Jobs Database – Computer Vision

INFO6105 AI Skunkwords Project

Yunan Shao

Abstract

Looking for jobs or internships seems a task of its own and the search is no longer based on sole fulfillment of the required job skills, but a lot of networking and recommendations is involved around too. The amount of work involved in finding the correct job builds a great amount of anxiety among the job seekers and the recruiters who want the right talent for their company.

There are two concerns that are to be addressed here. First, matching the job seekers with the right employers and second, provide guidance to aspiring job seekers on the skills that are in demand so that they can build them to stay relevant in the job market.

The job providers and job seekers form a large amount of data which provides for many interesting trends for analysis and interpretation to make the most of data available.

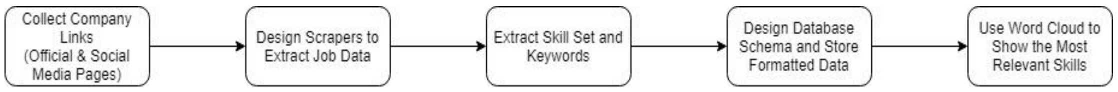
With the data currently available from the seekers and providers, these pitfalls can be fixed. The presence of information on job skills, salaries and user tendencies in many existing websites such as Indeed, LinkedIn, Glassdoor etc. can be utilized to match people to positions which may seem simply impossible without using AI to analyze data.

The jobs database would be a one stop solution to reduce the job search and talent acquisition stress levels. Artificial intelligence (AI) and machine learning can be utilized for complex task of matching work to talent so that it is efficient and less resume spamming.

1. Introduction

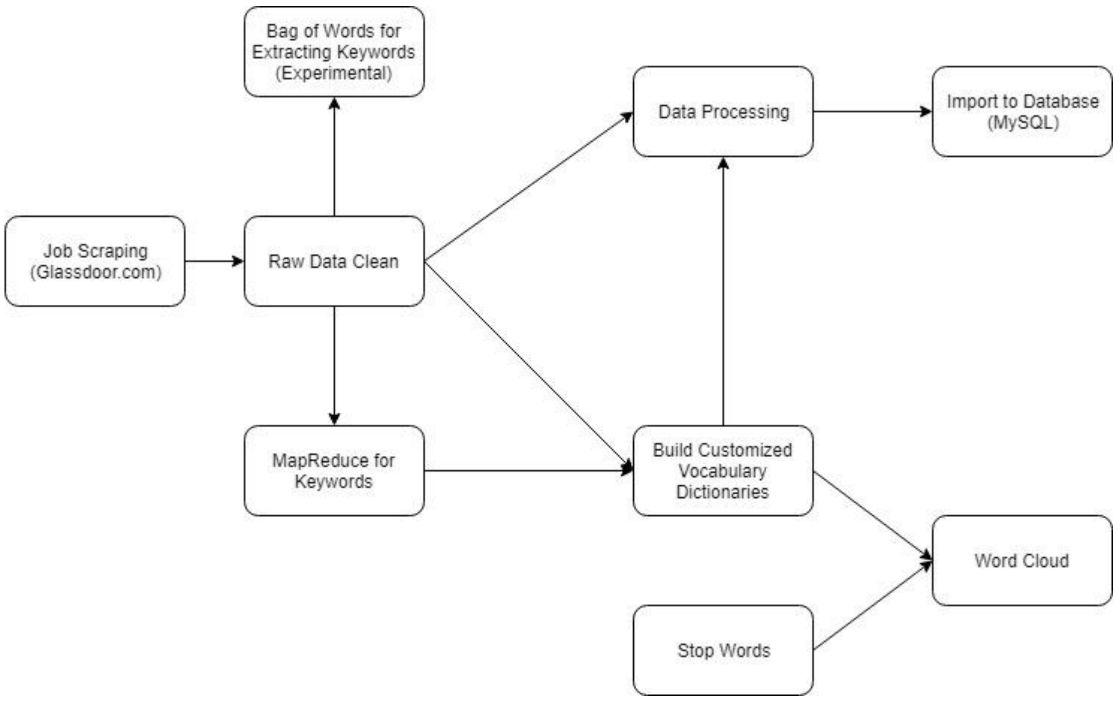
Jobs Database is one of the NEU AI Skunkwork Projects. It is a project that covers multiple topics in different courses including Data Scraping, Data Processing, Database Design and Machine Learning. The basic steps from original instructions are:

1. Collect company links (official website and social media pages).
2. Use scrapers to collect the jobs from company list.
3. Process the scraped data and extract key words.
4. Design database schema and import the formatted data into database.
5. Create word cloud to show the most relevant skills in the domain.


