IT5037

Machine Learning

Resume Parser

By-

Aakash LS (2020506001)

Adithya Subramani (2020506006)

Akash Nagarajan (2020506008)

INTRODUCTION:

Scope

In a dynamic job market where candidates navigate a labyrinth of opportunities, the Resume Parser stands out as an ML-driven ally designed to empower them. This innovative project utilizes the latest ML concepts to provide candidates with insights, guidance, and invaluable feedback, ultimately assisting them in finding roles that align with their skills and aspirations.

Motivation

The ever-evolving job market poses challenges for candidates seeking the right opportunities. The motivation behind the Resume Parser lies in addressing these challenges by leveraging cutting-edge ML techniques. It aims to simplify the job search process, offering a comprehensive tool that goes beyond traditional approaches. By harnessing the power of ML, the project strives to enhance the overall experience for candidates, helping them make informed decisions in their career journey.

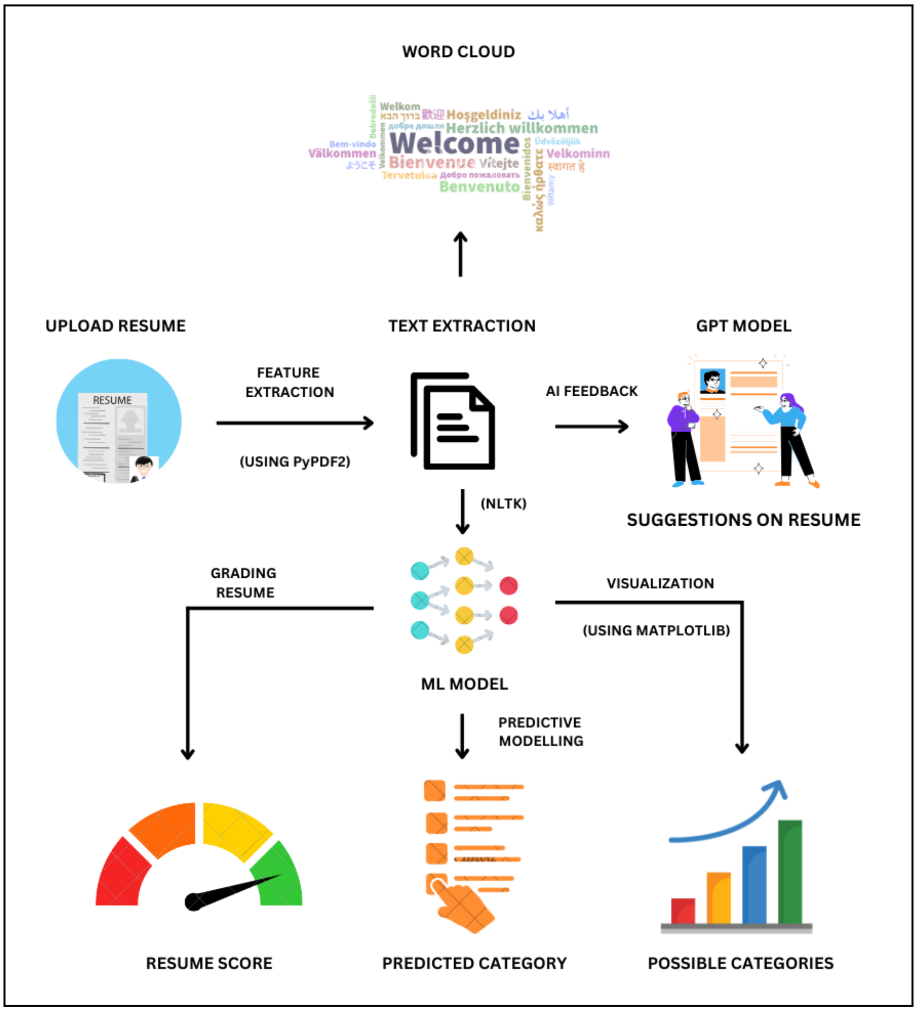
Objective

The Resume Parser's goal is to empower candidates with ML-enhanced insights, personalized guidance, and feedback, facilitating well-informed career choices in resume parsing and analysis.

PROBLEM STATEMENT

To develop a Resume Parser using Machine Learning (ML) concepts and various ML tools that provide candidates with valuable feedback and areas for improvement.

ARCHITECTURE



Tools Required:

Scikit-learn: The backbone for our machine learning endeavors, enabling text classification and predictive modeling.

Natural Language Toolkit (NLTK): Enhancing our text analysis through ML-driven language processing tools.

PyPDF2: A crucial PDF parsing tool for text extraction from documents.

WordCloud: Creating visual representations of word frequency to improve candidate insights.

Matplotlib: Our go-to tool for generating data visualizations and charts.

JSON: Facilitating seamless communication with ML models through standardized data interchange.

ML Concepts Utilized:

Text Classification: Employing advanced ML algorithms to categorize resumes into specific job roles with higher accuracy than manual methods.

Predictive Modeling: Applying ML for predictive tasks, such as suggesting suitable job roles based on the content of a candidate's resume.

Feature Engineering: Enhancing text analysis through the engineering of relevant features for improved classification and prediction.

Visualization: Creating visual representations to offer candidates a clear view of the top five suitable job roles for informed decision-making.

Summary:

This project represents a groundbreaking leap into the realm of ML-driven resume parsing. It goes beyond simple analysis, offering candidates transformative experiences by providing:

Role Suggestions: Leveraging advanced ML algorithms, the Resume Parser suggests the most suitable job roles based on resume content, guiding candidates towards roles that align with their qualifications and experiences.

Invaluable Feedback: The integration of ML-driven models enables personalized and constructive feedback on resumes. This feedback goes beyond grammar and formatting, offering insights and recommendations for content and presentation improvement.

Visualization of Opportunities: The project provides candidates with clear visualizations of the top five suitable job roles, empowering them to make informed career decisions.

