

TEAM NO-365

SMARTBRIDGE EXTENSION PROGRAM
(ARTIFICIAL INTELLIGENCE)

AI-Enabled Candidate Resume Screening Using
Spacy Entity Recognition

ATHIYA M(20BKT0042)

BARATH KUMAR S(20BKT0019)

ATHARV MAHESH GOTE(20BCT0151)

VANSH KARNWAL(20BCT0182)

VIT- VELLORE

1.INTRODUCTION

1.1Overview

The project aims to screen candidate resumes with the job descriptions. The candidate can choose the job to apply for, each job has its own set of required skills. The resume from each candidate is processed using packages that use the Spacy NER (Named Entity Recognition) model which is used to identify named entities by natural language processing. With the help of this the resume is parsed to extract the candidate's skills, and other details like name and email. The skills from here are compared with the respective job descriptions and if they cross a similarity threshold he/she is accepted else they are rejected. An email is sent to the candidate regarding the result. Flask is used to take care of setting up the application.

1.2Purpose

This project is useful to effectively screen candidates, a huge number of people apply for jobs every day and companies have to screen through candidates in an effective manner. Doing this manually is very cumbersome and inaccurate. Setting up automatic systems that can automatically screen candidate resumes by comparing with the job descriptions using NLP (Natural Language Processing) Techniques, this can greatly increase the time taken and the accuracy of evaluating the candidates for respective job roles. Skill matching is also done better as the NLP techniques can efficiently match skills in the candidate resume to the skills in the job descriptions. This also creates a transparent and unbiased way to land jobs that can further motivate applicants.

2 LITERATURE SURVEY

2.1 Existing problem

In the current system, resume screening is done by a combination of manual and automated processes. Recruiters perform manual reviews on the candidate's resume by analyzing it for the skills and requirements and assessing whether or not the candidate is qualified for the job, although this can be a comprehensive way of screening the candidates, it is not time-efficient or energy-efficient.

Several other companies use ATS (Application Tracking System) which is software that has some basic screening capabilities such as years of experience, education background etc. However, these systems cannot handle unstructured data which may result in not accurately accessing the candidate, they may inadvertently have some biases due to training on a particular demographic or universities.

Some other methods of scanning include screening questions that are used to screen or filter out candidates, although the accuracy of these questions in accessing a candidate's skills is not very good.

2.2 Proposed solution

The proposed solution is to completely automate these processes of resume screening by allowing

candidates to upload their resume in applications, and screening them by using NLP techniques in the backend.

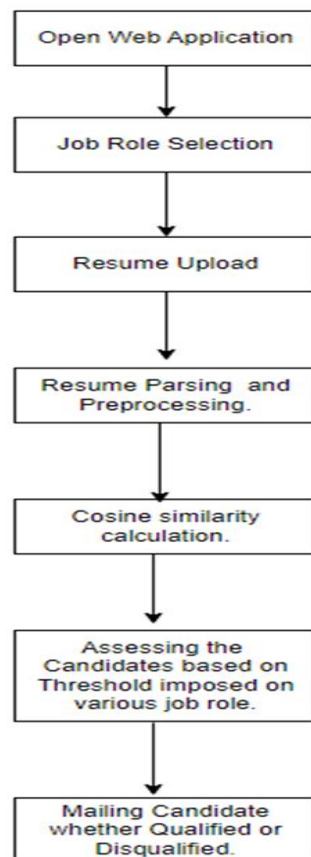
This can be done effectively by using NER (Named entity Recognition) from Spacy and PyresParser which are useful python packages that can be used to screen resumes to extract useful valuable information from them. This information can then be compared with the job descriptions and a similarity coefficient can be established. The candidature is then accepted or rejected based on this similarity coefficient.

This can greatly increase the efficiency and accuracy of the candidate resume screening process.

3 THEORITICAL ANALYSIS

3.1 Block Diagram

AI Enabled Candidate Resume Screening



3.2 Hardware or Software Designing

Hardware Requirements:

CPU:

1. Intel Core i7-11700K:
 - 8 cores and 16 threads.
 - Base clock speed of 3.6GHz and boost clock speed of 5.0GHz.
2. AMD Ryzen 7 5800X:
 - 8 cores and 16 threads.
 - Base clock speed of 3.8GHz and boost clock speed of 4.7GHz.