Collect Company Links (Official & Social Media Pages)
Design Scrapers to Extract Job Data
Extract Skill Set and Keywords
Design Database Schema and Store Formatted Data
Use Word Cloud to Show the Most Relevant Skills

Original Workflow

However, I have found some problems with the original pattern and have modified my workflow to satisfy the ultimate goal: **Finding the most relevant skills in order to find a job in the domain.**


Bag of Words for Extracting Keywords (Experimental)
Data Processing
Import to Database (MySQL)
Job Scraping (Glassdoor.com)
Raw Data Clean
MapReduce for Keywords
Build Customized Vocabulary Dictionaries
Word Cloud
Stop Words

Current Workflow

I have also created some machine learning models after completing the

requirements in order to make this project more relevant to my course work.
I will explain and compare the methods I use, then conclude the findings for each step in the following sections.

2. Implementations - Basic

2.1. Scraper

As I worked on this project individually, I don’t have enough time to collect the links required in the instructions and I want to make the scraping part as automated as possible. I collect every set of data using scrapers.
In the beginning, I built a scraper for getting a company list from angel.co. There are other websites which contains information of companies such as Linkedin and Crunchbase. However, the structure of Linkedin doesn't support searching companies by domain very well (can't find useful result for computer vision at least). Crunchbase is a good resource and the web structure is easier for scraping. But it requires monthly payment for data access. Then, I use another scraper for searching the listed jobs for the companies from glassdoor.com. I think one website is already enough because using multiple websites could result duplicate records. However, there are two main problems in this pattern:
1. Social media pages don't contain information about jobs.
2. The jobs listed by company may not be relevant to the domain.
The final result I want to get from the raw data is a set of jobs with the desired 'professional' skills. The jobs listed by company could be sales or other titles that are not relevant to the domain.
After inspecting the data on web pages. I used one scraper only for keyword searching for multiple times. The advantage is I can get related jobs and it is not necessary that the companies list them are related in the domain. For example, Apple doesn't focus only in computer vision domain and if I searched the company by domain, it may not appear in my company list. But it's AI, camera team still have listed jobs which are related to this domain.
I use selenium webdriver to stimulate the human operation on webpages

and parse data from web elements through browser. The advantage of this approach is I can collect all the listed jobs by key word searching. But it is slower because I need to add random sleep time to prevent robot detections of the website and it needs extra configurations to deal with the pop-up windows and login information. The other approach I used in early stage is sending requests using requests library and parsing the response body with beautifulsoup. It is faster than selenium because it processes the request and response directly. The down side is, I cannot do operations such as 'next page' if I search with keywords instead of company names.

The combined raw data I collected contains about 2000 records and the attributes of jobs are title, company name, location, salary, description.

2.2. Data Clean & Processing

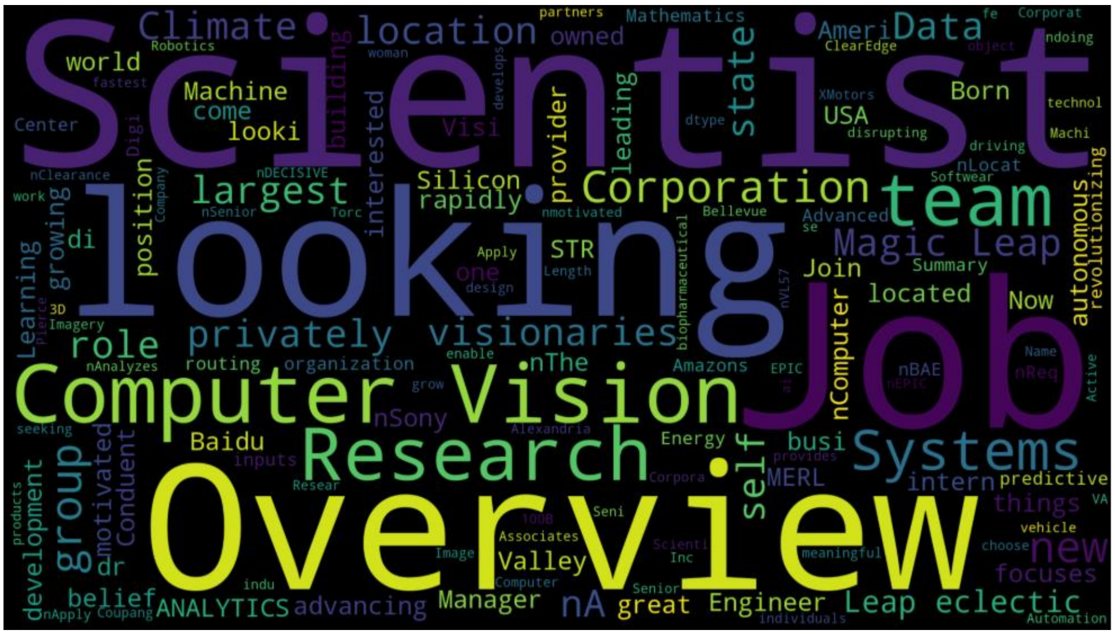
The original raw data set that I got has about 2000 jobs by combining the result of 'Computer Vision', 'Computer Vision Engineer' and 'Computer Vision Scientist'. But there are still many jobs are not related to the domain such as frontend programmer. So, I performed an extra filter on the collected dataset and get a final version with about 800 jobs. I extracted the programming languages, technology related terms and education level requirements in the descriptions using pre-defined list of words then store the combined data into MySQL database. Extra keywords can be extracted using the same pattern.

