Mining JPL jobs

Abstract

This is a deep dive into more than 700+ NASA's Jet Propulsion Lab job posts to extract insights on the specifications and requirements.

Index Terms

JPL,NASA,Python,Selenium, D3

Contents

I. Introduction

I am a big fan of Jet Propulsion Laboratory (JPL). I have created an interactive post that shows the full time line of all of the JPL missions, and an interactive dashboard to slice and dice the missions. I have been following the job posts from JPL for quite a while now, and created some automated tools to collect the job postings from LinkedIn everyday since the beginning of the year 2020. My ultimate goal is to extract statistics out of the required skills to land a job at JPL. This post is a detailed inspection of 722 JPL job posts.

II. Crawling the Web

About a year ago, I partnered with a web developer to build a webpage for our group at work. I am not a web-developer, and I concentrated on the automated web-testing, which was quite fun! It was based on a Python based Selenium web crawler. I scheduled it to run daily to launch the page and crawl every corner of it to look for failues and generate a report. It also enabled us to check if the page was accessible or not. It would even generate automated emails if the page is not accessible. This was quite an experience. I later modified the tool to collect data from the internet for various projects, one of which is collecting data from a job page.

Let me provide a quick tutorial on running Selenium with Python3. You can find the code in my repository or copy it below.

Python 3 & Selenium browser launch

Hide

email: quarktetra@gmail.com

Find the interactive HTML-document here.

```
browser="Chrome"
#browser="FF"
url="https://www.indeed.com/"

if  browser=="FF":
    from selenium.webdriver.firefox.firefox_binary import FirefoxBinary
    driver = webdriver.Firefox( executable_path='C:\Python3_scheduled\geckodriver.exe')    # get gecko
if  browser=="Chrome":
    from selenium import webdriver
    driver = webdriver.Chrome('C:\Python3_scheduled\chromedriver.exe')    # get chrome driver from http

driver.get(url)
driver.implicitly_wait(10)
```

What this code does is very simple: it triggers an instance of Firefox or Chrome depending on the selection at the top of the script. It is important to note that you will need to download the driver files, see the links in the code, and save them to a folder associated with the path in the code so that Python can launch the browser. Depending on the site you are working on, you may need to log in with you credentials. For example, in the case of LinkedIn, the credentials can be submitted to the correct fields with the following code.

Logging in

Hide

It is very important to note that it is never a good idea to include the credentials in plain text in the code. I always keep my credential encrypted elsewhere and load them via the code. After the log in, one navigates to a URL that includes the search parameter, which will return the list of jobs. It is then somewhat a tedious coding exercise to locate the fields of interest and compile the data. Unfortunately, it is a running target since as the page design changes, the field identifiers may change causing failures in the code. This means that one needs to keep on top of the code, so that it stays functional. I needed to update the code every other month on the average.

I have been very persistent in maintaining the code over the last year, and scheduled the code to run daily. I now have data on 722 JPL job posts, which I will slice and dice below. Let us peek at the (almost) raw data in Tab. II below, which is interactive and searchable. The data is illustrated in Fig. 1 on the right, where each circle represents a job listing (hover over the circles to see more.)

Show 10 entries

Search:

	Type	Level	Function	Title
1	Contract	Entry	Other	channel-marketing-manager
2	Contract	Associate	Management Manufacturing	pricing-analyst-i
3	Contract	Entry	Engineering/Science	electronics-engineer-i
4	Contract	Associate	HR/PR/Legal/Education/Admin	qa-technical-training-specialist-ii-instructor
5	Contract	Associate	Accounting/Auditing/Finance	accounting-specialist-ii
6	Contract	Associate	Management Manufacturing	subcontract-manager-ii
7	Contract	Associate	Other	techonologist-ii-tms
8	Contract	Entry	Engineering/Science	mechanical-engineer-level-1
9	Contract	Entry	Information Technology	experienced-subcontract-compliance-advisor
10	Contract	Associate	Management Manufacturing	experienced-pricing-analyst

Showing 1 to 10 of 722 entries

Previous 1 2 3 4 5 …

JPL jobs posted over the year 2020 and early 2021.