CODE:

```
import streamlit as st
import spacy
from PyPDF2 import PdfReader
import plotly.express as px
import nltk
import re
import pickle
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import g4f
import json

g4f.debug.logging = True
g4f.check_version = False
print(g4f.version)
print(g4f.Provider.Ails.params)

try:
    nlp = spacy.load("en_core_web_sm")
except OSError:
    spacy.cli.download("en_core_web_sm")
    nlp = spacy.load("en_core_web_sm")
nltk.download('punkt')
nltk.download('stopwords')

#loading models
clf = pickle.load(open('clf.pkl','rb'))
tfidfd = pickle.load(open('tfidf.pkl','rb'))
```

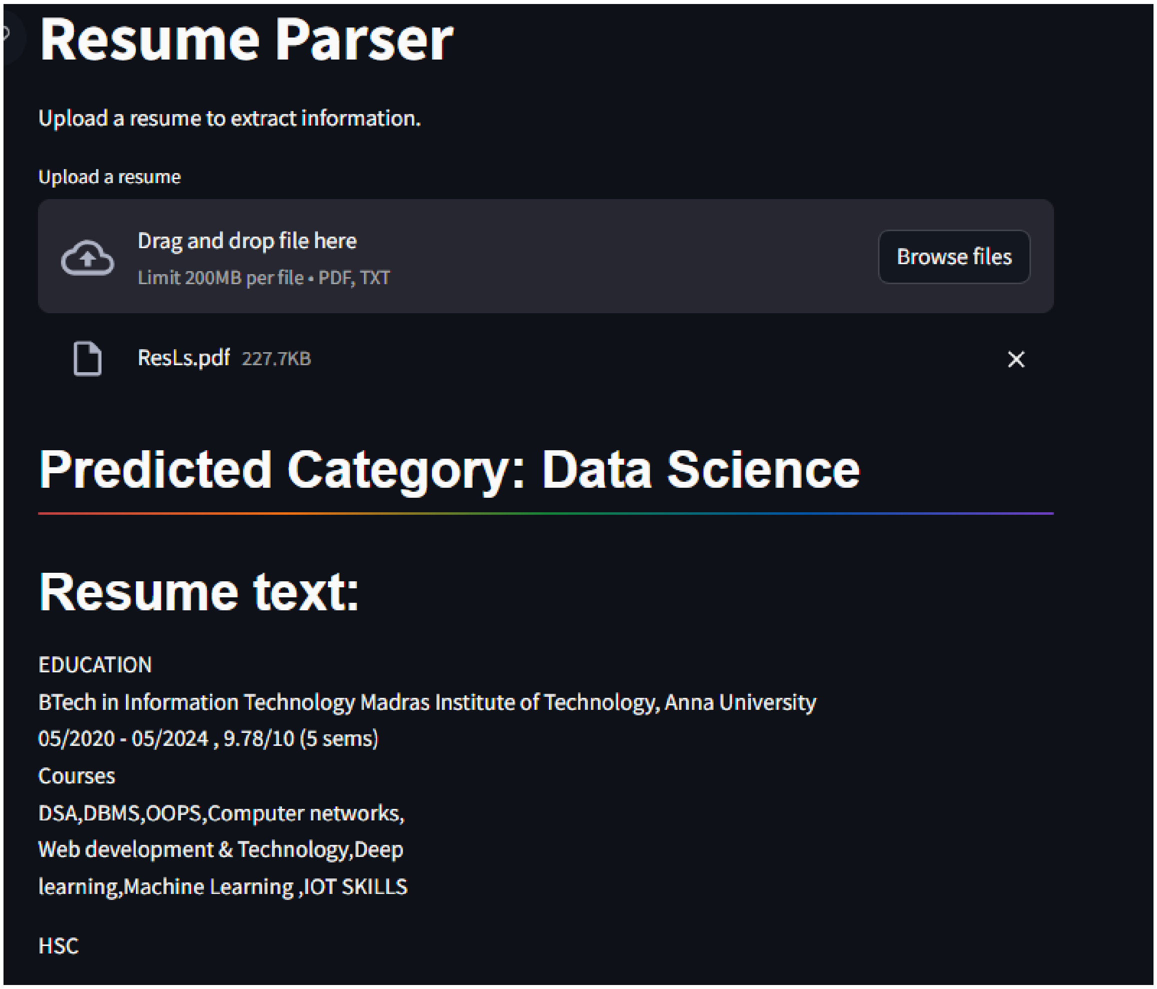
```
def get_gpt_feedback(resume_text):
     query = "Provide your open and honest feedback for the resume with the following content:"+resume_text
      query+=".Your response must be in the following format and should contain 2 sections/headings:\n1.Review\n2.Suggestions.\n"
     query+="Your response must be straight to the content without unnecessary info. Convey your message in bullet points"

   try:
      response = g4f.ChatCompletion.create(
         model="gpt-3.5-turbo",
         messages=[{"role": "user", "content": query}],
      )
   except RuntimeError:
      response = "Oops! The sserver is currently down."
   return response

def get_role_conf(resume_text):
    query="For the given resume, send a dictionary of values where the key would be the role suitable for the resume"
    query+="and the value would be the probability of the resume getting selected for the role."
    query+="reply only with the dictionary WITHOUT ANY EXTRA DETAILS. Suggest 5 such role-probability pairs in the dictionary"
   query+="Here is the resume "+resume_text

   try:
      response = g4f.ChatCompletion.create(
         model="gpt-3.5-turbo",
         messages=[{"role": "user", "content": query}],
      )
   except RuntimeError:
      response = "Oops! The sserver is currently down."
   return response

def clean_resume(resume_text):
   clean_text = re.sub('http\S+\s*', ' ', resume_text)
   clean_text = re.sub('RT|cc', ' ', clean_text)
   clean_text = re.sub('#\S+', '', clean_text)
```

```
    clean_text = re.sub('@\S+', '  ', clean_text)
    clean_text = re.sub('[%s]' % re.escape("""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""), ' ', clean_text)
    clean_text = re.sub(r'[^\x00-\x7f]', r' ', clean_text)
    clean_text = re.sub('\s+', ' ', clean_text)
    return clean_text

def extract_text_from_pdf(pdf_file):
    pdf_text = ""
    pdf_reader = PdfReader(pdf_file)
    for page in pdf_reader.pages:
        pdf_text += page.extract_text()
    return pdf_text

def extract_education(text):
    doc = nlp(text)
    education = []
    education_keywords = ["education", "qualification", "degree", "institute"]

    for token in doc:
        if token.text.lower() in education_keywords:
            education_info = []
            for i in range(1, 4):
                next_token = doc[token.i + i]
                if next_token.is_alpha or next_token.is_digit:
                    education_info.append(next_token.text)
            education.append(" ".join(education_info))

    return education

def extract_experience(text):
    doc = nlp(text)
    experience = []
    experience_keywords = ["experience", "work experience", "employment", "job"]
    for token in doc:
        if token.text.lower() in experience_keywords:
            experience_info = []
            for i in range(1, 6):
                next_token = doc[token.i + i]
                if next_token.is_alpha or next_token.is_digit:
                    experience_info.append(next_token.text)
            experience.append(" ".join(experience_info))

    return experience

def extract_and_score_resume(resume_text, necessary_fields, skills):
    # Custom scoring logic: Calculate score based on the number of necessary fields and
skills
