Computer Science Job Postings in New York, Massachusetts, and Connecticut: A Study on Job Requirements and Salary Trends

Abstract

Website scraping is the process of collecting data from websites and using the data to find and analyze patterns. In this paper, we demonstrated an effective way to perform web scraping on Indeed.com to find and analyze computer science job postings in New York, Massachusetts, and Connecticut based on the job requirements and salary. This paper will demonstrate how the data is collected, along with how data should be effectively cleaned up. Once that is completed, we will focus on trends and patterns we will aim to look for with the data collected. The purpose of this paper is to provide valuable information to employees in the field and graduate students, empowering them to make informed decisions and avoid being taken advantage of in the job market.

Introduction

Web scraping is the process of gathering raw data from the internet to create a dataset usually for analysis. The process of web scraping can be done in many ways such as manually collecting data, incorporating a browser extension, or even directly from a site through its application programming interface (API) [1]. The significance of web scraping lies in its ability to gather information from the internet. However, without proper filtering, classification, or analysis of that information, the data collected could bear no meaning. The objective of web scraping is to capture data and present it in a more organized way. Dedicated web scrapers, such as Agenty (a Chrome extension for web page data extraction) and JobSpy, offer more efficient alternatives to manual scraping [4].

Other options include proprietary web scraping tools like OutScraper, which is a paid service, as well as developing custom web scrapers using programming languages such as Python, JavaScript, or Java. Creating custom scrapers provides the flexibility to customize data collection based on your needs but requires programming and networking skills [1]. Despite the advantages, web scraping comes with challenges. Many websites oppose scraping and may implement captchas or even ban IP addresses. Alternatives for obtaining data include requesting permission from the website or accessing it through their API, often at a cost. Each approach has its merits depending on use cases and the volume of data needed.

The information stored in the dataset can be formatted in various ways with one popular way being in CSV format which is compatible with Microsoft Excel and Google Sheets. Another very common way is with JSON format which is the most typical for rendering data on websites and web apps [5]. The choice of how information is stored ultimately depends on the intended use. CSV formats are user-friendly, as applications like Microsoft Excel and Google Sheets can easily interpret and display the data. Additionally, popular libraries for data interpretation and analysis, such as Python Pandas and R, readily support the reading of CSV files. This makes CSV a convenient choice for data exchange and analysis purposes.

Once data is collected, it is ideal to first perform a data cleanup as an important step in the data preprocessing phase. This cleanup can be done manually or using tools such as Python Pandas, or dplyr library in R, along with other relevant libraries. The goal is to address inaccuracies, identify and handle outliers that could impact the analysis, and manage missing or duplicate data [6]. Utilizing Natural Language Processing (NLP) for data analysis, especially when dealing with unstructured textual data. Python's Natural Language Toolkit (NLTK) is a commonly used library for NLP tasks. It enables the extraction of meaningful insights from large bodies of text, such as job listings' descriptions, to glean valuable information about a job position [1]. NLTK can be applied to remove stop words (commonly used words or phrases), tokenize text to break it down into individual units like words or sentences, and facilitate quicker extraction of meaning. Additionally, plotting libraries like ggplot2 and Matplotlib can help visualize the results of the analysis.

The importance of web scraping varies depending on the specific use case. You can employ web scraping to collect information to help you with looking at Price comparisons, weather data, business details, government data, market analysis, job market, and real estate. These are just a few of the applications for web scraping.

In our research study, we will focus on Indeed.com and utilize web scraping techniques. We will explore how to employ data analysis techniques to closely examine the demand for Computer Science jobs in Connecticut, Massachusetts, and New York. Our study will concentrate on trends and patterns in the job market and on relevant skills to gain insights into the tech industry in these three states for both current industry professionals and undergraduate students nearing graduation. This analysis aims to highlight important factors, including preferred locations, sought-after skills by current employees, and expected salary ranges. The goal is to provide valuable information to employees in the field and graduate students, empowering them to make informed decisions and avoid being taken advantage of in the job market.