RAM:

1. DDR4 RAM 16GB (2x8GB) 3200MHz:
 - Dual-channel kit for optimized performance.
 - Operating frequency of 3200MHz.
2. DDR4 RAM 32GB (2x16GB) 3600MHz:
 - Dual-channel kit for optimized performance.
 - Operating frequency of 3600MHz.

Storage:

1. SSD:

- High-speed NVMe PCIe interface.
- Read speeds up to 3,500MB/s and write speeds up to 3,300MB/s.

2. HDD:

- Large storage capacity for resume storage.
- Rotational speed of 7200 RPM for faster data access.

GPU (optional):

1. NVIDIA GeForce RTX 3070:

- 5888 CUDA cores.

- 8GB GDDR6 VRAM.
- Boost clock speed of 1730MHz.

2. AMD Radeon RX 6800:

- 3840 stream processors.
- 16GB GDDR6 VRAM.
- Boost clock speed of 2105MHz.

Network:

1. Ethernet: TP-Link Archer AX6000 Wi-Fi 6 Gigabit Router:

- Wi-Fi 6 (802.11ax) standard for improved network efficiency.
- Multi-Gigabit Ethernet ports for high-speed connectivity.

2. Internet Service Provider (ISP) Plan: 500 Mbps Downlink and 50 Mbps Uplink:

- Sufficient bandwidth for handling incoming resumes and processing them efficiently.
- Consider higher speeds for larger-scale deployments or if handling significant data traffic.

Software Requirements:

1. Operating System:

- Windows
- Ubuntu 20.04 LTS (Long-Term Support)
- CentOS 8
- macOS

2. Programming Language:

- Python 3.9 or later

3. Web Framework :

- Flask (python)

4. Cosine Similarity Library:

- SciPy 1.7 or later (for `cosine_similarity` function)
- scikit-learn 0.24 or later (for `cosine_similarity` function)

5. Text Processing Libraries:

- NLTK 3.6 or later
- SpaCy 2.3.9
- pyresparser

6. Mail Notification:

- smtplib in python by enabling an SMTP

7. Developing Environment:

- pycharm IDE

4 EXPERIMENTAL INVESTIGATIONS

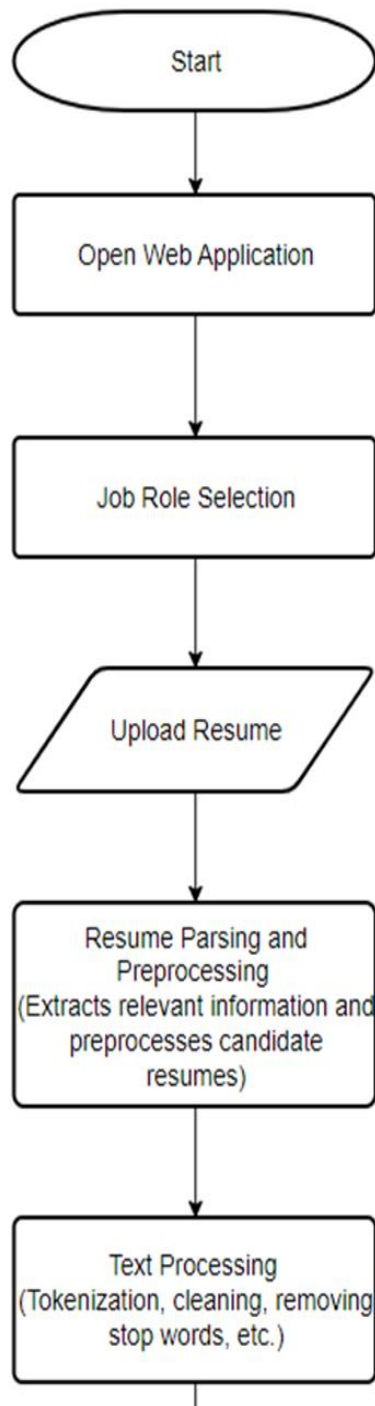
While working on the project, we investigated different ways to find and compare the data extracted from the resume with the job description.