    necessary_fields_count = sum(1 for field in necessary_fields if field.lower() in
resume_text.lower())
    skills_count = sum(1 for skill in skills if skill.lower() in resume_text.lower())

    # Define your own scoring formula here; this is just a simple example
    score = float(necessary_fields_count*10) + float(skills_count * 4) +
float(len(resume_text)/67.0)

    return float(score)

def extract_skills(text):
    doc = nlp(text)
    skills = []

    skills_keywords = ["skills", "technologies", "languages", "tools"]

    for token in doc:
        if token.text.lower() in skills_keywords:
            skill_info = []
            for i in range(1, 6):
                next_token = doc[token.i + i]
                if next_token.is_alpha:
                    skill_info.append(next_token.text)
            skills.extend(skill_info)

    return skills

st.title("Resume Parser")
st.write("Upload a resume to extract information.")
uploaded_file = st.file_uploader("Upload a resume", type=["pdf", "txt"])

if uploaded_file is not None:
    if uploaded_file.type == "application/pdf":
        resume_text = extract_text_from_pdf(uploaded_file)
    else:
        resume_text = uploaded_file.read().decode("utf-8")
    cleaned_resume = clean_resume(resume_text)
    input_features = tfidfd.transform([cleaned_resume])
    predictions = clf.predict(input_features)
    prediction_id = predictions[0]
    # st.write(predictions)
    wordcloud = WordCloud(width=800, height=400,
background_color="white").generate(resume_text)

    # Map category ID to category name
    category_mapping = {
        15: "Java Developer",
        23: "Testing",
        8: "DevOps Engineer",
        20: "Python Developer",
        24: "Web Designing",
        12: "HR",
        13: "Hadoop",
        3: "Blockchain",
        10: "ETL Developer",
        18: "Operations Manager",
        6: "Data Science",
        22: "Sales",
        16: "Mechanical Engineer",
        1: "Arts",
        7: "Database",
        11: "Electrical Engineering",
        14: "Health and fitness",
        19: "PMO",
        4: "Business Analyst",
        9: "DotNet Developer",
        2: "Automation Testing",
        17: "Network Security Engineer",
        21: "SAP Developer",
        5: "Civil Engineer",
        0: "Advocate",
    }
```

```
category_name = category_mapping.get(prediction_id, "Unknown")

# st.write("Predicted Category:"+str(category_name))
st.header("Predicted Category: "+category_name,divider="rainbow")

st.header("Resume text:")
st.write(resume_text)
st.header("",divider="rainbow")

education = extract_education(resume_text)
experience = extract_experience(resume_text)
skills = extract_skills(resume_text)
necessary_fields = ["degree", "experience", "qualification","projects"]

score = extract_and_score_resume(resume_text,necessary_fields,skills)
st.header(f"Resume Score: {score:.2f}")

# Create a donut chart to visualize the custom score
fig = px.pie(values=[score, 100 - score], names=["Score", "Remaining"])
fig.update_traces(hole=0.4, textinfo="percent+label")
st.plotly_chart(fig, use_container_width=True)
st.header("",divider="rainbow")
st.header("Word Cloud")
st.image(wordcloud.to_array())
st.header("",divider="rainbow")
st.write("Extracted Education:")
if education:
   st.write(education)
else:
   st.write("No education information found in the resume.")
st.write("Extracted Experience:")
if experience:
   st.write(experience)
else:
   st.write("No experience information found in the resume.")
st.write("Extracted Skills:")

if skills:
   st.write(skills)
```

```
else:
    st.write("No skills found in the resume.")
st.header("",divider="rainbow")

st.header("AI feedback")
feedback = get_gpt_feedback(cleaned_resume)
st.write(feedback)
st.header("",divider="rainbow")
st.header("Suitable roles")
role_conf = get_role_conf(cleaned_resume)
# st.write(role_conf)
data = json.loads(role_conf)
st.bar_chart(data)
st.header("",divider="rainbow")
```