Methodology

Data collection for the project involved using an open-source software JopSpy to collect and scrape specific data from Indeed.com. The software was written in Python, utilizing BeautifulSoup along with Tag, ThreadPoolExecutor, and Future libraries to bypass Indeed's anti-bot mechanisms and extract the correct information. The program was designed to work not only on Indeed.com but also on the LinkedIn Job Board and ZipRecruiter, with a primary focus on Indeed.com.

JobSpy helps you obtain the following data from the sites. However, during our data collection, we observed that some of the information gathered was not specific to Indeed.com but also included data that would be found in LinkedIn Jobs and ZipRecruiter. So some of the data for those categories was missing from the dataset. Ultimately, it is capable of gathering “Job_Url, Site, Title, Company, Location, Job_Type, Date_Posted, Interval, Min_Amount, Max_Amount, Currency, Is_Remote, Num_Urgent_Words, Benefits, Emails, Description”. Of the following categories, we utilized “Title, Location, Min_Amount, Max_Amount, Is_Remote, Description” for our analysis.

After collecting our data set, we manually filtered out all job listings that were either too broad for “Computer Science” or not directly related to it. Examples of excluded positions include “Professors, Deans, Science Teachers, Adjunct Professors, Managers, Sales managers, Janitors," and similar roles, full list on our GitHub. Our criteria for filtering were to focus on job listings where the position either directly involved the application of computer science knowledge or where at least 60% of the job responsibilities require proficiency in computer science. Examples of job listings included “Project Managers, Business Analyst, Software Engineer, Information Technology (IT)", and similar roles that align with the field of computer science.

In our data analysis, we used two distinct approaches to examine trends within our datasets. For the creation of word clouds and looking at the frequency of programming languages for each state we utilized Python, along with Panadas and MatplotLib libraries. The word cloud generation and frequency of programming languages specifically focused on the "Description" category, as we determined that it would be ideal to extract meaningful data once the data has been cleared. Whilst analyzing and creating figures for, is the job remote, comparison of salary between locations, and proportion of job positions in each city. We employed the use of R, along with ggplot2 and dplyr libraries, which we saw as practical for our use case.

The data cleaning process involved leveraging the Natural Language Toolkit (NLTK), from which we used the following libraries, nltk.corpus’s stopwords and brown. These libraries are a collection of commonly used words in the English language. We specifically utilized the brown library, which consists of a million words in the English language, and selected the 500 most frequently occurring words. Additionally, we created custom stopwords, which included many of the companies names and other words that the stopwords and brown library missed.

Using the Pandas library from Python, we called the “Description” column, removed any special characters and missing values (NA), and then tokenized all the words in the column. After tokenizing, we implemented a loop that will check each word in each row of the column against the stopword libraries. If a match was found, the word was removed from the row. Saving the new cleaned-up data into a CSV file for further use.

For the word cloud generation, we utilized the Matplotlib library along with the WordCloud library. Initially, we loaded the cleaned-up data CSV file and then we combined all rows into a single string, and then using the WordCloud library we generated a word cloud consisting of 200 words.

Generating the programming frequency per state, utilized a similar approach. Starting with the cleaned-up data CSV file, and using Matplotlib and Pandas libraries. We first created an array consisting of programming languages we wanted to look out for, which came to about 20 popular languages. Using Pandas, we looped through each row and counted each time a language would show up in the "Description" column. Then utilizing the Matplotlib library, we generated a pie chart illustrating the top 10 programming languages based on their frequency in the job descriptions.

In the analysis of whether the job is remote per state, we employed R and utilized ggplot2 and dplyr to create the graphs. The process began by reading the original CSV file into R studio and then we removed any Na values in the “Remote” category. Subsequently, we used the factor function to transform True or False values into "Remote" or "Not Remote" labels. Following this, we generated a data frame containing these "Remote" or "Not Remote" values. Using ggplot2, we crafted a relative frequency bar graph to visually represent the percent of remote versus non-remote jobs.

