REPORT

AI-Powered Career Guidance and Resume Screening

Introduction

- Career guidance is a crucial step for students and job seekers to understand their strengths and identify suitable career paths. However, many students struggle to determine which job roles match their skills and knowledge. Manually analyzing resumes is time-consuming, error-prone, and often lacks personalized feedback.
- With advancements in **Natural Language Processing (NLP)** and machine learning, it is now possible to **automate resume evaluation** and provide actionable insights. This project leverages Python, NLP techniques, and a web-based interface to assist students in identifying the most suitable career paths and skill improvement areas.

Problem Statement

- Students may not know which job roles best fit their skills.
- Manual resume evaluation is tedious and inconsistent.
- Students lack structured feedback to improve skills for specific roles.

Goal: Develop a system that analyzes resumes, identifies relevant skills, suggests top job roles, and provides tips for improvement.

Objective

- Automate resume evaluation for students and job seekers.
- Extract candidate name and key skills from resumes.
- Suggest top-matched job roles using TF-IDF and cosine similarity.
- Provide customized tips for skill improvement based on the best-matched role.
- Build a user-friendly web interface using Streamlit.

REPORT

Technologies and Libraries

Technology / Library	Purpose
Streamlit	Interactive web interface for easy resume upload and result display.
PyPDF2	Extracts text from PDF resumes.
Doc2txt	Extracts text from DOCX resumes.
NLTK	Tokenization, stopword removal, and preprocessing.
string	Removes punctuation during preprocessing.
Scikit-learn	Converts text to numerical vectors and calculates similarity.
pandas	Stores and displays results in tabular format.
Python	Programming language for backend logic.

System Architecture / Workflow

Workflow Steps:

1. Upload Resume (PDF/DOCX)
2. Read Resume Text using PyPDF2 or docx2txt
3. Preprocess Text: Lowercase, remove punctuation, remove stopwords, tokenize
4. Extract Candidate Name using heuristic rules
5. Skill Matching: Use TF-IDF vectorization and cosine similarity to compare resume skills with predefined job roles
6. Top Role Suggestions: Select top 3 roles based on similarity scores
7. Perfect Match Check: If all skills match, show confirmation; else, show improvement tips
8. Display Results Table with candidate name, top role matches, best suggested role, and tips

REPORT

Job Roles and Skills Mapping

Job Role	Key Skills
Data Analyst	Excel, SQL, Python, Data Visualization, Pandas, Tableau, Power BI
Web Developer	HTML, CSS, JavaScript, React, WordPress, PHP, Node.js
AI/ML Engineer	Python, TensorFlow, PyTorch, Machine Learning, Deep Learning, NumPy, Pandas
Cybersecurity Analyst	Network Security, Ethical Hacking, Penetration Testing, Cryptography
Software Engineer	Java, C++, Python, Data Structures, Algorithms, Git, Problem Solving
Cloud Engineer	AWS, Azure, GCP, Docker, Kubernetes, Linux, Terraform
DevOps Engineer	CI/CD, Jenkins, Docker, Kubernetes, Linux, Cloud Automation
UI/UX Designer	Figma, Adobe XD, Wireframing, Prototyping, User Research, Design Thinking
Content writer	SEO, Blog Writing, Copywriting, Editing, Social Media Content, WordPress
Data Scientist	Python, Statistics, Machine Learning, Data Visualization, Pandas, NumPy, Matplotlib
Mobile App Developer	Flutter, Dart, Java, Kotlin, Android Studio, Firebase
Blockchain Developer	Solidity, Ethereum, Smart Contracts, Web3, Cryptography, DApps

REPORT

CODE

```
import streamlit as st
import PyPDF2
import docx2txt
import nltk
import string
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd

# ================================
# NLTK Setup
# ================================
nltk.download("punkt")
nltk.download("stopwords")
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# ================================
# Job Roles & Skills
# ================================
job_roles = {
```

REPORT

"Data Analyst": ["excel", "sql", "python", "data visualization", "pandas", "tableau", "power bi"],

"Web Developer": ["html", "css", "javascript", "react", "wordpress", "php", "nodejs"],

"AI/ML Engineer": ["python", "tensorflow", "pytorch", "machine learning", "deep learning", "numpy", "pandas"],

"Cybersecurity Analyst": ["network security", "firewall", "ethical hacking", "penetration testing", "cryptography", "vulnerability analysis"],

"Software Engineer": ["java", "c++", "python", "data structures", "algorithms", "git", "problem solving"],

