Indeed Scrapper Documentation

Files:

Scrapper_indeed.py

This is the file which scrapes the data from Indeed web portal to search through the jobs according to the skills of the user in his/her resume and job preferences (like job location, type of job and experience level) of the user.

Variables:

- resume_skills= list of skills described in resume
- all_skills= list of skills present in database
- match_threshold = defined for matching skills in user resume and skills in the job description.
- role = job role preference
- Location = job location preference
- no_of_jobs_to_retrieve = number of job postings to be retrieved from indeed
- Data - loaded json file for database connection.
- Final_dict – to store the job description links scrapped from indeed.

Code Segment:

get_job_description(resume_skills, all_skills, match_threshold, role, location, no_of_jobs_to_retrieve, data):

This function is used to scrape the indeed website, the function takes the resume_skills,all_skills, match_threshold, role, location, no_of_jobs_to_retrieve, data as parameters and returns a number of jobs openings which the user wants to see.

First all the jobs postings urls are collected and stored in a list named job_urls.

```
def get_job_description(resume_skills,all_skills, match_threshold, role, location, no_of_jobs_to_retrieve, data):
    options = Options()
    options.add_argument("--window-size=1920,1200")
    options.add_argument('--headless')
    options.add_argument('--no-sandbox')
    options.add_argument('--disable-dev-shm-usage')
    driver = webdriver.Chrome(options=options, executable_path=ChromeDriverManager().install())
    url = "https://www.indeed.com/jobs?"
    #------------------Job perferences(input from user)-----------------------------------#
    data={}
    data["q"] = role
    data["l"] = location
    data["jt"]="parttime"
    data["explvl"]="senior_level"
    #-------------------------------------------------------------------------------------#
    url_parts = list(urllib.parse.urlparse(url))
    query = dict(urllib.parse.parse_qsl(url_parts[4]))
    query.update(data)
    url_parts[4] = urllib.parse.urlencode(query,quote_via=urllib.parse.quote_plus)
    url = urllib.parse.urlunparse(url_parts)
    driver.get(url)
    job_urls = []
    c = 0
    jobcards = driver.find_element_by_id('mosaic-provider-jobcards')
    jobs = jobcards.find_elements_by_xpath("./*")
    print(len(jobs))
    for text in jobs:
        if text.get_attribute('href'):  ### get all the job postings URL'sz
            job_urls.append(text.get_attribute('href'))
            c = c + 1
```

Then the job_urls list is iterated to get the job description and each job description is stored in a dictionary named final_dict with job_url as key. And this dictionary final_dict is returned.

```
            if (c >= no_of_jobs_to_retrieve):
                break
    final_dict = {}
    # ========== Iterate through each url and get the job description ==================================
    for i in job_urls:
        time.sleep(5)
        jobs = []
        driver.get(i)
        job_description = driver.find_element_by_xpath('//*[@id="jobDescriptionText"]').text
        jobs.append(job_description)
        final_dict[i] = job_description

    final_result=ke.get_user_id_to_list_of_job_ids(resume_skills,final_dict,all_skills,match_threshold)

    return final_result
```