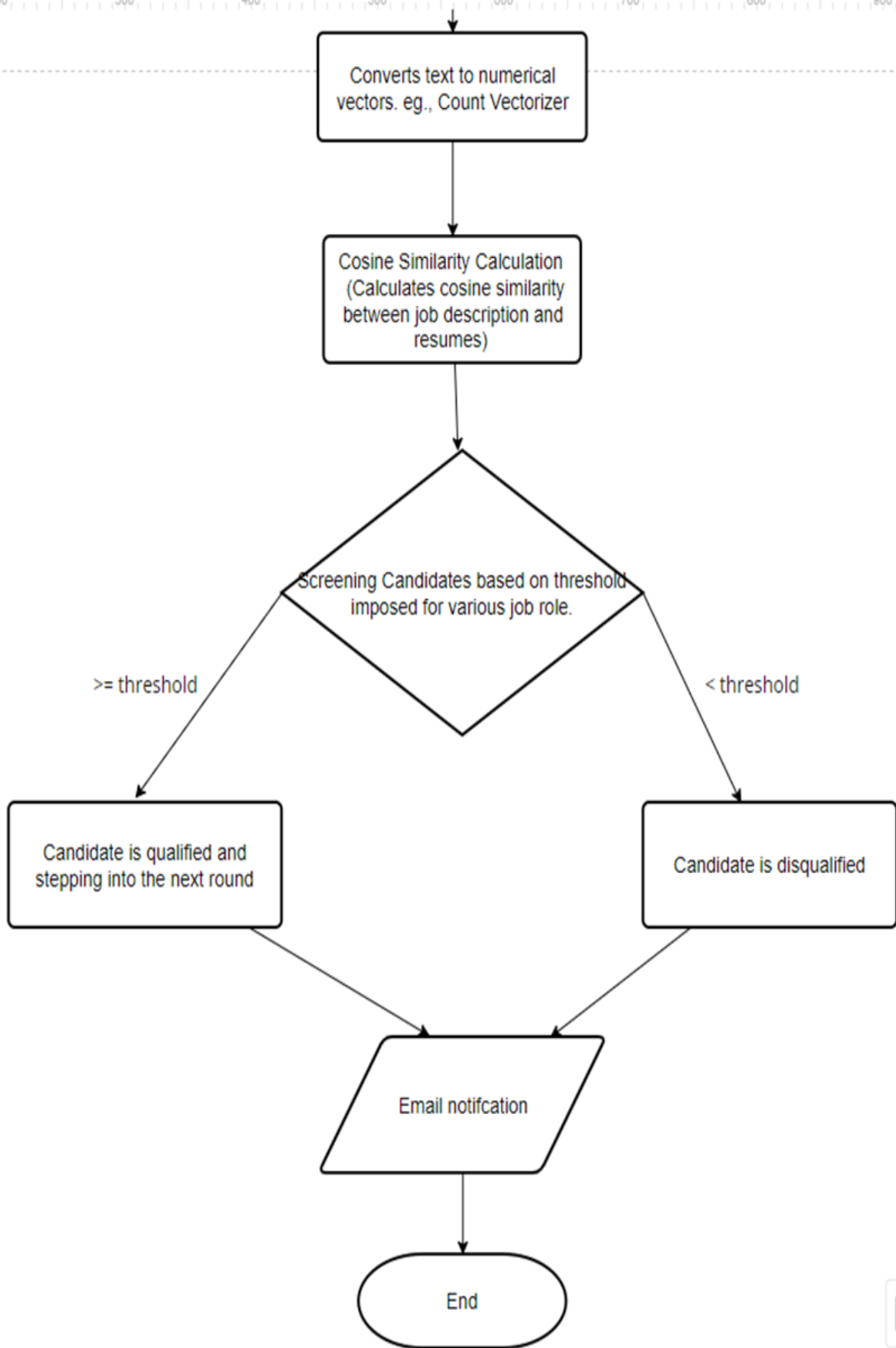
A simple way would be to extract the skills and compare and proceed depending on how many skills were matched, a more effective way would be to vectorize using NLP techniques and compare using cosine similarity to find the similarity between the candidate resume and job description. This would be more accurate and better at comparing. When using this, different job roles can have different similarity ratings, so we can set up similarity thresholds for candidates to be accepted.

Another improvement can be to directly compare the job description requiring a list of skills, the job description can be processed using NLP techniques and directly compared with information extracted from the candidate's resume.