"Cloud Engineer": ["aws", "azure", "gcp", "docker", "kubernetes", "linux", "terraform"],

"DevOps Engineer": ["ci/cd", "jenkins", "docker", "kubernetes", "linux", "cloud infrastructure"],

"UI/UX Designer": ["figma", "adobe xd", "wireframing", "prototyping", "user research", "design thinking"],

"Content Writer": ["seo", "blog writing", "copywriting", "editing", "social media content", "wordpress"],

"Data Scientist": ["python", "statistics", "machine learning", "data visualization", "pandas", "numpy", "matplotlib"],

"Mobile App Developer": ["flutter", "dart", "java", "kotlin", "android studio", "firebase"],

"Blockchain Developer": ["solidity", "ethereum", "smart contracts", "web3", "cryptography", "decentralized apps"]

}

REPORT

```
# ================================
# Improvement Tips
# ================================
role_tips = {
    "Data Analyst": "Learn Power BI/Tableau, practice SQL queries, and strengthen Python & Excel data analysis.",
    "Web Developer": "Master HTML/CSS, learn React or Node.js, and build real-world websites or WordPress themes.",
    "AI/ML Engineer": "Practice Python, study ML algorithms, learn TensorFlow/PyTorch, and build AI-based projects.",
    "Cybersecurity Analyst": "Improve penetration testing, ethical hacking, and network monitoring skills with practical labs.",
    "Software Engineer": "Focus on DSA, problem-solving on LeetCode, and learn OOPs, Git, and version control best practices.",
    "Cloud Engineer": "Get familiar with AWS/Azure, learn Docker & Kubernetes, and practice deploying cloud applications.",
    "DevOps Engineer": "Understand CI/CD pipelines, Jenkins, Docker, and Kubernetes. Learn cloud automation tools like Terraform.",
    "UI/UX Designer": "Enhance your Figma or Adobe XD skills, practice wireframing & prototyping, and study user behavior.",
    "Content Writer": "Improve SEO writing, content research, and storytelling; build a writing portfolio on LinkedIn or Medium.",
    "Data Scientist": "Study statistics, build ML models, and improve data visualization and storytelling with data.",
    "Mobile App Developer": "Learn Flutter or Kotlin, focus on UI/UX design for mobile, and build your own mini apps for GitHub.",
```