OUTPUT SCREENSHOTS:



Resume Parser

Upload a resume to extract information.

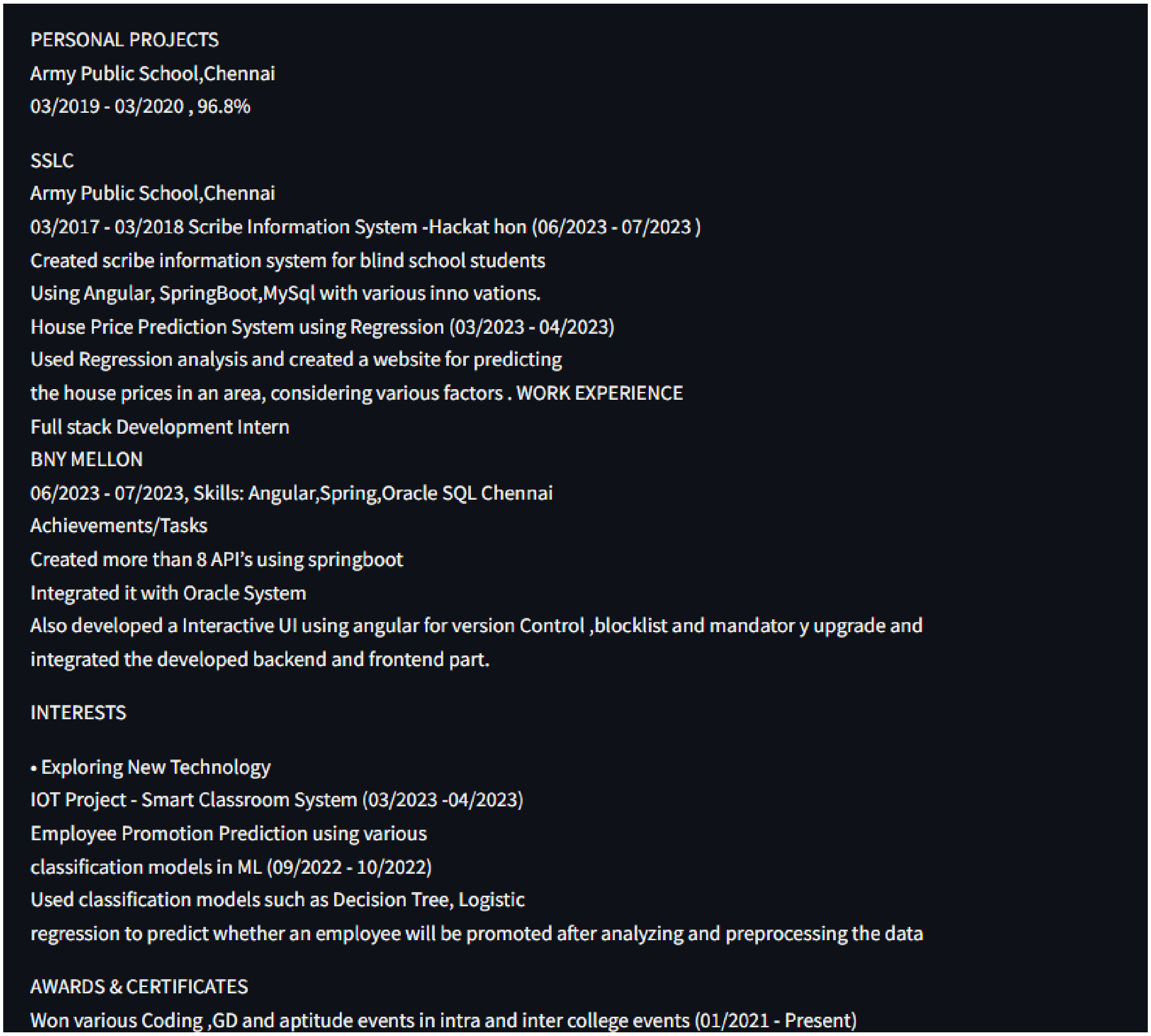
Upload a resume

Predicted Category: Data Science

Resume text:

EDUCATION
BTech in Information Technology Madras Institute of Technology, Anna University
05/2020 - 05/2024 , 9.78/10 (5 sems)
Courses
DSA,DBMS,OOPS,Computer networks,
Web development & Technology,Deep
learning,Machine Learning ,IOT SKILLS

HSC



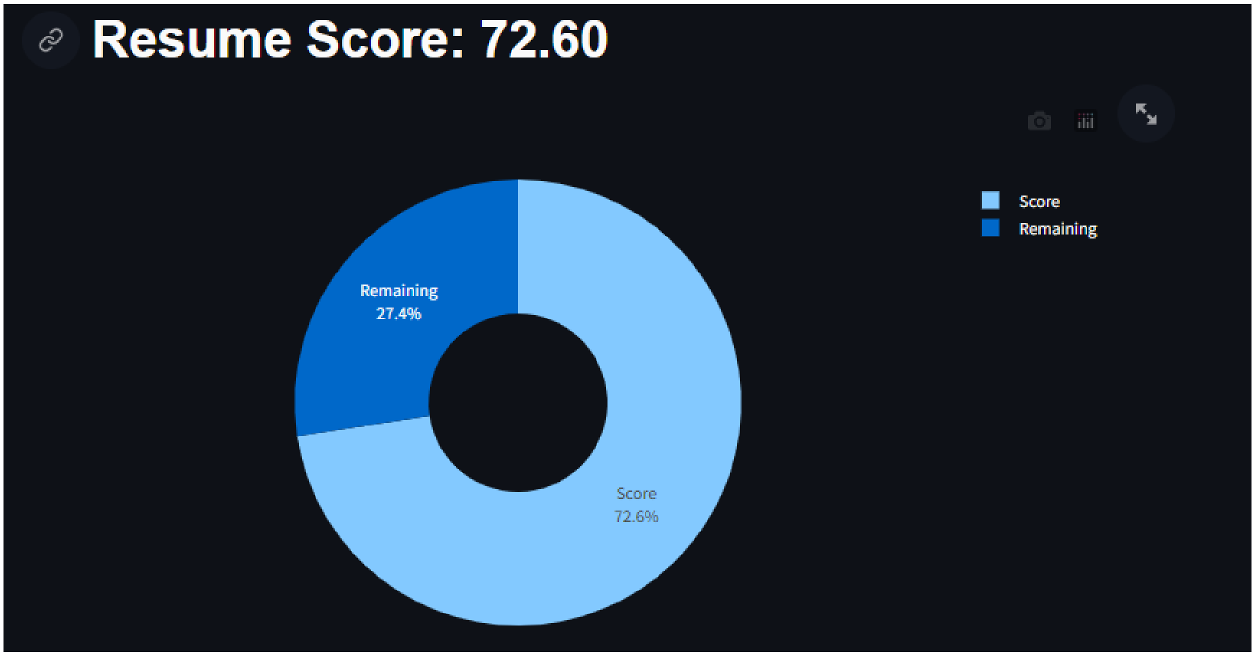
PERSONAL PROJECTS
Army Public School,Chennai
03/2019 - 03/2020 , 96.8%

SSLC
Army Public School,Chennai
03/2017 - 03/2018 Scribe Information System -Hackat hon (06/2023 - 07/2023)
Created scribe information system for blind school students
Using Angular, SpringBoot,MySql with various inno vations.
House Price Prediction System using Regression (03/2023 - 04/2023)
Used Regression analysis and created a website for predicting
the house prices in an area, considering various factors . WORK EXPERIENCE
Full stack Development Intern
BNY MELLON
06/2023 - 07/2023, Skills: Angular,Spring,Oracle SQL Chennai
Achievements/Tasks
Created more than 8 API's using springboot
Integrated it with Oracle System
Also developed a Interactive UI using angular for version Control ,blocklist and mandator y upgrade and
integrated the developed backend and frontend part.

