Efficient Job Search Using Clustering

Aman Shrestha, Nishith Atreya

Introduction

There are an overwhelming number of candidates applying for a limited number of jobs in the job market. Therefore, it has always been a challenge for students to look for jobs when they are graduating. Many people along with students are competing in the workforce without much work experience and very little professional network which makes it harder to land a job than someone who has been in the industry for several years. Due to the huge amount of information that can be found online, job applicants can be confused, dissatisfied and can easily lose their confidence during their job search. Therefore, it is crucial to develop an efficient system which is tailored to each individual which can increase their efficiency and reduce stress while applying for jobs.

Inorder to tackle the problem of the diverse requirements of different job applications, this project tries to cluster them together so that it's more efficient to search based on skills, location and other factors. This helps job applicants narrow their search and efficiently apply to places which are more likely to hire them.

We work on a dataset which consists of thousands of job openings in indeed.com from companies all around the US. We preprocess the data by cleaning and replacing the irrelevant values by average of the attribute or categorizing it as an unknown value. Irrelevant attributes have been removed from the dataset to make it easier to work with the dataset. Then, we used several clustering algorithms based on the attributes in the dataset like location, salary, job description etc. If any two job openings have similar attributes and features, they'll most likely be aggregated in the same cluster. Moreover, by using multiple clustering algorithms, we will generate a wide variety of clusters which have very different features than clusters generated from a single clustering algorithm. Now using these clusters, we try to optimize the result of all the clustering techniques by merging all of the outputs. The merging will be based on the intersection of all the clustering outputs. Job opportunities that are found to be in optimum agreement with other outputs will be included in the final result.

Related Works

There are many different recommendation systems used in data mining. Our main challenge was finding the right model that fits the data available well.

Bo Li, Yibin Liao and Zheng Qin have stated that there are "a lot of different machine learning techniques used in different recommendation systems, such as Naive Bayes classification, decision tree, k-means clustering and improvement, and so forth". They have mentioned that "the idea of hybrid recommendation approach gave them inspiration to combine different approaches to implement a movie recommendation." Bo Li, Yibin Liao and Zheng Qin, used DBSCAN, affinity propagation, hierarchical clustering, and random clustering for their movie recommendation using Pre-Computed Clustering. [Bo Li et al. 2014]

Harsh Jain and Misha Kakkar explain that “Hybrid, as the name suggests, is the amalgamation of various types of recommendation systems into one. It can provide a more accurate range of recommendations as it depends on multiple systems and can compute various factors in one go.” [Jain and Kakkar 2019]

Zhou, Liao, Chen and Ge have worked on a job recommendation algorithm for graduates in China using applicant data from campus records. Zhou, Liao, Chen and Ge “propose a personalized preference collaborative filtering recommendation algorithm (P2CF), which is a job recommendation framework with hierarchical structure” [Zhou, Q. et al. 2019].

Nilashi M, Salahshour M. et. al proposed a method where they developed a hybrid recommendation system based on PCA, ANFIS and Fuzzy SOM. Although these aren’t what has been used in this project, the authors have highlighted that “their experiments confirmed that the proposed recommendation method significantly improves predictive accuracy of multi-criteria CF measured by standard accuracy metrics”. This further supported the approach of a hybrid system for recommendation. [Nilashi M. et al. 2016]

Based on the idea of hybrid models, this project also combines multiple clustering algorithms. The different clustering algorithms used for this project are inspired by the work by Putri, Leu and Seda in their project “Design of an Unsupervised Machine Learning-Based Movie Recommender System”. They have used “K-Means algorithm, birch algorithm, mini-batch K-Means algorithm, mean-shift algorithm, affinity propagation algorithm, agglomerative clustering algorithm, and spectral clustering algorithm" to recommend different movies [Cintia Ganesha Putri, D. et al. 2020]. This project uses some of the clustering algorithms suggested in this project. These models have been used in the job dataset in contrast to the movie dataset that they worked in.