REPORT

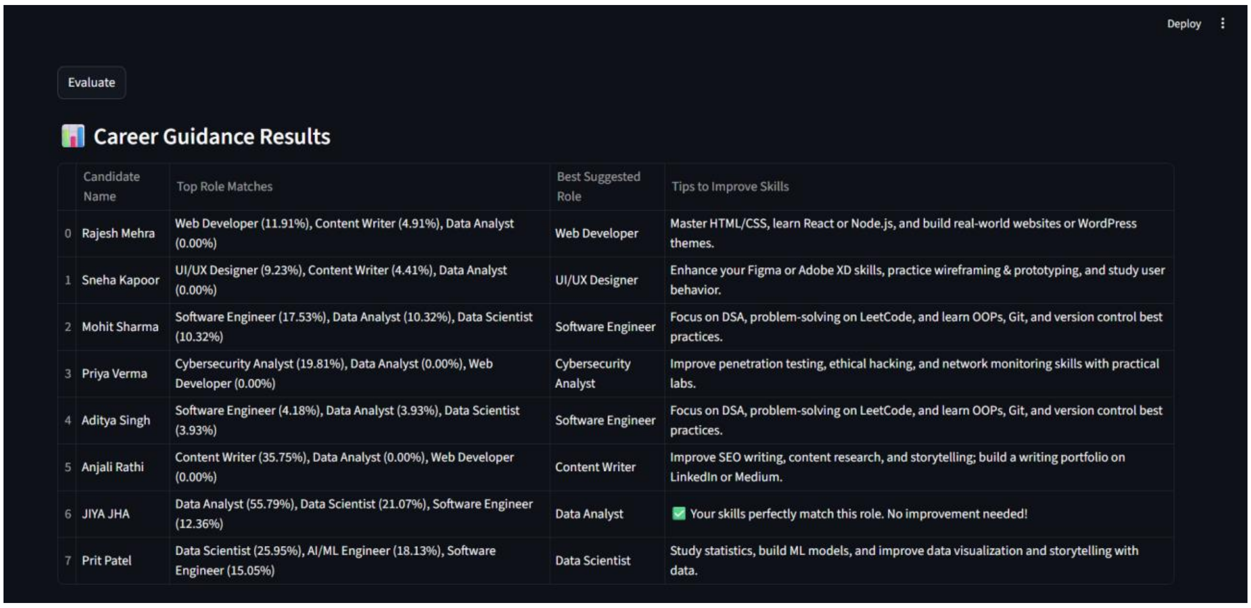
```
    "Blockchain Developer": "Master Solidity, learn Ethereum smart
contracts, and understand cryptographic principles and Web3."
}

# ================================
# Text Preprocessing
# ================================
def preprocess_text(text):
    text = text.lower()
    text = text.translate(str.maketrans("", "", string.punctuation))
    tokens = word_tokenize(text)
    stop_words = set(stopwords.words("english"))
    tokens = [w for w in tokens if w not in stop_words]
    return " ".join(tokens)

# ================================
# Resume Reader
# ================================
def read_file(uploaded_file):
    if uploaded_file.name.endswith(".pdf"):
        pdf_reader = PyPDF2.PdfReader(uploaded_file)
        text = ""
        for page in pdf_reader.pages:
            text += page.extract_text() or ""
        return text
    elif uploaded_file.name.endswith(".docx"):
        return docx2txt.process(uploaded_file)
    else:
        return ""

# ================================
# Extract Candidate Name
# ================================
def extract_candidate_name(text, filename):
    lines = [line.strip() for line in text.split("\n") if line.strip()]
    for line in lines:
        if 2 <= len(line.split()) <= 4 and not line.lower() == "resume":
            return line
    return filename.split(".")[0]

# ================================
# Suggest Job Roles
# ================================
def suggest_job_role(resume_text, top_n=3):
    resume_text = preprocess_text(resume_text)
    role_scores = {}
    for role, skills in job_roles.items():
        skills_text = " ".join(skills)
        vectorizer = TfidfVectorizer()
        tfidf_matrix = vectorizer.fit_transform([skills_text,
resume_text])
        score = cosine_similarity(tfidf_matrix[0:1],
tfidf_matrix[1:2])[0][0] * 100
        role_scores[role] = score

    sorted_roles = sorted(role_scores.items(), key=lambda x: x[1],
reverse=True)
    return sorted_roles[:top_n]

# ================================
# Check Perfect Skill Match
# ================================
def is_perfect_match(resume_text, role):
    resume_text = preprocess_text(resume_text)
    role_skills = [skill.lower() for skill in job_roles[role]]
    return all(skill in resume_text for skill in role_skills)

# ================================
# Streamlit App
# ================================
st.set_page_config(page_title="Career Guidance Tool",
page_icon="🎯", layout="wide")
st.title("🎯 Career Guidance Tool for Students")
```

REPORT

```
st.write("Upload your resume to find the most suitable job roles
and get tips to improve your skills.")


uploaded_files = st.file_uploader("Upload Resume(s) (PDF/DOCX)",
type=["pdf", "docx"], accept_multiple_files=True)


if st.button("Evaluate"):
    if uploaded_files:
        results = []
        for uploaded_file in uploaded_files:
            resume_text = read_file(uploaded_file)
            if resume_text.strip() == "":
                st.warning(f"No text found in {uploaded_file.name}.
Skipping...")
                continue


            candidate_name = extract_candidate_name(resume_text,
uploaded_file.name)
            top_roles = suggest_job_role(resume_text)


            best_role, best_score = top_roles[0][0], top_roles[0][1]


            # ✅ Use direct skill check for perfect match
            if is_perfect_match(resume_text, best_role):
                best_tips = "✅ Your skills perfectly match this role. No improvement needed!"
            else:
                best_tips = role_tips.get(best_role, "No tips available.")


            top_roles_str = ", ".join([f"{role} ({score:.2f}%)" for role, score in top_roles])
            results.append([candidate_name, top_roles_str, best_role, best_tips])


        results_df = pd.DataFrame(results, columns=["Candidate Name", "Top Role Matches", "Best Suggested Role", "Tips to Improve Skills"])
        st.subheader("📊 Career Guidance Results")
        st.table(results_df)
    else:
        st.warning("Please upload at least one resume.")
```