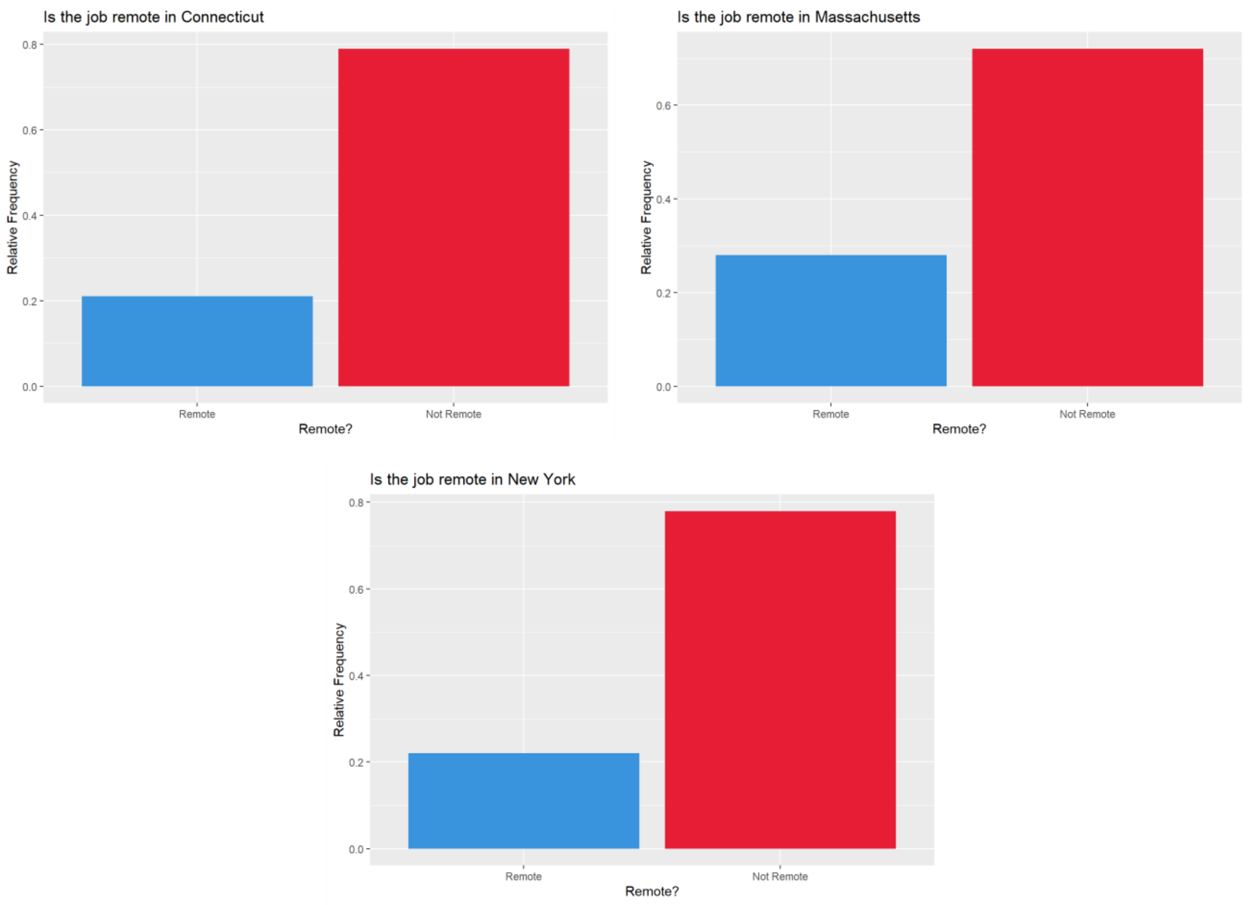
In analyzing the comparison of salaries between locations in their corresponding state, we initially generated a data frame consisting of values from the "Location" and "Min_Amount" categories. We created a table to calculate the frequency of each location in the dataset while excluding locations with fewer than 20 postings. Additionally, we filtered out the keywords “State, USA”, which were any listings that didn’t have a county, along with “USA”, and any NA values using the dplyr library. We also filtered out any minimum salary value less than 10000, which removed all zeroes and all hourly salaries in our data. Utilizing ggplot2, we constructed a side-by-side boxplot illustrating the distribution of minimum salaries versus location for the top locations per state.