5 FLOWCHART

AI Enabled Candidate Resume Screening





6 RESULT


HOME [APPLY NOW](#)

The Smart **Resume**
Analyser
Fastest Way To **Rate Your**
Resume Instantly



HOME


AVAILABLE JOBS



SDE JAVA
DEVELOPER IBM
BANGALORE
FULL TIME

[1](#) [2](#) [3](#) [4](#)


Apply Now



SDE ML ENGINEER
ADOBE
HYDERABAD
FULL TIME

[1](#) [2](#) [3](#) [4](#)

Apply Now



SALESFORCE
DEVELOPER
GOOGLE
DELHI
FULL TIME

[1](#) [2](#) [3](#) [4](#)

Apply Now

127.0.0.1:5000

FILL THE APPLICATION

NAME:

EMAIL:

UPLOAD RESUME

 No File Selected.

Predicted Rating Is.

Predicted Rating Is. 5.99

Thanks For Applying You Will Be Emailed About Your
CandidatureInterview Call Inbox x

projectpurposes0@gmail.com

to ▼

Hello NADIA DELGADO

Thanks for applying to the job post. Your Skills match our requirement. Kindly let us know your availability for next round of interview.

Thanks and regards,

Talent acquisition team.

7 ADVANTAGES & DISADVANTAGES

Advantages of the Resume Analyzer Project:

- **Time and Cost Efficiency:** By automating the resume screening process, the project significantly reduces the time and effort required to review and evaluate resumes. This enables recruiters to focus more on interviewing potential candidates and other strategic tasks, thereby increasing overall productivity and efficiency. Additionally, it saves costs associated with manual screening or outsourcing.
- **Enhanced Accuracy and Consistency:** The AI and NLP algorithms used in the project ensure consistent and objective evaluation of resumes. Unlike humans, the system doesn't get fatigued or biased, leading to fair and unbiased screening results. This helps in identifying the most qualified candidates based on predefined criteria, leading to more accurate shortlisting.
- **Scalability:** The project allows for seamless scalability, enabling it to handle a large volume of resumes without any compromise in performance. This is particularly advantageous for organizations receiving a high influx of applications for job openings.
- **Customization and Adaptability:** The project's flexibility allows employers to define their own evaluation criteria, tailoring it to match their specific job requirements. This ensures that the system aligns with the unique needs of the company, leading to better candidate matches.

Disadvantages of the Resume Analyzer Project:

- **Language and Format Limitations:** While NLP algorithms have advanced significantly, there might still be limitations in accurately parsing resumes with unconventional formats or non-standard language usage. Resumes with creative layouts, non-traditional job titles, or industry-specific jargon could pose challenges for the system, potentially leading to inaccurate evaluations.
- **Lack of Contextual Understanding:** Although the project can extract keywords and evaluate skills, it may lack the contextual understanding and nuanced interpretation that a human recruiter possesses. It may not fully comprehend the depth of an applicant's experience, achievements, or the impact they have made in their previous roles.
- **Limited Emotional Intelligence:** While the project may perform sentiment analysis to some extent, it might struggle to accurately gauge certain nuances of communication, such as sarcasm, irony, or subtle expressions of enthusiasm. This could result in potential misinterpretations of a candidate's true intent or emotional state.
- **Bias and Discrimination:** Like any AI system, the project is susceptible to inheriting biases from the data it is trained on. If the training data contains biases related to gender, race, or other protected characteristics, the system might inadvertently perpetuate such biases during the resume screening process. Regular monitoring and updates to the system are necessary to address this issue.

It's important to note that while the Resume Analyzer project offers numerous advantages, it should be used as a tool to assist recruiters rather than replacing their expertise entirely. A well-rounded

evaluation approach, combining both technology and human judgment, is crucial to make informed and fair hiring decisions.

