Resume Analyzer Documentation

Overview

The Resume Analyzer is a tool that helps users analyze their resumes and improve them for specific job roles. It uses artificial intelligence (AI) to:

1. Extract and summarize the content of a resume.
2. Analyze the resume's strengths, weaknesses, opportunities, and threats (SWOT analysis).
3. Provide a match score between the resume and a job description.
4. Identify missing skills and recommend relevant Udemy courses.
5. Offer actionable suggestions for improving the resume.

The tool is built using Python and leverages libraries like Streamlit for the user interface, **Google Generative AI** for text analysis, and Pandas for data handling.

How It Works

1. Setting Up the Environment

The code starts by importing necessary libraries and setting up the Streamlit interface.

- **Libraries Used:**
  - pandas: For handling CSV files (e.g., skills and Udemy courses).
  - streamlit: For creating the web interface.
  - google.generativeai: For using Google's AI model (Gemini) to analyze text.
  - PyPDF2: For extracting text from uploaded PDF resumes.
  - os: For handling environment variables (e.g., API keys).
  - re: For text pattern matching (e.g., extracting skills).
  - langchain.prompts: For creating structured prompts for AI analysis.
- **Streamlit Setup:**
  - The page title is set to "Resume Analyzer" with an icon.
  - A sidebar allows users to navigate between two pages:
    1. **Page 1: Resume Analysis** (for analyzing resumes).
    2. **Page 2: Improvement Suggestions** (for getting tips to improve resumes).

2. Google API Key

The tool uses Google's AI model, which requires an API key.

- If the API key is not set in the environment, the user is prompted to enter it.
- If no key is provided, the tool stops and shows a warning.

3. Loading Skills

The tool loads a list of skills from a CSV file (skill.csv).

- The skills are grouped by category (e.g., "Programming", "Design").
- If the file is missing or has errors, the tool stops and shows an error message.

4. Extracting Text from PDF

When a user uploads a resume (PDF), the tool extracts the text.

- The PyPDF2 library reads the PDF and extracts text from each page.
- If the extraction fails, an error message is shown.

5. Extracting Skills from Text

The tool compares the extracted resume text with the list of skills.

- It identifies which skills are mentioned in the resume.
- Missing skills (those in the job description but not in the resume) are highlighted.

6. Recommending Udemy Courses

For missing skills, the tool recommends relevant Udemy courses.

- Course data is loaded from a CSV file (udemy_courses.csv).
- The tool searches for courses that match the missing skills and displays their titles and URLs.

7. AI-Powered Analysis

The tool uses Google's AI model to analyze the resume and provide insights.

- **General Summary:** A brief overview of the resume.
- **SWOT Analysis:** Strengths, Weaknesses, Opportunities, and Threats based on the job description.
- **Match Score:** A percentage score showing how well the resume matches the job description.
- **Improvement Suggestions:** Tips to improve the resume (e.g., formatting, missing sections).

8. User Interface

The tool has two pages:

Page 1: Resume Analysis

- Users upload a resume and enter job details (title and description).
- The tool provides:
  - A general summary of the resume.
  - A SWOT analysis.
  - A match score.
  - Missing skills and recommended Udemy courses.

Page 2: Improvement Suggestions

- Users upload a resume and get actionable tips to improve it.
- A chatbot allows users to ask specific questions about their resume.

Key Features

1. Resume Text Extraction

- Extracts text from uploaded PDF resumes.
- Handles errors gracefully (e.g., if the PDF is corrupted).

2. Skill Matching

- Compares the resume text with a predefined list of skills.
- Identifies missing skills required for the job.

3. AI-Powered Insights

- Uses Google's AI model to generate summaries, SWOT analyses, and improvement suggestions.

By: Anil Matholiya

- Provides a match score to evaluate how well the resume fits the job.

4. Course Recommendations

- Recommends Udemy courses for missing skills.
- Displays course titles and links for easy access.

5. Chatbot for Improvement

- Allows users to ask specific questions about their resume.
- Provides detailed feedback based on the resume content.

How to Use

1. **Upload Your Resume:**
   - Go to Page 1: Resume Analysis.
   - Upload your resume in PDF format.
   - Enter the job title and description.

2. **Analyze Your Resume:**
   - Click the "Analyze Resume" button.
   - View the general summary, SWOT analysis, match score, and missing skills.

3. **Improve Your Resume:**
   - Go to Page 2: Improvement Suggestions.
   - Upload your resume and get actionable tips.
   - Use the chatbot to ask specific questions.

4. **Learn Missing Skills:**
   - If the tool identifies missing skills, it will recommend relevant Udemy courses.
   - Click the course links to explore them.

Example Workflow

1. **User Uploads Resume:**

Uploads a PDF resume for a "Software Engineer" role.

Enters the job title and description.

2. **Tool Analyzes Resume:**

Provides a summary of the resume.