In the analysis of Title vs Location per state, we initiated by creating a data frame containing the "Location" and "Title" categories. Then we filtered out any location that didn’t have at least 20 listings, and then keywords such as “USA”, “State, USA”, and any NA values. Following this, we generated a table for the "Title" category, removing any job titles with fewer than 24 job listings. Subsequently, we created a conditional table that accounted for job listings by frequency for each location. Utilizing ggplot2, we generated a relative frequency stacked bar graph using the conditional table to showcase the distribution of job titles across different locations within each state.

Full source code can be found on our GitHub page - https://github.com/Deuos/Indeed-Job-Listings-Senior-Research

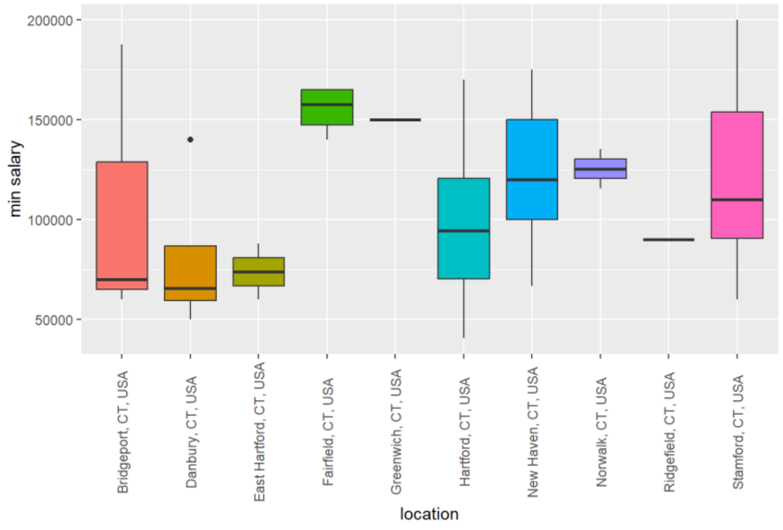
Results

Using JobSpy, we were able to scrape ~50% of the job listings for the keyword "Computer Science" in Connecticut, Massachusetts, and New York. Which resulted in about 500 listings for Connecticut, 2000 for Massachusetts, and 2500 for New York put combined into a CSV file. However, we encountered our first problem when the number of listings reached high levels. JobSpy began experiencing issues, resulting in a 429 error, which indicated that Indeed.com started to block the request. Additionally, we noticed that a couple of rows for the “Description” category were missing values entirely, which we concluded was due to Indeed.com’s anti-bot mechanism. This is our analysis based on the data we collected.