2.3. Word Cloud

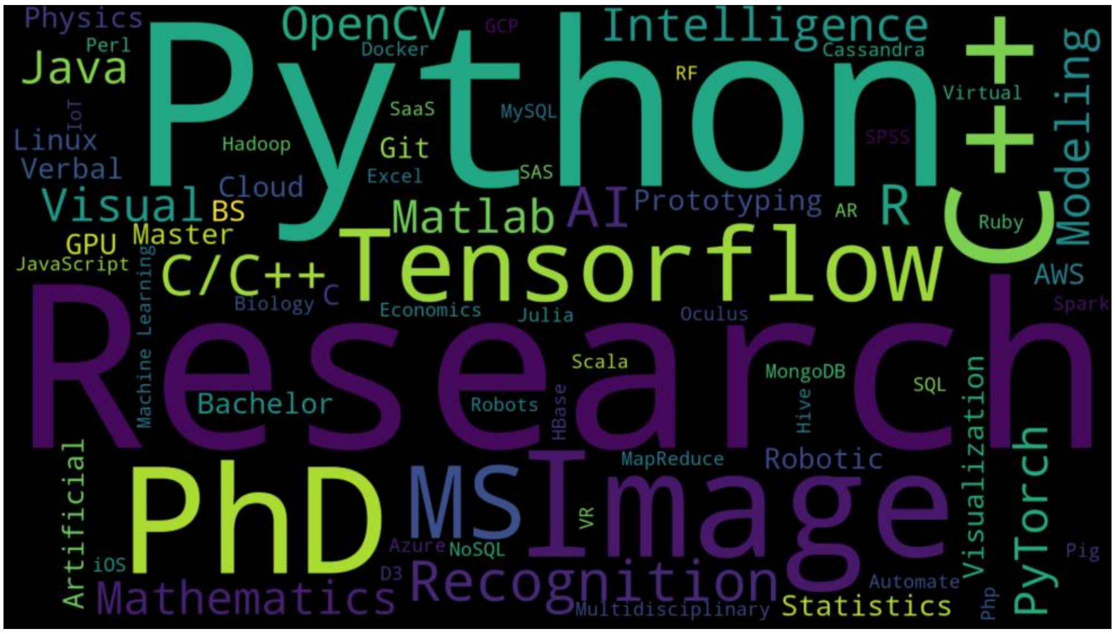
A tag cloud (word cloud, or weighted list in visual design) is a novelty visual representation of text data, typically used to depict keyword metadata (tags) on websites, or to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or color. This format is useful for quickly perceiving the most prominent terms and for locating a term alphabetically to determine its relative prominence. When used as website navigation aids, the terms are hyperlinked to items associated with the tag.

We want to use word cloud to get a visualized overview of what are the related skills and keywords for the jobs in the domain. The basic pattern I use is: 1. Tokenize the descriptions 2. Stem/Lemmatize the tokens 3. Remove Stop Words 4. Create the word cloud.

The problem of this approach is that there are still many irrelevant words in the description after removing the stop words. Professor Brown suggested that I can check online resources such as Wikipedia page to get the related topics of the domain. I followed this pattern and built extra vocabulary dictionaries in order to get the related keywords.