Highlights strengths (e.g., "Proficient in Python") and weaknesses (e.g., "Lacks experience in cloud computing").

Gives a match score (e.g., "80% match with the job description").

3. **User Gets Improvement Tips:**

By: Anil Matholiya

The tool suggests adding a "Projects" section and quantifying achievements.

Recommends Udemy courses for missing skills like "AWS" and "Docker".

4. **User Asks Chatbot:**

Asks, "How can I improve my resume's formatting?"

The chatbot provides detailed tips (e.g., "Use bullet points and consistent fonts").

Requirements

Python Libraries:

- Install the required libraries using:

```
pip install pandas streamlit google-generativeai PyPDF2 langchain
```

Google API Key:

- Obtain a Google API key and enter it in the tool.

CSV Files:

- Ensure skill.csv and udemy_courses.csv are in the correct format and location.

Limitations

PDF Quality:

- The tool may struggle with resumes that have poor formatting or scanned images.

API Key Dependency:

- The tool requires a valid Google API key to function.

Skill Matching:

- The accuracy of skill matching depends on the quality of the skill.csv file.

Conclusion

The Resume Analyzer is a powerful tool for job seekers. It helps users understand how well their resume matches a job description, identifies areas for improvement, and recommends resources to learn missing skills. By following the documentation, even users with minimal coding knowledge can use the tool effectively.

By: Anil Matholiya

Block-by-Block Code Explanation

1. Importing Libraries

python Copy

```python
import pandas as pd
import streamlit as st
from google.generativeai import GenerativeModel
import PyPDF2
import os
import re
from langchain.prompts import PromptTemplate
```

- **What it does**: This block imports all the necessary libraries for the tool to work.
  - `pandas` : Used to handle CSV files (e.g., skills and Udemy courses).
  - `streamlit` : Used to create the web interface.
  - `google.generativeai` : Used to interact with Google's AI model (Gemini).
  - `PyPDF2` : Used to extract text from PDF resumes.
  - `os` : Used to handle environment variables (e.g., API keys).
  - `re` : Used for text pattern matching (e.g., extracting skills).
  - `langchain.prompts` : Used to create structured prompts for AI analysis.

2. Setting Up Streamlit

python Copy

```python
st.set_page_config(page_title="Resume Analyzer", page_icon="📄")
st.title("Resume Analyzer")
st.write("Upload your resume (PDF) and provide job details to analyze your fit for the role.")
```

- **What it does**: This block sets up the Streamlit web interface.
  - `st.set_page_config` : Sets the page title and icon.
  - `st.title` : Displays the main title of the app.
  - `st.write` : Provides a brief description of the tool.

3. Sidebar for Navigation

```python
page = st.sidebar.radio("Select Page", ["Page 1: Resume Analysis", "Page 2: Improvement Suggestions"])
```

- **What it does**: Adds a sidebar to the app with two pages:
  1. **Page 1: Resume Analysis** (for analyzing resumes).
  2. **Page 2: Improvement Suggestions** (for improving resumes).

4. Google API Key Setup

```python
api_key = os.getenv("GOOGLE_API_KEY")
if not api_key:
    api_key = st.text_input("Enter your Google API Key", type="password")
    if not api_key:
        st.warning("Please provide a valid API key.")
        st.stop()
os.environ["GOOGLE_API_KEY"] = api_key  # Set API Key
```

- **What it does**: Checks if the Google API key is provided.
  - If the key is not set in the environment, the user is prompted to enter it.
  - If no key is provided, the tool stops and shows a warning.

5. Initialize Gemini Model

```python
model = GenerativeModel("gemini-pro")
```

- **What it does**: Initializes the Google Gemini AI model for text analysis.

6. Load Skills from CSV

```python
@st.cache_data
def load_skills():
    try:
        skills_df = pd.read_csv("skill.csv")
        skills = skills_df.groupby('Subcategory')['Skill'].apply(list).to_dict()
        if '' in skills:
            skills['General'] = skills.pop('')  # Move unclassified skills under 'General'
        return skills
    except FileNotFoundError:
        st.error("skill.csv not found.")
        st.stop()
    except Exception as e:
        st.error(f"Error loading skill.csv: {e}")
        st.stop()

skills = load_skills()
```

- **What it does**: Loads skills from a CSV file (`skill.csv`) and groups them by category.
  - If the file is missing or has errors, the tool stops and shows an error message.
  - The `@st.cache_data` decorator ensures the data is loaded only once for better performance.

By: Anil Matholiya

7. Extract Text from PDF

```python
def extract_text_from_pdf(file):
    try:
        pdf_reader = PyPDF2.PdfReader(file)
        text = "\n".join(page.extract_text() or "" for page in pdf_reader.pages)
        return text.strip() if text else None
    except Exception as e:
        st.error(f"Error extracting text from PDF: {e}")
        return None
```

- **What it does:** Extracts text from an uploaded PDF resume.
  - If the extraction fails, an error message is shown.