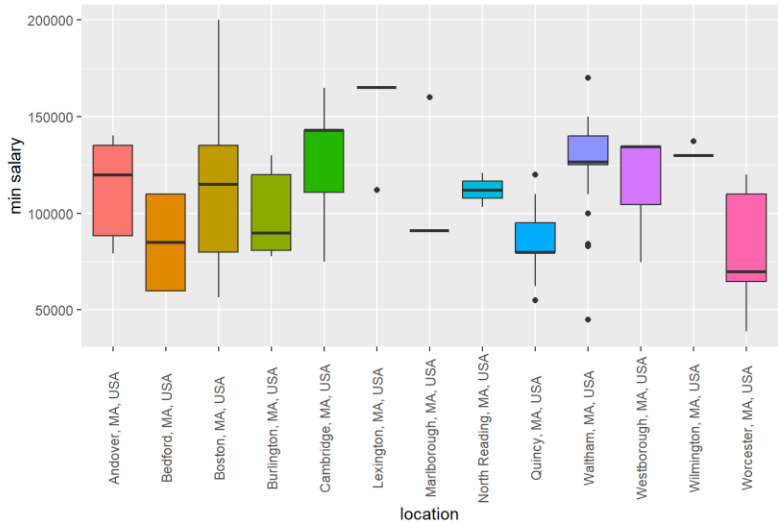


Figure(s) 1. Is the job remote? Shows the relative frequency of job postings within each state for the job type. The y-axis represents the percentage of jobs for each category while the x-axis represents the categories of whether a job is remote or not.

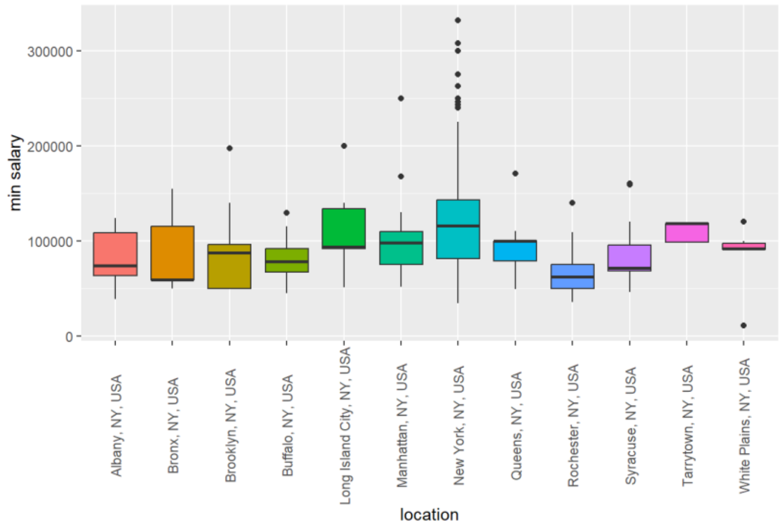
Comparison of Salary's between locations in Connecticut