III. Three ways to slice

Just to get an overall feel for the data, it is illustrative to group it using three different attributes:

1. Function,
2. Level,
3. Type .

This will still be a high level review of the data, and it will only provide a feel for the content. I will later create a filter to narrow down the data to what I am mostly interested in. We will also extract some useful statistics. Let's dive in and slice and dice the data.

IV. Functions

The job posts include a field related to the functional group. In the original data there are 39 individual functions, which is a bit too granular for my purposes. I re-mapped them to broader titles.

Show the mapping of the functional groups (Table 2)

Hide

```
## Warning: The ‘.dots‘ argument of ‘group_by()‘ is deprecated as of dplyr 1.0.0.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was generated.

## ‘summarise()‘ has grouped output by ’thefunction’. You can override using the ‘.groups‘ argument.
```

Show 15 entries

Search:

	Function	Function original
1	Engineering/Science	Engineering
2	Information Technology	Information Technology
3	Other	Other
4	Management Manufacturing	Management Manufacturing
5	HR/PR/Legal/Education/Admin	Education Training
6	HR/PR/Legal/Education/Admin	Administrative
7	Research Analyst IT	Research Analyst Information Technology
8	HR/PR/Legal/Education/Admin	Human Resources
9	Manufacturing	Manufacturing
10	Accounting/Auditing/Finance	Accounting/Auditing Finance
11	Customer Service/Sales	Business Development Sales
12	Customer Service/Sales	Finance Sales
13	HR/PR/Legal/Education/Admin	Public Relations
14	Information Technology	Other Information Technology Management
15	Customer Service/Sales	Sales

Showing 1 to 15 of 39 entries

Mapping the functional groups to broader titles
The remapping yields 9 functional groups, as in Tab. IV and Fig. 2.
Show the final functional groups (Table 3)
Hide

Show 15 entries Search:

	Function
1	Information Technology
2	Engineering/Science
3	HR/PR/Legal/Education/Admin
4	Other
5	Customer Service/Sales
6	Management Manufacturing
7	Research Analyst IT
8	Accounting/Auditing/Finance
9	Manufacturing

Showing 1 to 9 of 9 entries Previous

Re-grouped functions.

It is not surprising to see that the top two function categories are Information Technology (205 jobs) and Engineering/Science (201 jobs). The Engineering/Science jobs are more relevant for my background, but before applying any filters, let's slice the data in a different way.

V. Levels

The data can be put into 6 levels, as in Fig. 3a. and Fig. 3b.

The jobs listings are dominated by Entry level (231 instances).

VI. Types

Finally we can take a look at the types of the jobs: There are 4 types, as in Fig 4a and 4b. Most of the jobs are Full-Time with 538 instances.

VII. Keywords

As we are crawling the posts, we can not only collect the meta data, but also the inner text of the job information. In the HTML page, the job description lives in a particular element. I locate that element and grab its inner html. I save it for later analysis.

The cloud on the right shows the words with sizes scaled with their frequency of appearances in the job postings.

One may be inclined to omit the obvious words, such as "JPL," however, that would be a mistake. The frequency of the word "JPL" appearing serves as a gauge. For example, we see that the word "data" appears almost as frequently as "JPL" which underscores the importance of data, and presumably data science, to JPL.

VIII. Languages

As we have the full text content of all the postings, let;s take a quick look at frequency of programming languages appearing in the posts. To this end, I came up with a list to search for:

The languages are roughly ordered by the amount of experience I have with them. For the languages/tools listed after 'Knime,' my experience is limited. However, this I will decide if I need to invest time to learn more about them if they show up frequently. Figure 6a and b show the ones from my list appearing frequently in the job posts.

It is clear that Python dominates the languages JPL looks for.

IX. To be continued

This is a post in progress. I have a lot to add: What other qualifications do the look for the most? How about ML related tools? Do I meet these requirements? I am also building a ML based algorithm that will tell me if a job post is a good fit for me by matching my skills with the requirements. Stay tuned, and stay safe.