---

8. Extract Skills from Text

```python
def extract_skills_from_text(text, skill_dict):
    text_lower = text.lower()
    extracted_skills = {category: set() for category in skill_dict.keys()}

    for category, skills_list in skill_dict.items():
        for skill in skills_list:
            if re.search(rf"\b{re.escape(skill.lower())}\b", text_lower):
                extracted_skills[category].add(skill)

    return {category: list(skills) for category, skills in extracted_skills.items() if skills}
```

- **What it does:** Compares the resume text with the list of skills and identifies which skills are mentioned.
  - Uses regular expressions (`re`) to match skills in the text.

9. Recommend Udemy Courses

```python
@st.cache_data
def load_udemy_courses():
    try:
        return pd.read_csv("udemy_courses.csv")
    except Exception as e:
        st.error(f"Error loading Udemy courses: {e}")
        return None

def recommend_courses(missing_skills):
    df = load_udemy_courses()
    if df is None:
        return {}

    recommendations = {}
    for category, skills_list in missing_skills.items():
        for skill in skills_list:
            skill_courses = df[df["Title"].str.contains(skill, case=False, na=False) |
                               df["Subtype"].str.contains(skill, case=False, na=False)]
            if not skill_courses.empty:
                recommendations[skill] = skill_courses[["Title", "URL"]].head(3).to_dict(orient
="records")

    return recommendations
```

- **What it does:** Recommends Udemy courses for missing skills.
  - Loads course data from `udemy_courses.csv` .
  - Searches for courses that match the missing skills and displays their titles and URLs.

By: Anil Matholiya

10. Langchain Prompts

python Copy

```
summary_prompt = PromptTemplate(
    input_variables=["resume_text"], template="Provide a **general summary** of the following re
sume:\n\n{resume_text}"
)
swot_prompt = PromptTemplate(
    input_variables=["job_title", "job_description", "resume_text"],
    template="""Analyze the following resume based on the job role and provide a SWOT analysis
(Strengths, Weaknesses, Opportunities, and Threats):\n**Job Title:** {job_title}\n**Job Descript
ion:** {job_description}\n**Resume:**\n{resume_text}""",
)
score_prompt = PromptTemplate(
    input_variables=["job_title", "job_description", "resume_text"],
    template="""
    Provide a **match score (0-100%)** evaluating:
    - **Skill Match**
    - **Experience Match**
    - **Education Match**

    **Job Title:** {job_title}
    **Job Description:** {job_description}
    **Resume:**
    {resume_text}
    """,
)

improvement_prompt = PromptTemplate(
    input_variables=["resume_text"],
    template="""Analyze the resume and provide actionable improvement suggestions.

    Consider:
    - Formatting & Readability
    - Key Sections Missing (e.g., Projects, Certifications, Summary)
    - Use of Action Verbs
    - Quantification of Achievements
    - Grammar & Professionalism
    - ATS Optimization Tips

    Resume:
    {resume_text}
    """
)
```

- **What it does**: Defines structured prompts for the AI model to generate:
  - A general summary of the resume.
  - A SWOT analysis.
  - A match score.
  - Improvement suggestions.

By: Anil Matholiya

11. Extract Match Scores

python Copy

```python
def extract_match_scores(score_response):
    matches = re.findall(r"(\w+)\s*:\s*(\d+)%", score_response)
    score_dict = {k.lower(): int(v) for k, v in matches}

    if len(score_dict) == 3:
        skill_match = score_dict.get("skill match", 0)
        experience_match = score_dict.get("experience match", 0)
        education_match = score_dict.get("education match", 0)
        overall_match = (skill_match + experience_match + education_match) / 3

        return f"""
        **Skill Match:** {skill_match}%
        **Experience Match:** {experience_match}%
        **Education Match:** {education_match}%
        **Overall Match:** {overall_match:.2f}%
        """
    return f"Your resume matches **{score_response}** with the job description."
```

- **What it does:** Extracts and formats the match score from the AI's response.

---

12. Page 1: Resume Analysis

This block handles the functionality for **Page 1**. It allows users to upload a resume, enter job details, and view the analysis.

---

13. Page 2: Improvement Suggestions

This block handles the functionality for **Page 2**. It allows users to upload a resume and get improvement suggestions.

---

14. Chatbot for Improvement

This block provides a chatbot interface where users can ask specific questions about their resume.

How to Make Changes

- **Change CSV Files**:
  - Update `skill.csv` or `udemy_courses.csv` to add/remove skills or courses.
- **Modify Prompts**:
  - Edit the `PromptTemplate` blocks to change the AI's behavior.
- **Add New Features**:
  - Use the existing structure to add new analysis features or pages.

By: Anil Matholiya