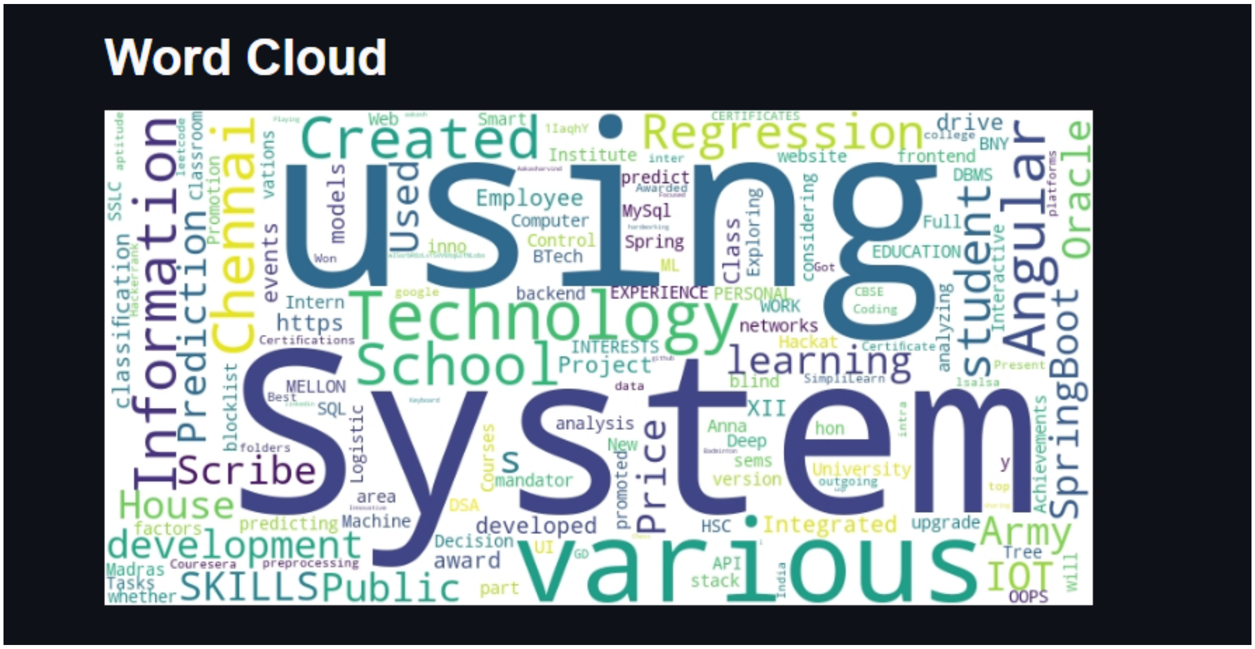
INTERESTS

• Exploring New Technology
IOT Project - Smart Classroom System (03/2023 -04/2023)
Employee Promotion Prediction using various
classification models in ML (09/2022 - 10/2022)
Used classification models such as Decision Tree, Logistic
regression to predict whether an employee will be promoted after analyzing and preprocessing the data

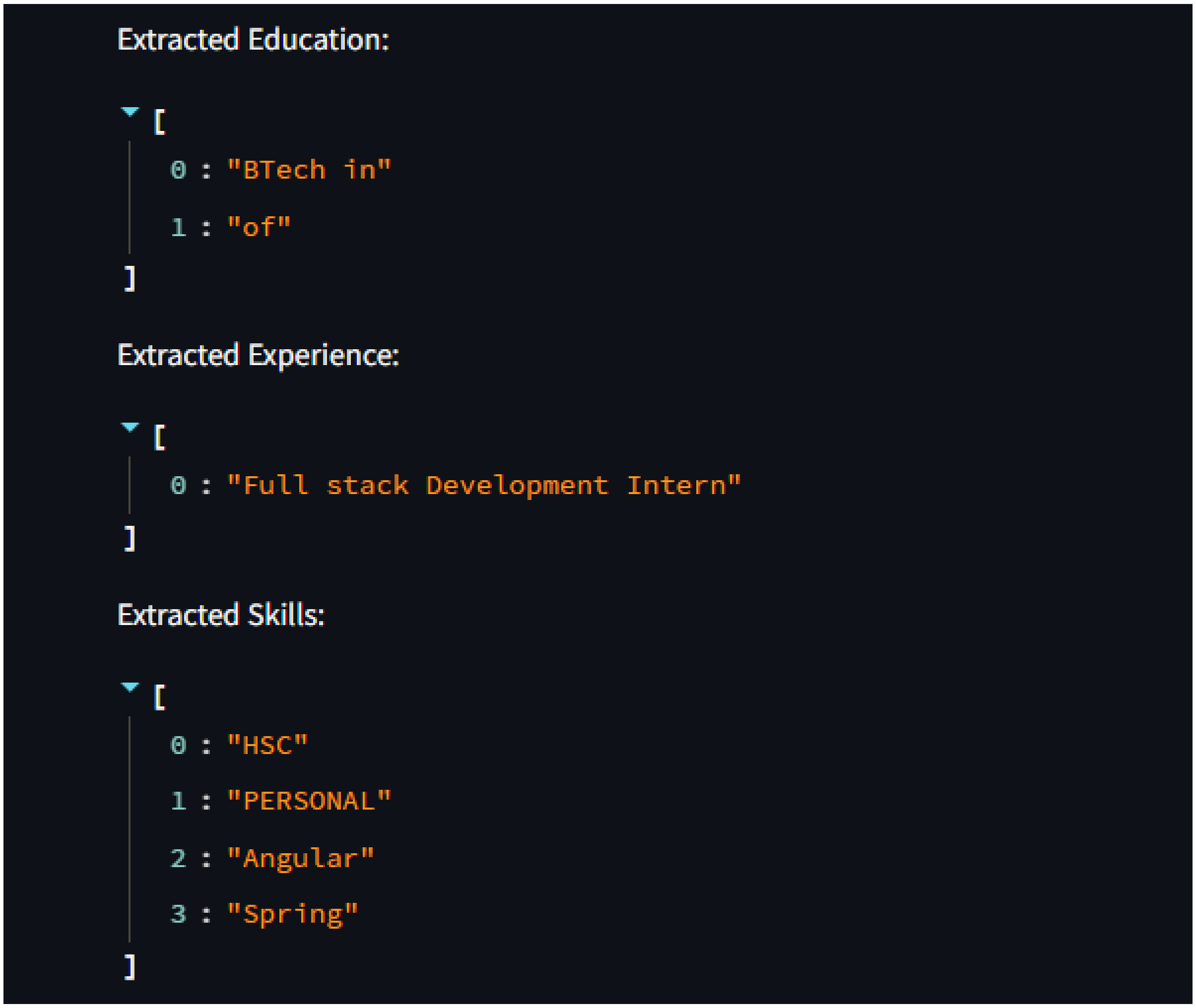
AWARDS & CERTIFICATES
Won various Coding ,GD and aptitude events in intra and inter college events (01/2021 - Present)