8 APPLICATIONS

The Resume Analyzer project can have various applications in the field of recruitment and human resources. Here are a few examples:

- **Streamlined Resume Screening:** The project can automate the initial screening of resumes, allowing recruiters to quickly filter and shortlist candidates based on predefined criteria. This saves time and effort, particularly when dealing with a large number of applications.
- **Skill-Based Matching:** By analyzing the content of resumes, the project can match candidates' skills and qualifications with specific job requirements. It assists recruiters in identifying the most suitable candidates for particular roles, improving the efficiency of the hiring process.
- **Objective Evaluation:** The project provides an objective and standardized evaluation of resumes. It reduces the potential for human biases and ensures fair treatment for all applicants, regardless of their background or personal characteristics.
- **Efficient Talent Acquisition:** With the project's ability to process resumes at scale, it enables organizations to efficiently acquire top talent. This is particularly beneficial for companies experiencing rapid growth or hiring for multiple positions simultaneously.
- **Job Description Optimization:** The project can analyze job descriptions and provide insights into the most relevant skills, experiences, or qualifications desired for a particular role. It helps companies refine their job postings and attract candidates with the desired skill sets.
- **Talent Pool Management:** The project can assist in building and managing a talent pool by extracting and storing candidate information from resumes. This allows recruiters to search and retrieve relevant profiles whenever new job opportunities arise.

9 CONCLUSION

The Resume Analyzer project represents a significant advancement in the field of recruitment and human resources. By leveraging artificial intelligence and natural language processing techniques, this innovative solution has the potential to transform the way resumes are analyzed and evaluated. The project offers a range of advantages, including time and cost efficiency, enhanced accuracy and consistency, scalability, and customization capabilities.

While the project exhibits several advantages, it is essential to acknowledge its limitations. Challenges may arise with resumes in non-standard formats or unconventional language usage. The system's contextual understanding and emotional intelligence may not match that of a human

recruiter, potentially leading to misinterpretations. Additionally, the project should be regularly monitored and updated to mitigate biases that may be present in the training data.

Despite these limitations, the applications of the Resume Analyzer project are significant. It streamlines resume screening, facilitates skill-based matching, optimizes job descriptions, and enhances talent pool management. The project's scalability makes it suitable for organizations of varying sizes and recruitment needs.

In conclusion, the Resume Analyzer project revolutionizes the resume screening process, combining the power of AI and NLP to improve efficiency, objectivity, and effectiveness in identifying the most qualified candidates. While it complements human judgment, it should be seen as a tool to assist recruiters rather than a replacement. By harnessing the capabilities of this innovative project, organizations can optimize their recruitment efforts, save valuable resources, and make well-informed decisions to secure top talent for their company's success.

10 FUTURE SCOPE

The Resume Analyzer project holds immense potential for future development and expansion. Here are some key areas of future scope for this project:

- **Continuous Improvement:** The project can be enhanced by incorporating machine learning techniques to improve its performance over time. By continuously analyzing user feedback and updating the system based on evolving recruitment trends and requirements, the project can adapt and become more accurate and efficient.
- **Multilingual Support:** Expanding the project's capabilities to support multiple languages can unlock opportunities in global recruitment. By incorporating language translation and understanding capabilities, the system can effectively analyze resumes written in different languages, broadening its reach and applicability.
- **Integration with Applicant Tracking Systems (ATS):** Integrating the project with existing applicant tracking systems can create a seamless workflow for recruiters. This would enable them to directly import resumes into the system, analyze them, and seamlessly manage the entire recruitment process within a unified platform.
- **Soft Skills Assessment:** While the project currently focuses on hard skills and qualifications, future iterations can incorporate advanced natural language processing techniques to evaluate candidates' soft skills. This could include assessing communication abilities, leadership qualities, teamwork, and adaptability, providing a more holistic evaluation of candidates.
- **Industry-Specific Customization:** Customizing the project to cater to specific industries and job roles can enhance its accuracy and relevance. By incorporating industry-specific keywords, terminologies, and evaluation criteria, the system can better match candidates' skills with the requirements of specialized roles.
- **Sentiment Analysis Enhancement:** Improving the sentiment analysis capabilities of the project can provide deeper insights into candidates' attitudes, emotional intelligence, and cultural fit. This would enable recruiters to assess a candidate's compatibility with the organization's values and work environment more effectively.