Methodology

Dataset:

For this project, the dataset was retrieved from kaggle. User Elroy scrapped Indeed.com, an American employment-related search engine to find different features from multiple job postings [Elroy 2018]. The features that were present in the dataset are listed below:

- Job_Title:Title of Role
- Link:Web Link of Job Posting
- Queried_Salary:Salary Range of the Job Posting (Estimated/Actual if available)
- Job_Type:3 Categories of Job Types - data_scientist, data_analyst, data_engineer
- Skill:List of desired skills on indeed site
- No of Skill:Count of the number of desired skills
- Company:Company that posted the job posting
- No of Reviews:Number of Reviews for the Company
- No of Stars:Ratings for the Company
- Date Since Posted:Number of days since the job was posted - if less than a day, will be rounded up to a full day

- Description:Web scrape of part of the job description
- Location:State the job opening is located in
- Company_Revenue:Annual revenue of hiring company
- Company_Employees:Employee count of hiring company
- Company_Industry:Industry of hiring company

This description is also provided by Elroy in the dataset that can be downloaded. [Elroy 2018]

Data Preprocessing:

From the above features, the features listed below are used in the clustering algorithm:

- Queried_Salary: This data has been categorized into different ranges by user Elroy [Elroy 2018]. The given ranges are: '<80000', '80000-99999', '100000-119999', '120000-139999', '140000-159999', '>160000'. OnehotEncoder in python was used to encode this feature.
- Job_Type: For the dataset being used, the available Job_Type are: ‘data_scientist’, ‘data_analyst’, ‘data_engineer’. OnehotEncoder in python was used to encode this feature.
- Skill: This feature has a list of skills that are required for a job. User Elroy has already converted some values to different columns [Elroy 2018]. However, few other skills not included yet were required, therefore, a script used that would find the most common words in this column. Among these skills, a few ones were subjectively chosen based on their popularity, relevance and frequency in the column . They were:

  Database, Python, R, Java, C/C++, Tensorflow, JavaScript, Ruby, Jira.

These were converted to a column and every row which had these skills or an equivalent variant had an integer 1 in the corresponding column. For ‘Database’, variants of ‘sql’ and ‘DB’ were searched.

- Number of Reviews: This is an integer value of the number of reviews. The nan values were replaced by the median of the column. Minmaxscaler was used to normalize this feature.
- Number of Stars: This is a float value which represents the number of stars for the job posting. The nan values are replaced by mean of the column. Minmaxscaler was used to normalize this feature.
- Description: This is a string value which contains the description for the job. Similar to what was done for the ‘Skill’ feature, a script was written to find keywords in this description. The keywords looked for were:

  BA/BS/Bachelor/Bachelor's, Master/Master's, Ph.D/ Phd/ Doctorate, cloud/aws/gcloud/azure, design/designing, research/researching, analytic/analytics/analysis, finance/financial/finances, lead/leadership/leaders.

  Columns were created for each of these keywords and a value of 1 was put in the corresponding column if the description contains any variants of the column title.
- Location: User Elroy had created different columns for this already and the same columns were used. [Elroy 2018]
- Company_Revenue: This is a string value. There are four possible values for this feature: ‘Less than $1B(USD)’, ‘$1B - $5B’, ‘$5B- $10B’ and ‘More than $10B(USD)’. For the

rows that values didn't exist for, a value of 'Unknown_company_revenue' was kept. OnehotEncoder in python was used to encode this feature.

- Company_Employees: This is a string value. There are two possible values for this feature: '10,000+', 'Less than 10,000'. For the rows that values didn't exist for, a value of 'Unknown_company_employees' was kept. OnehotEncoder in python was used to encode this feature.
- Company_Industry: This is a string value. There are 6 values in this one: 'Consulting and Business Services', 'Internet and Software', 'Banks and Financial Services', 'Health Care', 'Insurance' and 'Others'. This was converted to columns by user Elroy in the original dataset itself. [Elroy 2018]

Clustering algorithm:

To cluster the data available in the most efficient way, different clustering algorithms have been used in this project. They are described below:

K-Means clustering algorithm: This algorithm clusters the data based on centroids whose centers are defined as the mean value of the points. One of the main challenges for this algorithm is finding the right number of clusters. For this, the elbow method was used where K-Means was used with different number of clusters and was used to find the one which gives the biggest difference in sum of squared errors. When running from cluster number 10 to 100, the following graph (Fig: 1) was obtained. There wasn't an obvious choice as to what number might be the best. Therefore, the number of clusters was selected to be 35 which was approximately the closest value for this dataset.