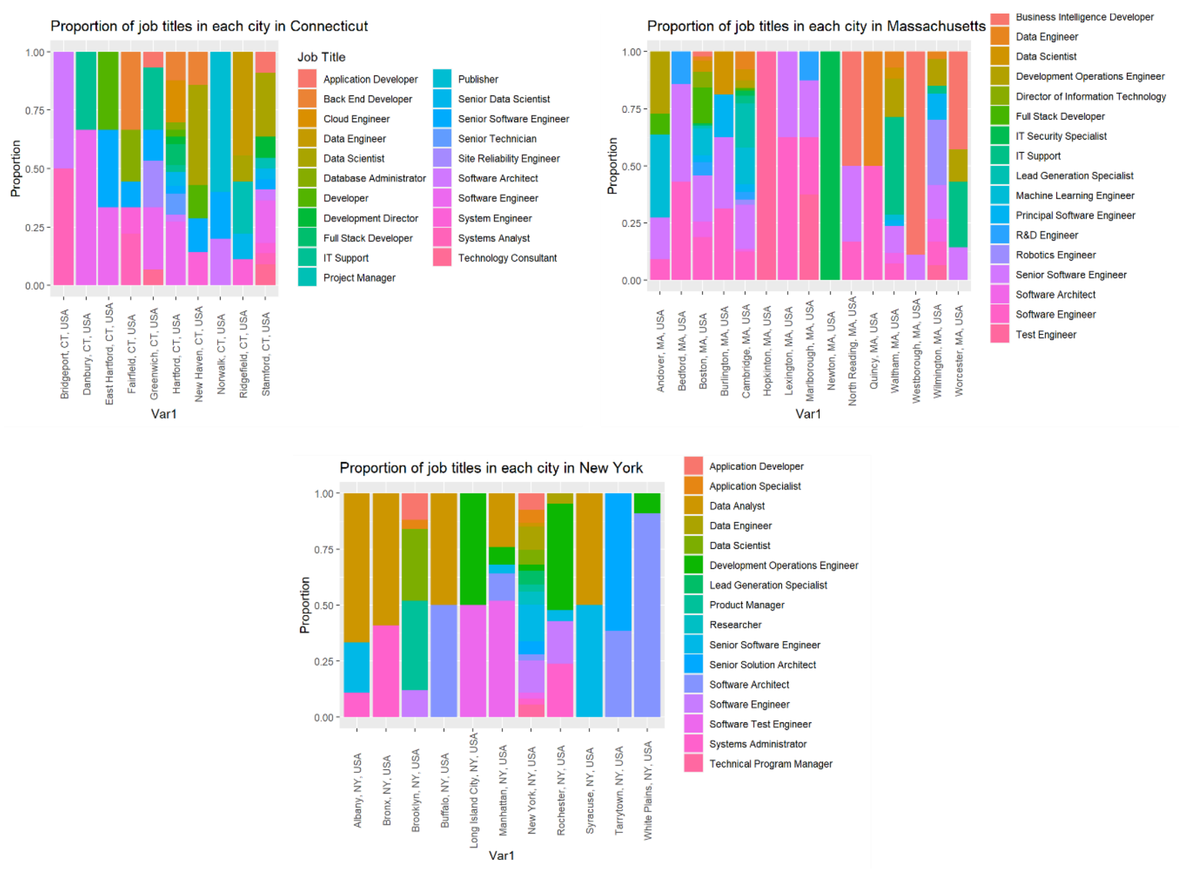
Comparison of Salary's between locations in Massachusetts



Comparison of Salary's between locations in New York



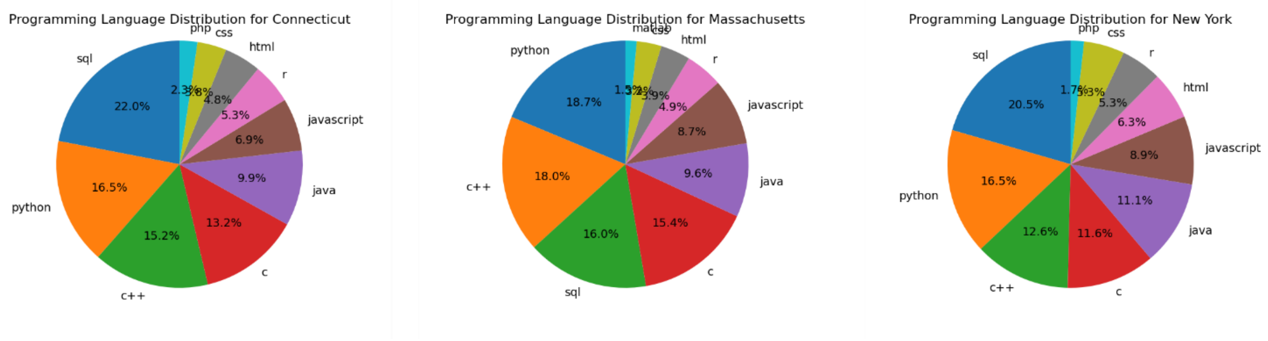
Figure(s) 2. Comparison of salary between locations. Shows side-by-side boxplots of the minimum salaries of jobs for cities within each state. The y-axis represents the minimum salaries for jobs while the x-axis represents the specific city location within the state.