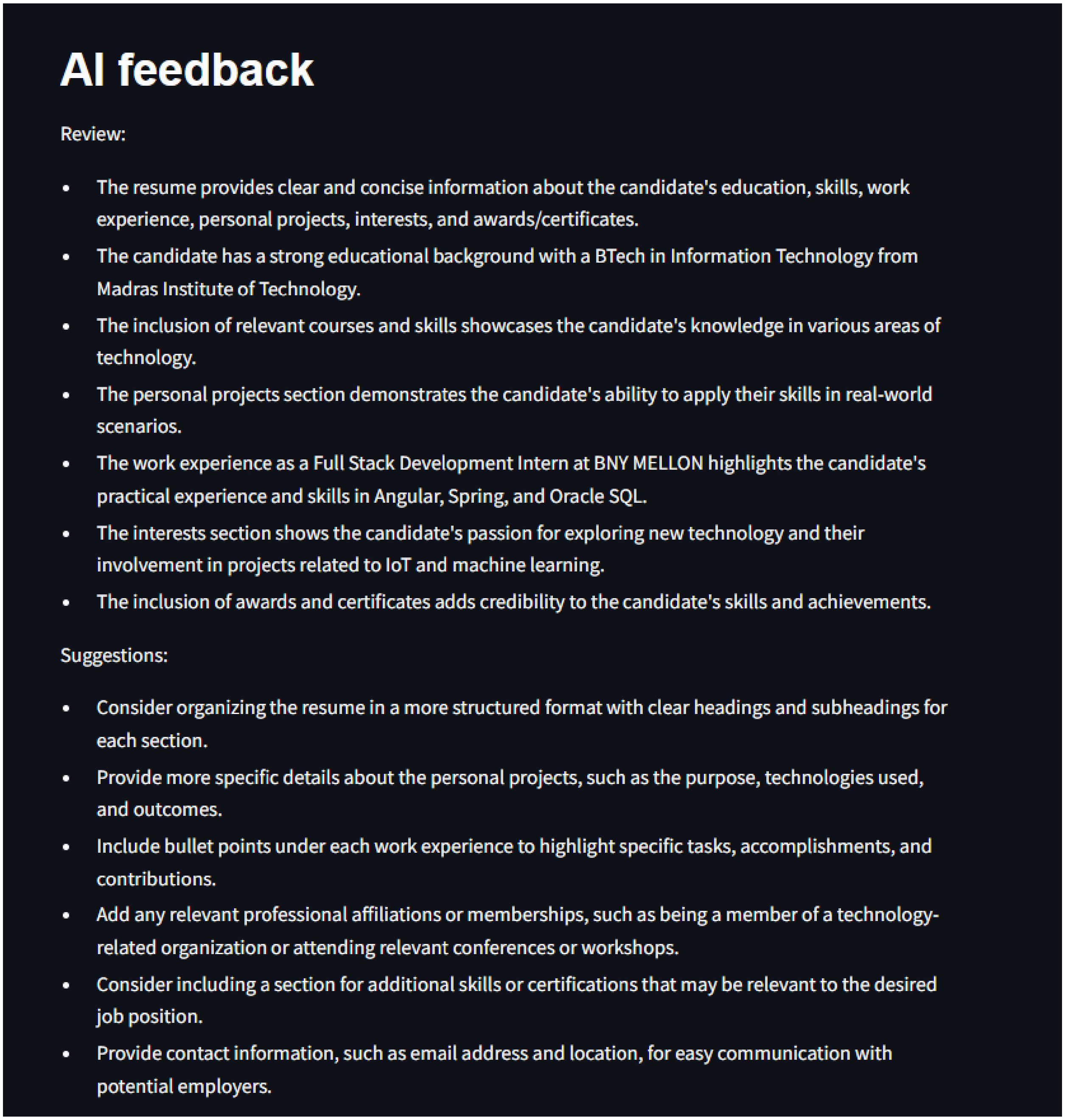
Resume Score: 72.60



Word Cloud



```
Extracted Education:

[
  0 : "BTech in"
  1 : "of"
]

Extracted Experience:

[
  0 : "Full stack Development Intern"
]

Extracted Skills:

[
  0 : "HSC"
  1 : "PERSONAL"
  2 : "Angular"
  3 : "Spring"
]
```



