**Accuracy achieved for each entity during testing:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Name of Entity** | **Accuracy achieved on 50 resumes** |
| 1 | Phone no | 90% |
| 2 | Email id | 96% |
| 3 | Skills | 86% |
| 4 | Education | 86% |
| 5 | Certification | 88% |
| 6 | External links | 84% |
| 7 | Companies | 70% |
| 8 | University | 80% |
| 9 | Personal Name | 70% |
| 10 | Experience | 68% |

**Steps to Run API for resume extraction:**

## Prerequisites:

* Python
* Virtualenv

**Step 1:** Flask Installation and Server Setup

We are assuming that you have already installed Python, and it’s up to date. So let’s set up our project and set up a virtual environment.

**Step2:** Install Virtual environment in local system for this use the following commands:

pip3 install virtualenv

mkdir flask-test

cd flask-test

virtualenv .

source bin/activate

Python3 -m pip install Flask

So far, we have created our project folder, installed and created a virtual environment for our

**Step3:** Now in your project directory where all virtual environment and your API codes are there. Please use the following commands:

*pip install –r requirements.txt*

It will install all the necessary packages in your system.

**Step4:** now run the following command

*flask app.py*

it will run the server now copy and paste the below URL:

\* Environment: production

WARNING: This is a development server. Do not use it in a production deployment.

Use a production WSGI server instead.

\* Debug mode: off

\* Running on <http://127.0.0.1:5000/> (Press CTRL+C to quit)

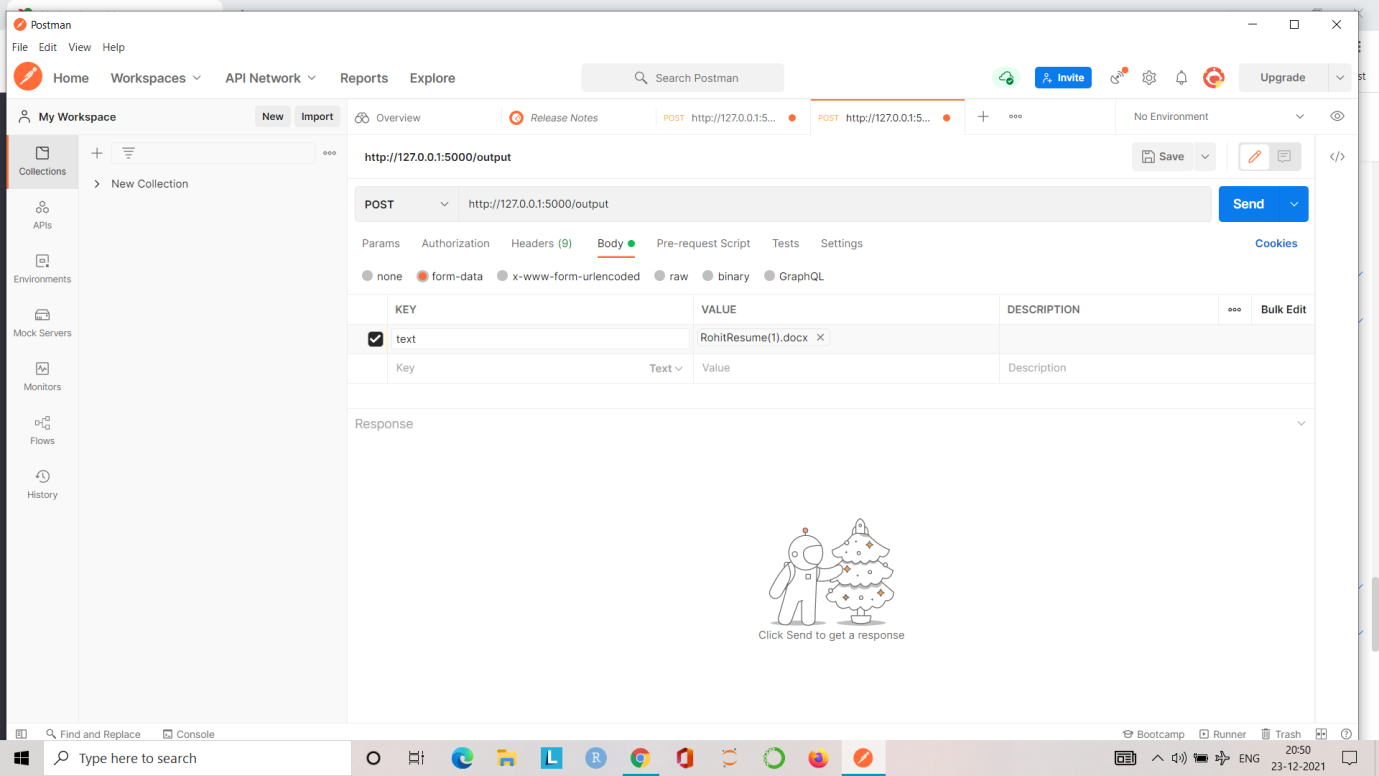
**Step5:** Now open the POSTMAN and paste the copied URL from command prompt,

