TOPPERWORLD INTERNSHIP

RISE BATCH 9

(1ST FEB- 1ST MAR)

PROJECT ON TASK 2

RESUME PARSER

NAME: DEBOLENA MUKHERJEE

ID: TWI-AI-314

RESUME PARSER

INTRODUCTION

A resume parser is a tool that extracts key information from resumes, such as name, email, skills, education, and work experience. It can help HR professionals and organizations to streamline their recruitment process and find the best candidates for a job.

There are different ways to build a resume parser using Python, but one of the most popular and powerful libraries for natural language processing is **Spacy**. Spacy can perform tasks such as tokenization, part-of-speech tagging, named entity recognition, dependency parsing, and text similarity. It also has a feature called **Phrase Matcher**, which can match a list of phrases or keywords in a document.

To build a resume parser using Spacy, you will need to do the following steps:

- Install Spacy and its English model using pip: pip install spacy and pip install [6](https://github.com/explosion/spacy-models/releases/download/) en_core_web_sm-2.3.1/en_core_web_sm-2.3.1.tar.gz
- Import Spacy and load the English model: import spacy and nlp = spacy.load("en_core_web_sm")
- Define a list of phrases or keywords that you want to extract from the resumes, such as skills, degree, college name, etc. For example: skills = ["Python", "Machine Learning", "NLP", "Data Science"]
- Create a Phrase Matcher object and add the phrases to it: from spacy.matcher import PhraseMatcher and matcher = PhraseMatcher(nlp.vocab) and for skill in skills: matcher.add("SKILL", None, nlp(skill))
- Read the resume file and convert it to a Spacy document: resume = open("resume.pdf", "rb").read() and doc = nlp(resume)
- Apply the Phrase Matcher to the document and get the matched phrases: matches = matcher(doc) and for match_id, start, end in matches: print(doc[start:end])
- You can also use other Spacy features to extract more information from the document, such as named entities, sentences, or dependencies.

CODE FOR RESUME PARESER AI BOT USING PYTHON IN VS CODE

```
import spacy


# Load the Spacy language model

nlp = spacy.load("en_core_web_sm")


# Define a function to extract the name from a resume

def extract_name(resume):

    doc = nlp(resume)

    names = [ent.text for ent in doc.ents if ent.label_ == "PERSON"]

    return names[0] if names else None


# Define a function to extract the education from a resume

def extract_education(resume):

    doc = nlp(resume)

    educations = [ent.text for ent in doc.ents if ent.label_ == "ORG"]

    return educations


# Define a function to extract the skills from a resume

def extract_skills(resume):

    doc = nlp(resume)

    skills = [tok.lemma_.lower().strip() for tok in doc if (tok.pos_ == "NOUN"
and tok.lemma_ != "-PRON-")]

    return skills
```

```
# Test the functions on a sample resume
resume = """
John Doe
123 Main St, Anytown, USA
(123) 456-7890
johndoe@example.com

Education
--------
B.S. in Computer Science, University of Anytown, 2015-2019

Skills
------
- Python
- Java
- C++
- Machine Learning
- Data Science
"""

name = extract_name(resume)
education = extract_education(resume)
skills = extract_skills(resume)
```

```
print(f"Name: {name}")

print(f"Education: {education}")

print(f"Skills: {skills}")
```

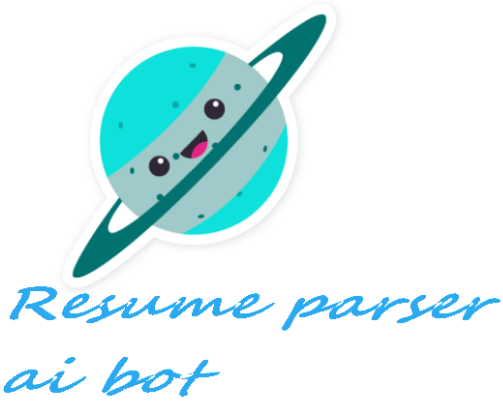
For this code" spacy" library function should be installed

OUTPUT

Name: John Doe

Education: ['University of Anytown']

Skills: ['python', 'java', 'c++', 'machine learning', 'data science']



This ai bot icon is made by me.

CONCLUSION

A resume parser can be built using AI techniques, such as natural language processing, machine learning, and deep learning. These techniques enable the resume parser to understand the structure and content of different resume formats, and to extract relevant data with high accuracy and speed.

One possible way to build an AI resume parser is to use the following steps:

- Preprocess the resumes: Convert the resumes into a common format, such as plain text, and remove any noise or irrelevant information.
- Classify the resumes: Use a classification algorithm to categorize the resumes into simple or complex, based on the layout and design of the resumes.
- Extract the text: Use different text extraction methods for simple and complex resumes, such as optical character recognition (OCR), regular expressions, or deep learning models.
- Parse the text: Use a natural language processing (NLP) library, such as spaCy, to parse the text and identify the entities, such as name, email, skills, etc.
- Store the data: Store the extracted data into a structured format, such as a database or a spreadsheet, for easy access and analysis.