Word Cloud Before



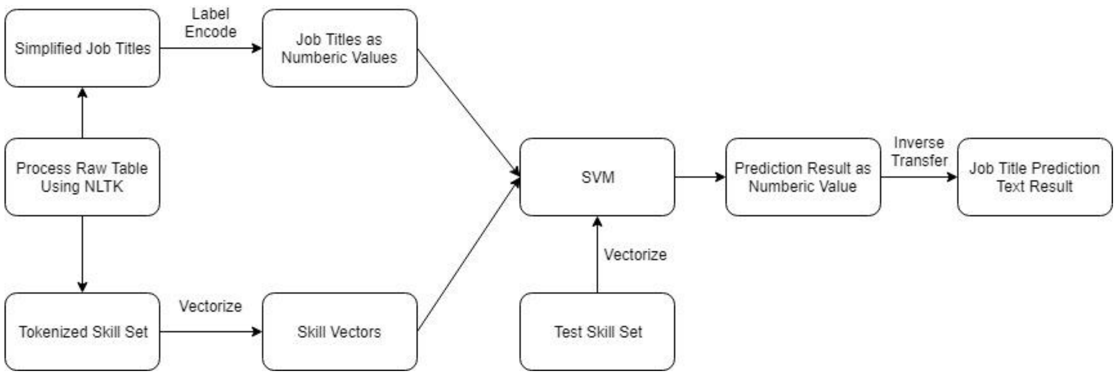
Word Cloud After

I created two method for creating the word cloud. The second one is using map reduce to get the count of words but the problem is still similar to the problem of the first method. There are too many extra words in the descriptions.
There is another way using Neuro-linguistic Programming and it is in the machine learning section.

3. Implementations – Machine Learning

3.1. Classification Using SVM

I created a simple classification model using Support Vector Machine to predict the job title using a set of skills. The reason I chose SVM is that the target variable is not linearly separable which means logistic regression won't work well.


Simplified Job Titles
Label Encode
Job Titles as Numberic Values
Process Raw Table Using NLTK
SVM
Prediction Result as Numberic Value
Inverse Transfer
Job Title Prediction Text Result
Vectorize
Tokenized Skill Set
Vectorize
Skill Vectors
Test Skill Set

Job Title Prediction Model

The result of the cross-validation for this model is about 0.35 which is not a good score. The reason could be:
1. Dataset is not large enough.
2. The job titles are not normalized enough in order to create accurate groups.

I think the second reason is more important but I don't have enough time to inspect every record to make a better method to normalized the job titles.

3.2. Skill Extraction

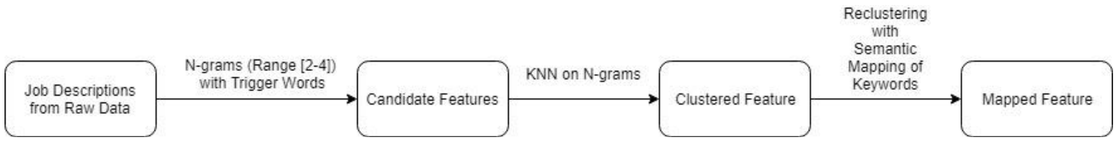
After finishing all requirements of this project, I start to think about how to improve the extraction of keywords. Because the current method I use takes a lot of time for inspecting the data. The answer I want to know is “How to extract the keywords in a more convenient way without wasting too much time collecting the information from the dataset?” The method I used to create the word cloud is frequency based, it is heavily affected by irrelevant words that appear many times. After searching for information of different NLP models, I chose the Bag of Word. Bag of Word uses TF-IDF (Term Frequency-Inverse Document Frequency) to measure the relevance. [1] Generally speaking, it gives weight to each word, not frequency.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$$tf_{i,j} = \text{number of occurrences of } i \text{ in } j$$
$$df_i = \text{number of documents containing } i$$
$$N = \text{total number of documents}$$

TF-IDF Formula 1

However, I didn’t finish with the complete model because of the time constrain. I will just explain the operations using the workflow diagram I designed.


Job Descriptions from Raw Data
N-grams (Range [2-4]) with Trigger Words
Candidate Features
KNN on N-grams
Clustered Feature
Reclustering with Semantic Mapping of Keywords
Mapped Feature

Steps:

1. Use N-grams with range that starts with trigger words such as “education”, “skill” to build a large set of n-grams.
2. Use KNN on stemmed n-grams set, which will generate clusters of different categories such as skill, knowledge, experience (currently finish and stop at this step)
3. Re-clustering each cluster using semantic mapping of keywords/tags
4. Match the skill tags to job description

Some of the categories like skill might need further clustering to get the final result.

5. Conclusions

The basic requirements and goals have been achieved properly and final outputs are

1. Scraped data in as csv format
2. A relational database with all the collected jobs
3. A word cloud shows the keywords in Computer Vision Domain

The machine learning part is designed by myself for showing my understanding of the concepts in Data Science Class. There are more powerful approaches can be used to design the skill extraction patter such as Word2Vec [2] and Nonnegative Matrix Factorization (NMF). [3]

Reference:

1. https://skymind.ai/wiki/bagofwords-tf-idf
2. https://skymind.ai/wiki/word2vec
3. http://mlg.postech.ac.kr/research/nmf
4. https://en.wikipedia.org/wiki/Tf%E2%80%93idf
5. https://github.com/natmod/glassdoor-scrape
6. https://github.com/rodrigosnader/angel-scraper
7. https://github.com/datamusing/employment_skills_extraction
8. https://github.com/2dubs/Job-Skills-Extraction

Appendix:

GitHub Link:

https://github.com/INFO6105-Spring19-02/project-ml-husky