and set the following parameters to test the API

**Parameters Names**

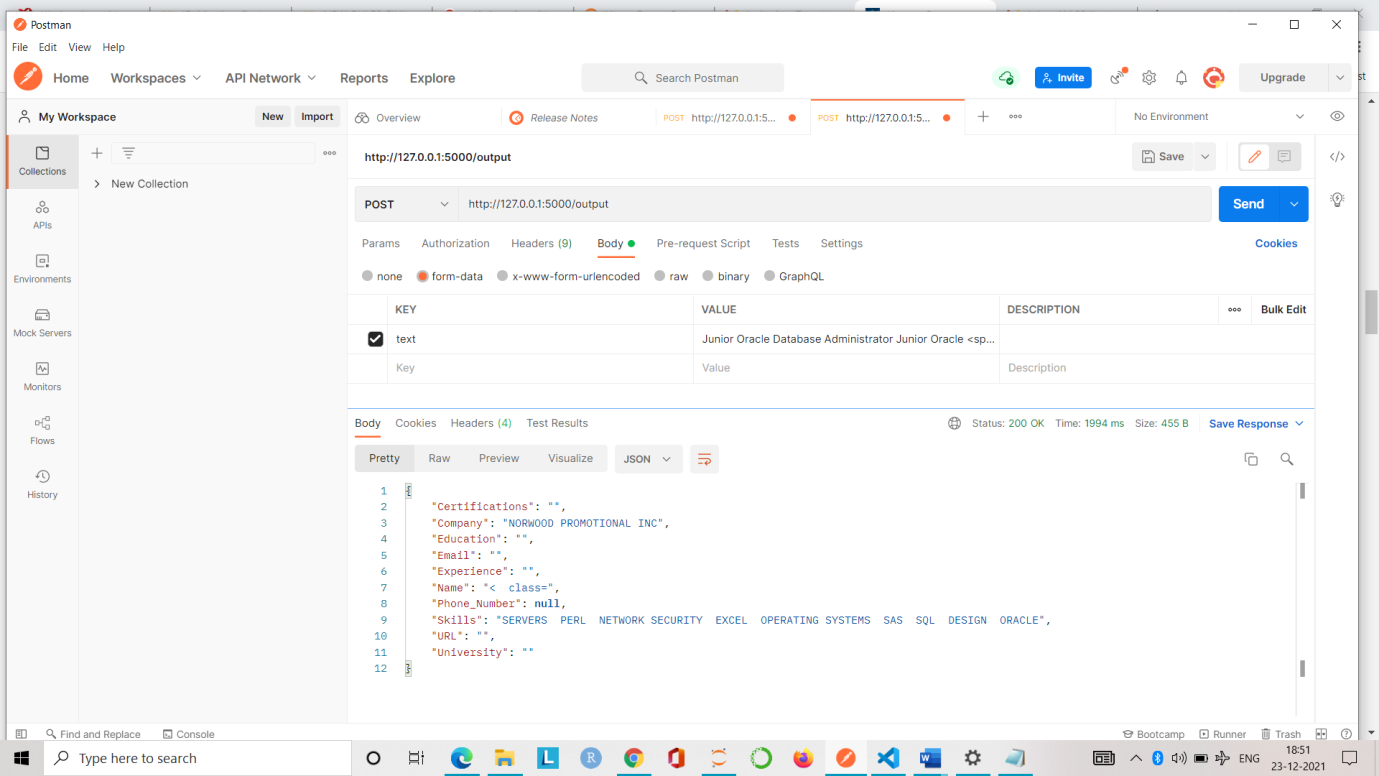
text type:file

The output should look like that,



**Step 5:** After setting the URL and parameters in the postman, Now upload the resume file from your local directory and click on the Send button

**Step 6:** After that you will see the output as given below:



**Future Task:**

To improve the entity recognition system using Deep Learning based approach if we got the accuracy better than this.

**2. R2: Candidate Ranking based on extracted keywords from Resume and Identifying top candidates according to JD**

**Approach Used:**

Rule based and dictionary based

**Steps used to complete R2:**

Step1: First we read resume and Job description JD in .docx or pdf format using the following libraries in python that can read pdf and docx files and convert into simple string format.