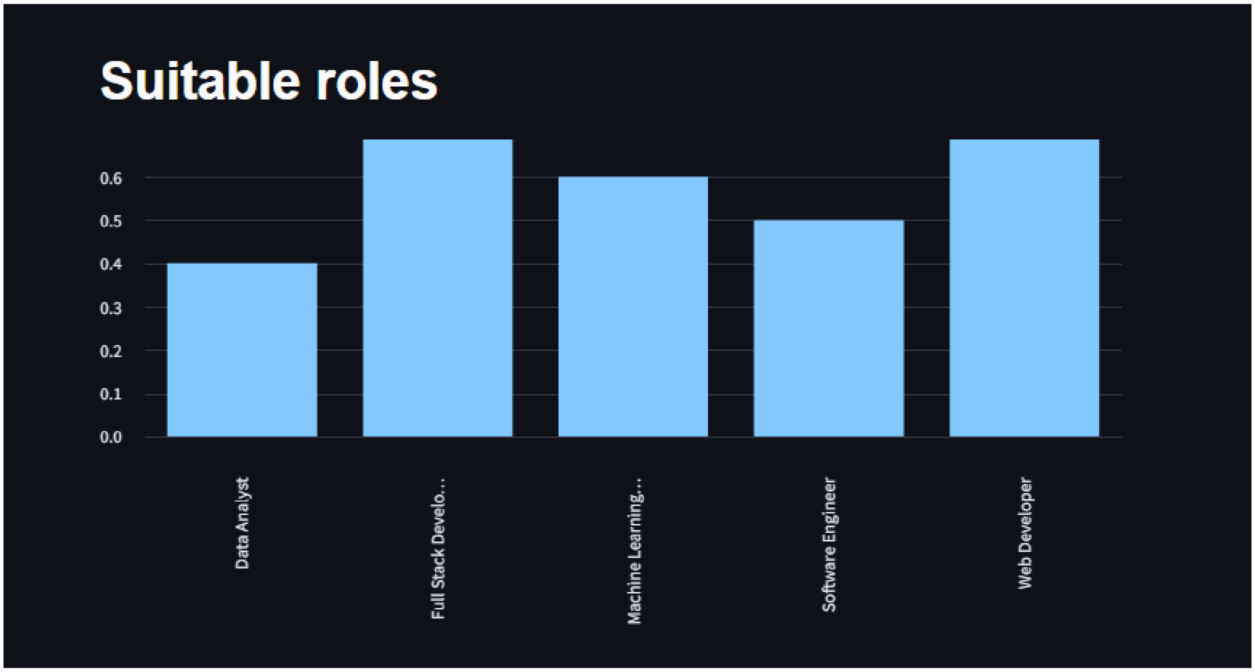
AI feedback

Review:

- The resume provides clear and concise information about the candidate's education, skills, work experience, personal projects, interests, and awards/certificates.
- The candidate has a strong educational background with a BTech in Information Technology from Madras Institute of Technology.
- The inclusion of relevant courses and skills showcases the candidate's knowledge in various areas of technology.
- The personal projects section demonstrates the candidate's ability to apply their skills in real-world scenarios.
- The work experience as a Full Stack Development Intern at BNY MELLON highlights the candidate's practical experience and skills in Angular, Spring, and Oracle SQL.
- The interests section shows the candidate's passion for exploring new technology and their involvement in projects related to IoT and machine learning.
- The inclusion of awards and certificates adds credibility to the candidate's skills and achievements.

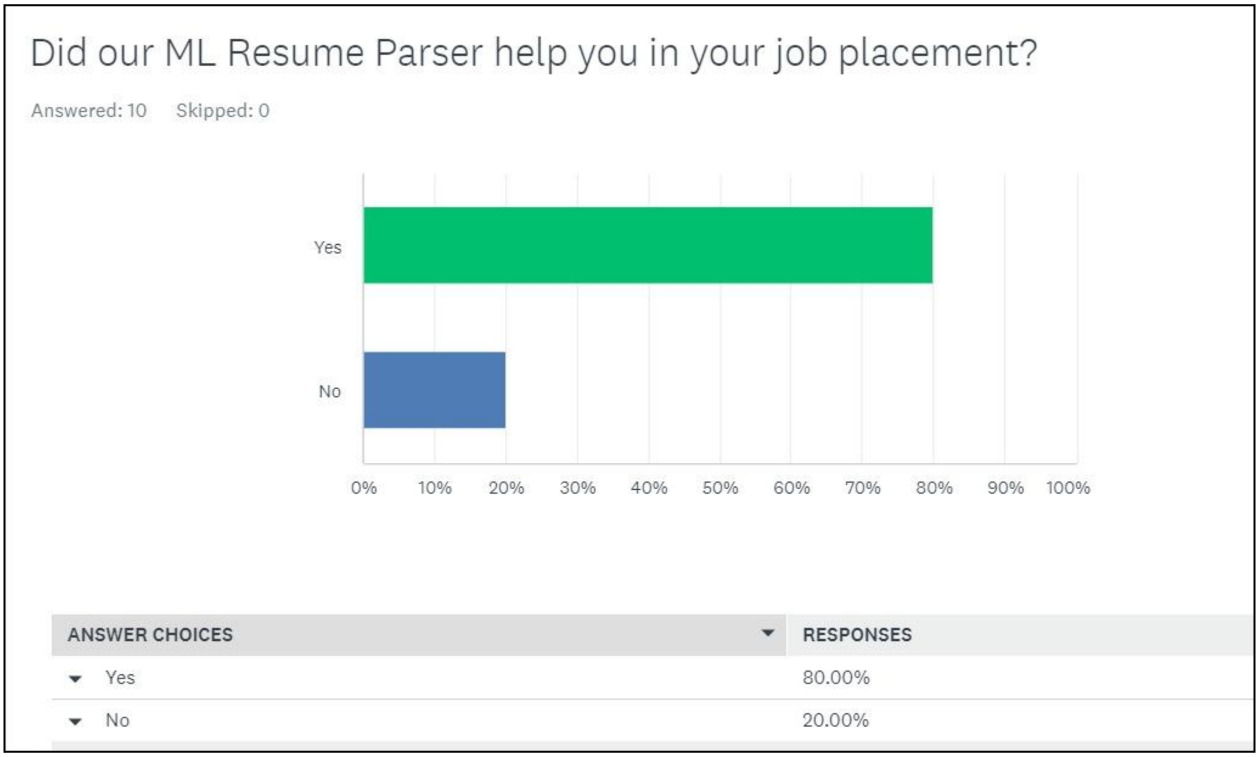
Suggestions:

- Consider organizing the resume in a more structured format with clear headings and subheadings for each section.
- Provide more specific details about the personal projects, such as the purpose, technologies used, and outcomes.
- Include bullet points under each work experience to highlight specific tasks, accomplishments, and contributions.
- Add any relevant professional affiliations or memberships, such as being a member of a technology-related organization or attending relevant conferences or workshops.
- Consider including a section for additional skills or certifications that may be relevant to the desired job position.
- Provide contact information, such as email address and location, for easy communication with potential employers.



Suitable roles

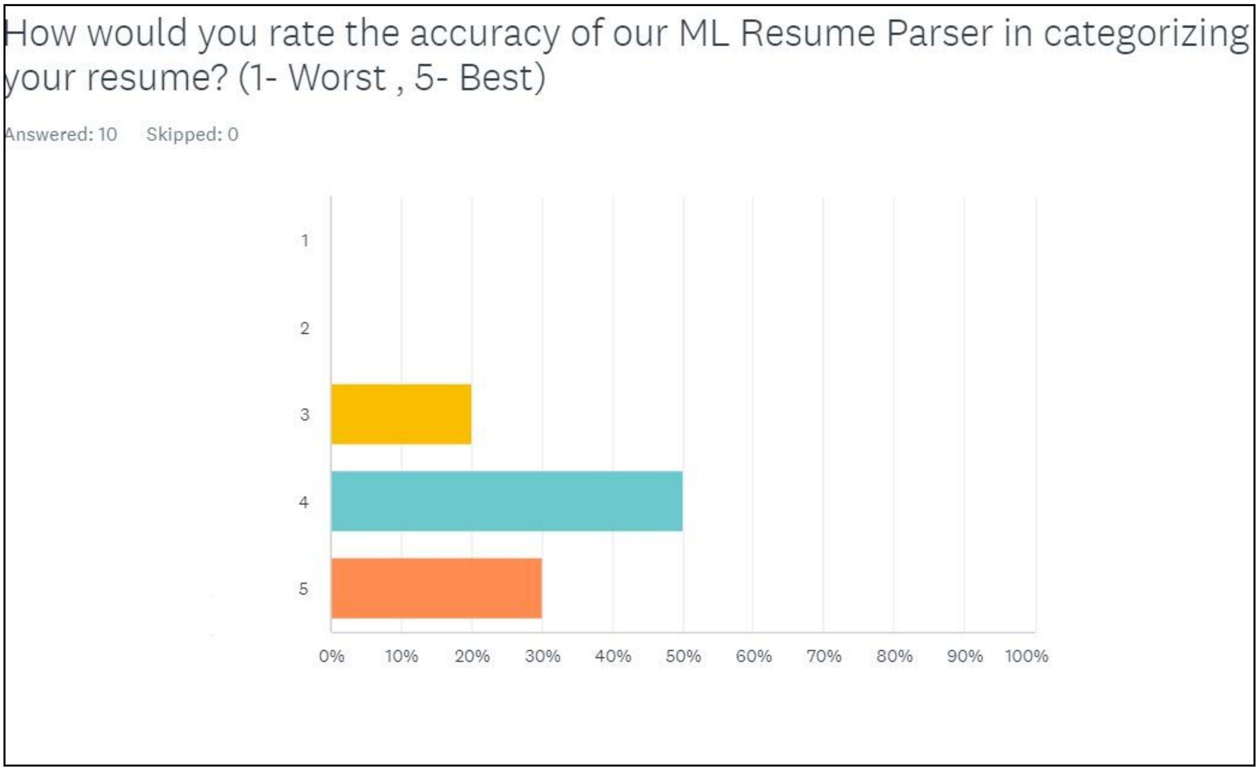
Survey Output:



Did our ML Resume Parser help you in your job placement?

Answered: 10   Skipped: 0

ANSWER CHOICES	RESPONSES
Yes	80.00%
No	20.00%



How would you rate the accuracy of our ML Resume Parser in categorizing your resume? (1- Worst , 5- Best)

Answered: 10   Skipped: 0

COMPARISON FROM EXISTING WORK:

In the paper [1]"An Automated Recommendation Approach to Selection in Personnel Recruitment" (Färber et al.), the authors propose an automated recommendation system for personnel recruitment. The system uses a combination of natural language processing (NLP) and machine learning (ML) techniques to identify and rank candidates based on their qualifications and experience.

In the paper [2]"Resume Parser Using NLP" (Anushka Sharma et al.),the resume parser extracts data from resumes and perform analysis to convert it into useful information for recruiters.The system uses NLP to save time and effort for recruiters, improve quality of selection.

Unlike existing solutions that provide limited information, our code stands out by extracting extensive details, including education, experience, skills, and predicting a job category. It introduces dynamic visualizations like word clouds and bar charts, offering recruiters a quick, intuitive understanding of a candidate's profile. Additionally, the integration of feedback from a language model enhances the evaluation process, providing valuable insights into potential issues within resumes. This multifaceted approach sets our code apart, significantly improving the efficiency and effectiveness of candidate assessments.

CONCLUSION:

Thus the Resume Parser has been developed which not just analyzes resumes but also helps individuals understand where they truly belong in the job market. By harnessing Machine Learning, it illuminates the path forward, offering invaluable feedback and suggesting the roles they genuinely deserve.

REFERENCES:

1. An Automated Recommendation Approach to Selection in Personnel Recruitment: https://link.springer.com/chapter/10.1007/978-3-319-11854-3_44
2. Resume Parser Using NLP: https://ieeexplore.ieee.org/document/9670958