Intelligent Job Search Agent built in Python

Following steps should be performed to run the program:

Just run the python file 'shantanu_deshmukh_career_job.py' normally, it will open an intuitive command line user interface as below:

```
Enter your search string
artificial intelligence san jose

Enter no. of results you want to display (value of k)
15

Edit advanced options (clustering and data scrapping depth)?
1. No
2. Yes
2

Do you want similar jobs to be grouped together(Enable clustering)?
1.Yes
2.No
1

Enter the no. of clusters
4

Enter no. of (paginated)results pages in result to be scraped (more pages results in more time)
4

Clear lookup table? 1.Yes 2.No
2

Scratching data over the internet..

Fetching jobs from Indeed...

Fetching jobs from ACM...

Fetching jobs from IEEE...
52 jobs fetched in 3.04 seconds, finding the best match for you..

15 jobs fetched from the lookup table in 3.05 seconds !!

Kindly open url -> file://C:\Users\deshm\Documents\workspace-sts-3.9.0.RELEASE\IntelligentJobFinder\testfile.html in browser to view results


Enter e to exit.. Enter any other key to rerun..
```

Enter appropriate values for each option.

First option asks for the search key word.

Second option is the value of k or the no. of results to be displayed on the results page.

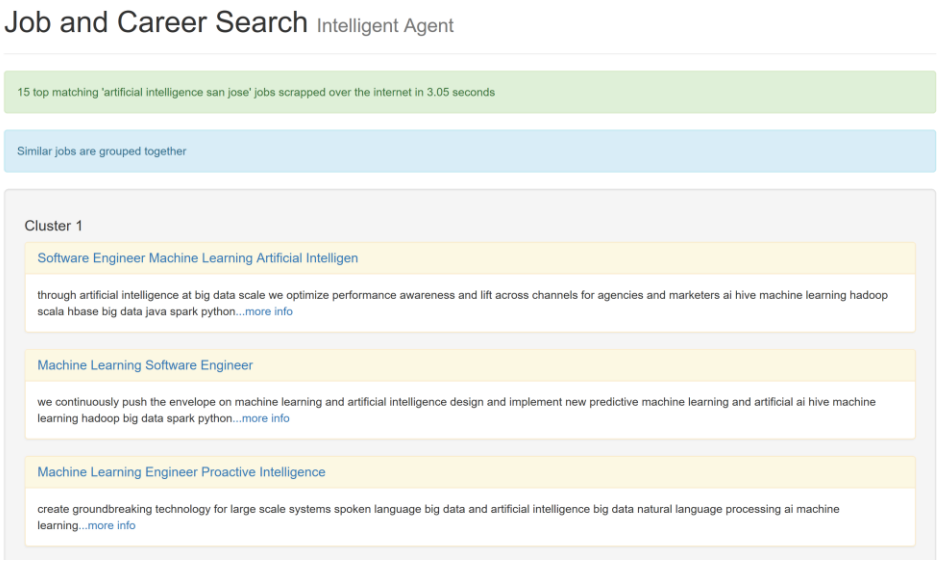
Third option asks for the advanced options, if user says no here only kNN algorithm will be applied and top k results will be displayed.

Fourth and fifth options are for the clustering mode, it sets the no. of clusters. Note that no. of clusters should be less then the value of k. The program will alert the user if wrong value is entered.

Sixth option is about scraping, how deep or till what page in the result should the agent go so as to pull in the data. Having higher value will fetch more data but slow down the agent.

Last option is to clear up the lookup table.

Output of the program looks is as below :



Features implemented in the program:

1. Baseline:
    - ✓ Informative comments
    - ✓ Command-line user interface
2. Satisfactory:
    - ✓ Environment and Agent class implemented
    - ✓ Lookup table created
    - ✓ Only External library used is BeautifulSoup
    - ✓ kNN algorithm used to find nearest matches
3. Baseline:
    - ✓ Persisting the table on every run, so that the table is prebuilt in next iterations
    - ✓ Timestamp protection to avoid stale data in the prebuilt table
4. Good:
    - ✓ Llyod's algorithm used to group similar jobs
5. Excellent:
    - ✓ Output of the program is in HTML