* pdfminer. Six (pip install pdfminer. Six) for reading pdf file
* doc2txt (pip install doc2txt==0.8) # For reading docx file

Step2: We use dictionary based approach to extracted following information from JD such as:

a. Overall experience required for particular job position

b. Skills mentioned in JD

c. Decision words such as Minimum, Maximum, having, Good to have and Good knowledge to build rules for matching scores.

d. Experience associated with each skill if mentioned in resume

Step3: We use dictionary and rule-based approach to extracted the following information from resume in order to match with JD. The information is given below:

a. All skills mentioned in resume

b. Exp. with each skill if it is mentioned in resume

c. Overall experience with candidate.

Step4: Then we have built some rules that are designed by experts in order to match the extracted data of JD with the resume to calculate the matching score.

Step5: Then we use matching algorithm that apply those rules on the matching score in order to give the final matching score and matching scores for all skills in a dataframe.

**Accuracy Achieved:**

**Steps to run API:**

## Prerequisites:

* Python
* Virtualenv

**Step 1:** Flask Installation and Server Setup

We are assuming that you have already installed Python, and it’s up to date. So let’s set up our project and set up a virtual environment.

**Step2:** Install Virtual environment in local system for this use the following commands:

pip3 install virtualenv

mkdir flask-test

cd flask-test

virtualenv .

source bin/activate

Python3 -m pip install Flask

So far, we have created our project folder, installed and created a virtual environment for our

**Step3:** Now in your project directory where all virtual environment and your API codes are there. Please use the following commands:

*pip install –r requirements.txt*

It will install all the necessary packages in your system.

**Step4:** now run the following command

*flask app.py*

it will run the server now copy and paste the below URL:

\* Environment: production

WARNING: This is a development server. Do not use it in a production deployment.

Use a production WSGI server instead.

\* Debug mode: off

\* Running on <http://127.0.0.1:5000/> (Press CTRL+C to quit)

**Step5:** Now open the POSTMAN and paste the copied URL from command prompt,

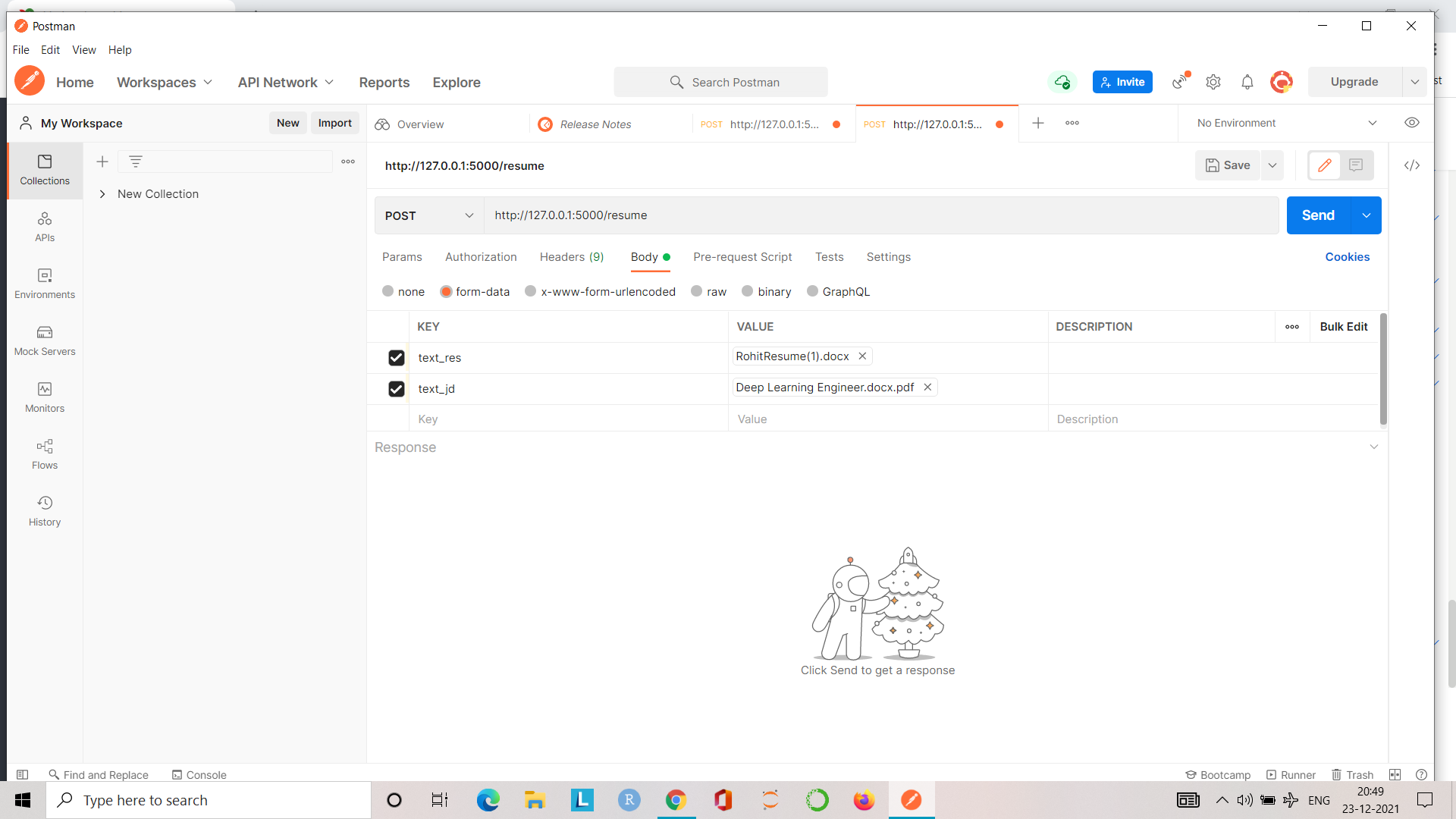
and set the following parameters to test the API