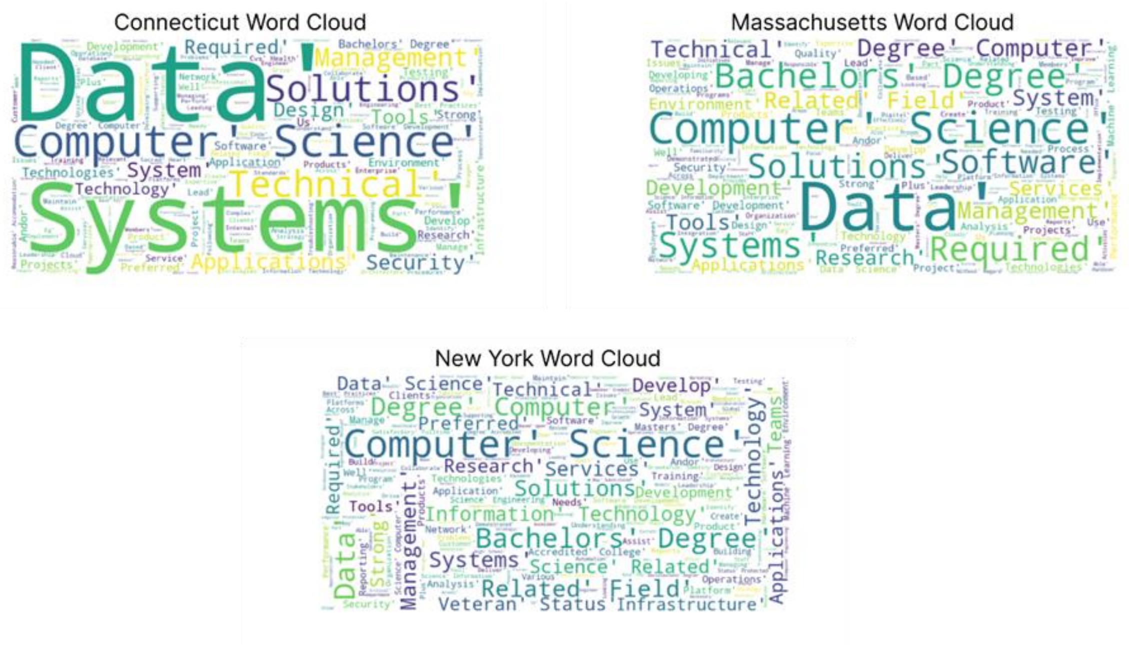
Figure(s) 3. Proportion of job titles in each location. Shows a relative stacked bar graphs of job titles for cities within each state. The y-axis represents the percentage of jobs for each job title while the x-axis represents the specific city location within the state.

Fig. 1 shows that Connecticut and New York have roughly the same relative frequency of job postings that are remote at a little over 20% while Massachusetts has a higher relative frequency of job postings that are remote at a little under 30%. **Fig. 2** shows that New York seems to have the lowest variability in minimum salary based on location as the median shown in the side-by-side boxplots are relatively close while Massachusetts and especially Connecticut have a higher variability as the median minimum salary in Fairfield is over $150,000 while the minimum salary in Danbury is under $75,000.

Fig. 3 shows the proportion of job titles within each city with all three showing trends of software architect and software engineer being in high demand.



Figure(s) 4. Pie chart of skills. Shows the percentage of jobs listings a specific programming language for each state. Displays the skills per location.



Figure(s) 5. Word cloud of descriptions. Shows the most commonly used words in the job descriptions as the more frequently a word appears the larger it becomes in the word cloud for each state.

Fig. 4 shows SQL is the most commonly listed programming language in New York and Connecticut while Python is the most commonly listed programming language in Massachusetts. We created a pie chart to see the percentage of skills that are most sought after and if those same programming languages are requested similarly throughout the various locations. **Fig. 5** shows bachelor's degree is the most required level of education in all three states with systems and security also being common within the word cloud.