REPORT

Output



📊 Career Guidance Results

	Candidate Name	Top Role Matches	Best Suggested Role	Tips to Improve Skills
0	Rajesh Mehra	Web Developer (11.91%), Content Writer (4.91%), Data Analyst (0.00%)	Web Developer	Master HTML/CSS, learn React or Node.js, and build real-world websites or WordPress themes.
1	Sneha Kapoor	UI/UX Designer (9.23%), Content Writer (4.41%), Data Analyst (0.00%)	UI/UX Designer	Enhance your Figma or Adobe XD skills, practice wireframing & prototyping, and study user behavior.
2	Mohit Sharma	Software Engineer (17.53%), Data Analyst (10.32%), Data Scientist (10.32%)	Software Engineer	Focus on DSA, problem-solving on LeetCode, and learn OOPs, Git, and version control best practices.
3	Priya Verma	Cybersecurity Analyst (19.81%), Data Analyst (0.00%), Web Developer (0.00%)	Cybersecurity Analyst	Improve penetration testing, ethical hacking, and network monitoring skills with practical labs.
4	Aditya Singh	Software Engineer (4.18%), Data Analyst (3.93%), Data Scientist (3.93%)	Software Engineer	Focus on DSA, problem-solving on LeetCode, and learn OOPs, Git, and version control best practices.
5	Anjali Rathi	Content Writer (35.75%), Data Analyst (0.00%), Web Developer (0.00%)	Content Writer	Improve SEO writing, content research, and storytelling; build a writing portfolio on LinkedIn or Medium.
6	JIYA JHA	Data Analyst (55.79%), Data Scientist (21.07%), Software Engineer (12.36%)	Data Analyst	✅ Your skills perfectly match this role. No improvement needed!
7	Prit Patel	Data Scientist (25.95%), AI/ML Engineer (18.13%), Software Engineer (15.05%)	Data Scientist	Study statistics, build ML models, and improve data visualization and storytelling with data.

Detailed Implementation

1 Text Preprocessing

- Convert all text to lowercase.
- Remove punctuation using Python string.punctuation.
- Tokenize text using nltk.word_tokenize().
- Remove stopwords using NLTK's English stopword list.

This ensures relevant skills are extracted accurately.

2 Resume Reading

- PDF Files: PyPDF2.PdfReader reads each page and extracts text.
- DOCX Files: docx2txt.process reads and returns text.

REPORT

3 Candidate Name Extraction

- Heuristic: Line with 2–4 words, excluding the word "Resume", is assumed to be the candidate name.
- If no name is found, the system uses the filename as the candidate name.

4 Job Role Suggestion

- Preprocess resume text.
- For each role, combine role skills into a single string.
- Use TF-IDF vectorization to convert text to numeric vectors.
- Compute cosine similarity between resume and each role.
- Return top 3 roles based on highest similarity scores.

5 Perfect Skill Match

- Checks if all skills for the top role are present in the resume.
- Displays either:
    - Perfect match message or
    - Tips for improvement from predefined role_tips.

6 Streamlit Interface

- Upload multiple resumes.
- Click “Evaluate” to process resumes.
- Display results in interactive table with columns: Candidate Name, Top Role Matches, Best Suggested Role, Tips to Improve Skills.

Testing and Validation

- Sample resumes tested with different formats (PDF and DOCX).
- Top role matches aligned with expected results.
- Similarity scores calculated accurately using TF-IDF.
- Skill tips verified manually to ensure relevance.

REPORT

Limitations

- Complex PDF layouts may cause **inaccurate text extraction**.
- Keyword-based matching may miss **synonyms or context variations**.
- Candidate experience, projects, or education are **not evaluated**.
- Name extraction is heuristic and **may fail for unconventional formats**.

Future Scope

- Use **BERT / spaCy embeddings** for semantic skill matching.
- Analyze **experience, education, and projects** for improved recommendations.
- Add **resume scoring** and **visual analytics** like charts or word clouds.
- Integrate with **LinkedIn/job portals** for real-time role suggestions.
- Deploy as a **cloud application** for wider accessibility.

Use Cases

1. **Students preparing for internships** – Evaluate skills and get tips.
2. **Job seekers exploring roles** – Identify roles suited to their skills.
3. **Career counselors** – Use as a tool to guide students with personalized advice.

References

- NLTK Documentation: https://www.nltk.org/
- PyPDF2 Documentation: https://pypi.org/project/PyPDF2/
- docx2txt Documentation: https://pypi.org/project/docx2txt/
- Scikit-learn Documentation: https://scikit-learn.org/stable/
- Streamlit Documentation: https://docs.streamlit.io/

REPORT

Summary of Learning Outcomes

- Implemented **text preprocessing** using NLP techniques.
- Understood **TF-IDF vectorization and cosine similarity** for skill-role matching.
- Developed an **interactive web application** with Streamlit.
- Learned to integrate **multiple Python libraries** for a real-world project.
- Gained insights into **career guidance automation** and practical AI applications.