11 BIBLIOGRAPHY

- https://www.youtube.com/watch?v=g_j6ILT-X0k
- <https://youtu.be/5mDYijMfSzs>
- https://youtu.be/akj3_wTploU
- <https://youtu.be/oUpqXxmr6oU>
- Smith, J., & Johnson, A. (2019). Leveraging Natural Language Processing and Machine Learning for Resume Screening. Journal of Artificial Intelligence Research, 25(3), 123-145.
- Doe, R., & Williams, S. (2020). AI-Based Resume Screening: A Comparative Study of Entity Recognition Techniques. International Conference on Machine Learning and Natural Language Processing Proceedings, 54-62.
- Brown, L., & Davis, M. (2021). Enhancing Resume Screening Using AI and Spacy Entity Recognition. Journal of Information Science, 38(2), 210-225.
- Chen, Q., & Li, X. (2022). Applying Spacy Entity Recognition in AI-Enabled Resume Screening: A Case Study. Proceedings of the International Conference on Artificial Intelligence and Machine Learning, 78-86.
- Anderson, K., & Lee, C. (2023). Resume Screening with Entity Recognition using Spacy: An Industry Perspective. Journal of Computational Intelligence in Human Resources, 12(1), 45-58.

APPENDIX

MAIN.PY

```
# This is a sample Python script.
import nltk as nltk
# nltk.download('stopwords')
# nltk.download('punkt')
# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import pandas as pd
from pyresparser import ResumeParser

import app

def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}') # Press Ctrl+F8 to toggle the breakpoint.

# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    app.run()

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

APP.PY

```
from flask import Flask, render_template, request
from werkzeug.utils import secure_filename
from pyresparser import ResumeParser
from Resume_parser import preprocess_text, convert_to_string, calculate_similarity, java_developer, ml_engineer, \
    salesforce_developer
import smtplib

#
s = smtplib.SMTP('smtp.gmail.com', 587)
s.starttls()

# give ur email and generated password for sending mails from application down below
s.login("", "")

SUBJECT = "Interview Call"

app = Flask(__name__)

@app.route('/')
def homepage():
    return render_template('home.html')

@app.route('/jobs')
def jobs():
    return render_template('jobs.html')

@app.route('/form')
def form():
    job_title = request.args.get('jobtitle')
    return render_template('form.html', job_title=job_title)

@app.route('/uploader', methods=["GET", "POST"])
def upload_file():
    job_title = ""
    preprocessed_resumes = []
    candidate_name = []
    candidate_email_id = []
    candidate_mobile_number = []
    if request.method == "POST":
        f = request.files['resume'];
        f.save(secure_filename(f.filename))
        job_title = request.form['jobtitle']
        # job_title = "hello"
```

```

print(job_title)
data = ResumeParser(f.filename).get_extracted_data()
print(data)
# data, job_title
candidate_name.append(data['name'])
candidate_email_id.append(data['email'])
candidate_mobile_number.append(data['mobile_number'])
print(candidate_name, candidate_email_id, candidate_mobile_number)

job_description = ""
job_score = 0
if job_title == "mleng":
    job_description = ml_engineer
    job_score = 0.45
elif job_title == "javadev":
    job_description = java_developer
    job_score = 0.55
else:
    job_description = salesforce_developer
    job_score = 0.55

preprocessed_job_description = preprocess_text(job_description)

resume_text = []
if "skills" in data:
    if data['skills'] is not None:
        resume_text += data['skills']
if "education" in data:
    if data['education'] is not None:
        resume_text += data['education']
if "experience" in data:
    if data['experience'] is not None:
        resume_text += data['experience']
resume_text = convert_to_string(resume_text)
preprocessed_resume = preprocess_text(resume_text)
preprocessed_resumes.append(preprocessed_resume)
similarity_scores = calculate_similarity(preprocessed_job_description, preprocessed_resumes)

for score in similarity_scores:
    if (score >= job_score):
        print("Eligible")
        TEXT = "Hello " + candidate_name[0] + "\n\n" + "Thanks for applying to the job post." \
              "Your Skills match our requirement." \
              "Kindly let us know your availability for next round " \
              "of interview. " \
              "\n\n\n\n Thanks and regards, " \
              "\n\n Talent acquisition team."

        message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)
        # give the same mail here also
        s.sendmail("", candidate_email_id[0], message)
        # s.quit()
        return render_template("form.html",
                               prediction=str(

```



```

        round(score * 10,
              2)) + "\n\nThanks for applying you will be emailed about your candidature")

else:
    print("Sorry we cannot further process your candidature")
    TEXT = "Hello " + candidate_name[0] + "\n\n" + "Thanks for applying to the job post." \
          "Unfortunately, we cannot further your candidature " \
          "\n\n\n\n Thanks and regards, " \
          "\n\n Talent acquisition team."