**Parameters Names**

1. text\_res type:file

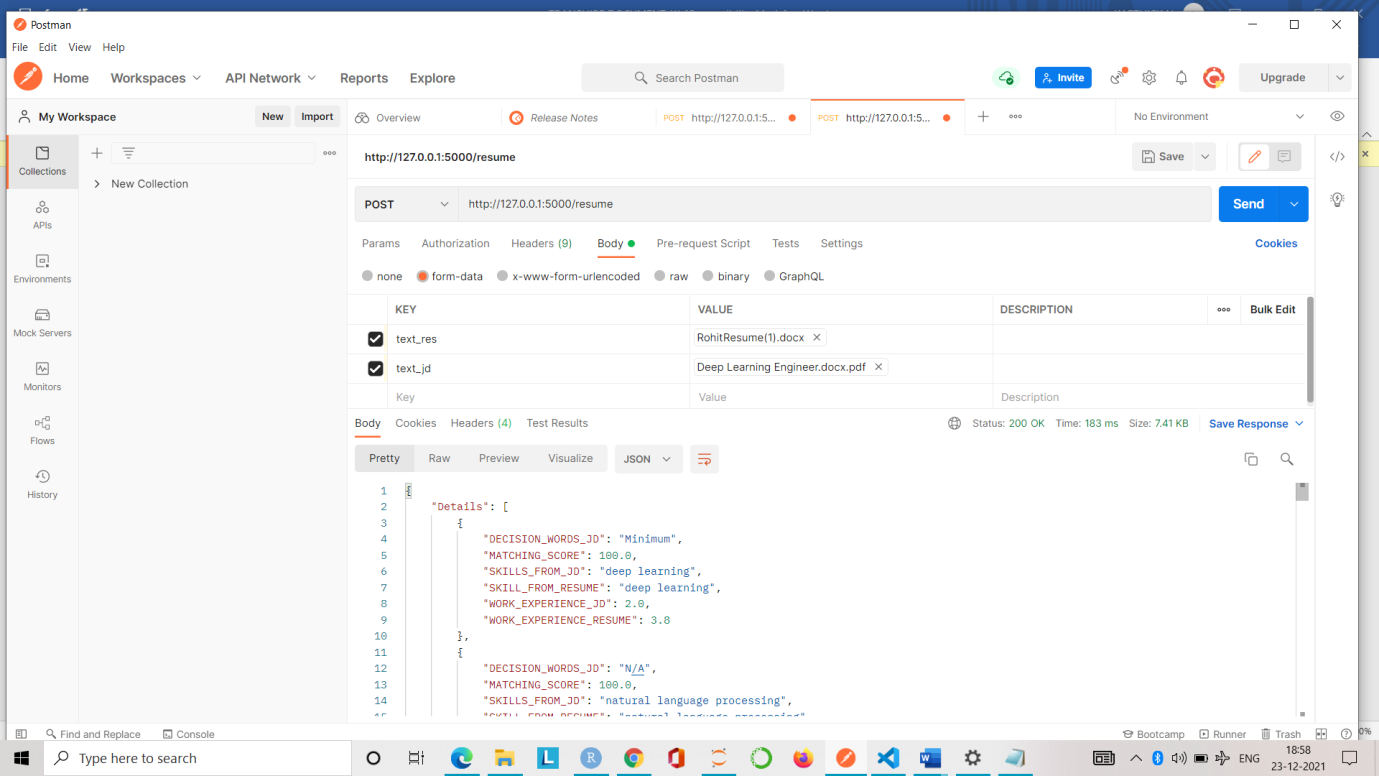
2. text\_jd type:file

The output should look like that,



**Step 6:** After setting the URL and parameters in the postman, Now upload the resume file and JD file from your local directory and click on the Send button

**Step 7:** After that you will see the output as given below:



**Future Scope:**

Till now we have used dictionary based and rule based. In future we will use deep learning based approaches to enhance this accuracy.