Discussion

In the pedagogical results software engineer and data scientist were the two most frequent jobs, python and java were the two most listed programming language, and machine learning and bachelor's degree were some of the most frequent words in their word cloud [1]. From our results the programming language requested was different as in New York and Connecticut the most common one was SQL followed by Python while Massachusetts was Python followed by C++ (**Fig. 2**). This shows that if you feel stronger in SQL you should focus your job search on New York and Connecticut over Massachusetts since the first two states have over 20% of job postings requesting SQL. One commonality between our results is that we also saw that bachelor's degree was the most common degree as shown in the word cloud (**Fig. 5**). **Fig. 4** shows how some of the most common job titles posted consisted of software engineer and data scientist which matches the results from the articles web scraping. Our results were similar for the most part to the article although the programming languages requested were slightly different.

The significance of these results is that it can help college students and employees find jobs that better fit them within the computer science industry based on their skills, interests, and requirements. For example, if someone's biggest requirement for a job was that it is remote, they can see that Massachusetts has the highest relative frequency of job postings that are offered as remote so they can focus more of their attention to that state for a higher chance of finding their remote job (**Fig. 1**). One of the limitations we faced with our work was that we weren't able to scrape specifically entry level jobs. Another limitation is that Indeed has measures in place to try and prevent web scraping of their website which ended up causing issues such as when we scraped over 500 job postings as the descriptions stopped getting properly extracted. Indeed also blocked us from scraping more than 50 job postings at a time and made it so we had to wait a while between each scrape, or we'd get errors, timed out, or blocked.

You can extract data directly from a site's application programming interface (API) which could be the first part of possible extensions and future works for this project by figuring out how to get data from a site's API in order to gather more results more efficiently [1]. However, potential issues for this would be that a site's information may be broken down into multiple sections and may not even allow one to save a copy of the site's data meaning depending on the site's level of protection this may not be possible [2]. We could also try scraping for specifically entry level jobs in order to get more specific data. Further extensions can involve comparing the specific jobs to one another in terms of stuff such as their salary and programming language required instead of the broader comparisons we made. Another extension of this work can involve going beyond just the job titles and skills by looking at the skill sets demanded in combination. For example, in the labor construction industry, the three most frequent dual skills required made up 50% of the job postings requiring dual skills [3]. This shows how certain combinations of skills can be in more demand, which can help people know what is sought after in order for them to learn to increase their employability.

Author Contributions

JYL designed the study. KP developed the model, performed the experiment, and collected the data. JYL analyzed the data. JYL and KP wrote the paper.

References

[1] S. Lunn, J. Zhu, and M. Ross, "Utilizing Web Scraping and Natural Language Processing to Better Inform Pedagogical Practice," presented at the IEEE Frontiers in Education, Uppsala, Sweden, Oct. 2020. doi: 10.1109/FIE44824.2020.9274270.
[2] S. de S. Sirisuriya, "A Comparative Study on Web Scraping," pp. 135–140, 2015.
[3] H. J. Oh, S. Chang, and B. Ashuri, "Investigation of Skill Sets for Multiskilled Labor Development in Construction," presented at the ASC 2021. 57th Annual Associated Schools of Construction International Conference, 2021, pp. 100–107. doi: 10.29007/tsk3.
[4] R. Diouf, E. N. Sarr, O. Sall, B. Birregah, M. Bousso, and S. N. Mbaye, "Web Scraping: State-of-the-Art and Areas of Application," in *2019 IEEE International Conference on Big Data (Big Data),* Los Angeles, CA, USA: IEEE, Dec. 2019, pp. 6040–6042. doi: 10.1109/BigData47090.2019.9005594. K

[5] U. Nandwani, R. Mishra, A. Patil, and W. Siddiqui, "Data Analysis by Web Scraping using Python," *International Journal for Research in Engineering Application & Management (IJREAM)*, vol. 07, pp. 12–15, 2021. K

[6] D. Dasari and P. S. Varma, 'Employing Various Data Cleaning Techniques to Achieve Better Data Quality using Python', in *2022 6th International Conference on Electronics, Communication and Aerospace Technology*, Coimbatore, India: IEEE, Dec. 2022, pp. 1379–1383. doi: 10.1109/ICECA55336.2022.10009079.