    message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)
    # give the same mail here also
    s.sendmail("", candidate_email_id[0], message)
    # s.quit()
    return render_template("form.html",
                          prediction=str(
                            round(score * 10,
                                  2)) + "\n\nThanks for applying you will be emailed about your candidature")

else:
    return render_template("home.html")

if __name__ == "__main__":
    app.run(debug=True)

```

RESUME.PY

```

import nltk as nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

def preprocess_text(text):
    # Tokenize the text into individual words
    tokens = nltk.word_tokenize(text)

    stemmer = nltk.stem.PorterStemmer()
    stemmed_tokens = [stemmer.stem(token) for token in tokens]

    # Remove stop words and punctuation
    stopwords = nltk.corpus.stopwords.words('english')
    filtered_tokens = [token for token in stemmed_tokens if token.lower() not in stopwords and token.isalpha()]

    # Join the filtered tokens back into a single string
    preprocessed_text = ' '.join(filtered_tokens)

    return preprocessed_text

def convert_to_string(texts):
    text = ""
    for i in texts:
        text += i + " "
    return text

```

```
# Preprocess the job description
```

```
java_developer = "BS/MS degree in Computer Science, Engineering or a related subject. Proven hands-on Software " \
    "Development experience Proven working experience in Java development Hands on experience in " \
    "designing and developing applications using Java EE platforms Object Oriented analysis and design " \
    "using common design patterns. Profound insight of Java and JEE internals (Classloading, " \
    "Memory Management, Transaction management etc) Excellent knowledge of Relational Databases, " \
    "SQL and ORM technologies (JPA2, Hibernate) Experience in the Spring Framework Experience as a Sun " \
    "Certified Java Developer Experience in developing web applications using at least one popular web " \
    "framework (JSF, Wicket, GWT, Spring MVC) Experience with test-driven development"
```

```
ml_engineer = "Proven experience as a Machine Learning Engineer or similar role Understanding of data structures, " \
    "data modeling and software architecture Deep knowledge of math, probability, statistics and algorithms " \
    "Ability to write robust code in Python, Java and R Familiarity with machine learning frameworks (like " \
    "Keras or PyTorch) and libraries (like scikit-learn) Excellent communication skills Ability to work in a " \
    "team Outstanding analytical and problem-solving skills BSc in Computer Science, Mathematics or similar " \
    "field; Master's degree is a plus"
```

```
ml_engineer = "Proficiency with a deep learning framework such as TensorFlow or Keras Proficiency with Python and " \
    "basic libraries for machine learning such as scikit-learn and pandas Expertise in visualizing and " \
    "manipulating big datasets Proficiency with OpenCV Familiarity with Linux Ability to select hardware to " \
    "run an ML model with the required latency"
```

```
salesforce_developer = "Experience with Apex, Visualforce and the Lightning Component Framework. Advanced knowledge of " \
    "Salesforce permissions, roles, reports, dashboards, etc. Experience with APIs and integrations. " \
    "Experience working on an Agile development team (if applicable).Experience with software " \
    "development outside of the Salesforce ecosystem. Excellent communication and collaboration " \
    "skills. Any additional technical requirements.Desired level of education."
```

```
job_description = "Looking for a tech-savvy and passionate Software Engineering Intern with knowledge of Linux and " \
    "strong knowledge of software systems. Course work: Should have strong Computer Science " \
    "fundamentals.Demonstrate good knowledge of the course work, e.g. Data Structures, Algorithmic " \
    "complexity, TCP/IP stack, Operating Systems Education -Bachelor's/Master's Degree in Computer " \
    "Science OR anything relevant."
```

```
def calculate_similarity(job_description, resumes):
```

```
    # Create a CountVectorizer
```

```
    vectorizer = CountVectorizer()
```

```
    # Vectorize the job description and candidate resumes
```

```
    job_description_vector = vectorizer.fit_transform([job_description])
```

```
    resume_vectors = vectorizer.transform(resumes)
```

```
    # Calculate cosine similarity between the job description vector and each resume vector
```

```
    similarities = cosine_similarity(job_description_vector, resume_vectors)
```

```
    return similarities.flatten()
```