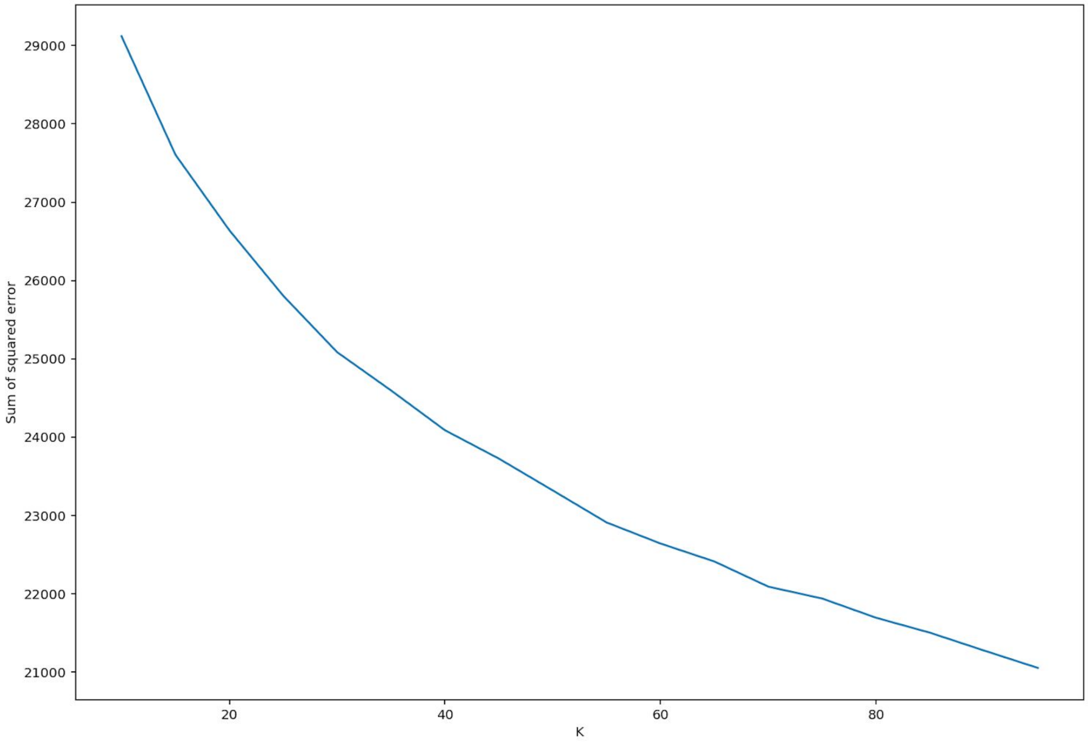


Fig 1: Result of using elbow method with number of clusters 10 to 100.

Once the number of clusters was selected, the model was fitted with the prepared data to run the K-Means algorithm. This resulted in 35 clusters (selected previously) with a maximum number of 252 in a cluster and minimum of 48 in a cluster.

Mini Batch KMeans clustering algorithm: The algorithm Mini batch K-Means was developed as a modification of the K-Means algorithm. It uses mini-batch to reduce time in very complex and large-scale calculations of datasets [Cintia Ganesha Putri, D. et al. 2020].This method gives slightly different results. The same number of clusters, 35, as for the K-Means algorithm was used for this algorithm. This algorithm resulted in a maximum number of 282 in a cluster and minimum of 45 in a cluster. This is similar to what was obtained with K-Means .

Birch Algorithm: Balanced iterative reducing and clustering using hierarchies (birch clustering) uses a hierarchical data structure that calls CF-tree for increment and dynamically clusters data points [Cintia Ganesha Putri, D. et al. 2020]. For consistency in the number of clusters, the same number, 35 was used to fit this model. This algorithm resulted in a maximum number of 378 in a cluster and a minimum of 78 in a cluster.

Agglomerative clustering: This algorithm organizes objects into a hierarchy using a bottom-up or top-down strategy, respectively. Agglomerative methods start with individual objects as clusters, which are iteratively merged to form larger clusters [Han, J. et al. 2012]. For consistency in the number of clusters, the same number, 35 was used to fit this model. This algorithm resulted in a maximum number of 354 in a cluster and a minimum of 37 in a cluster.

Affinity Propagation: Affinity Propagation is an algorithm based on message passing between data points. In affinity Propagation, "Each item being clustered sends messages to all other items informing its targets of each target's relative attractiveness to the sender. Each target then responds to all senders with a reply informing each sender of its availability to associate with the sender, given the attractiveness messages that it has received from all other senders. Senders absorb the information, and reply to the targets with messages informing each target of the target's revised relative attractiveness to the sender, given the availability messages it has received from all targets. The message-passing procedure proceeds until a consensus is reached on the best associate for each item, considering relative attractiveness and availability" [Thavikulwat 2008]. Unlike the algorithms previously used in this project, this model doesn't require a pre-defined number of clusters. A parameter "preference" has been changed to -35. "Preferences for each point - points with larger values of preferences are more likely to be chosen as examples. The number of examples, i.e. clusters, is influenced by the input preferences value."[Pedregosa et al. 2011] With this, the algorithm resulted in 173 clusters with a maximum of 76 in a cluster and a minimum of 4 in a cluster.

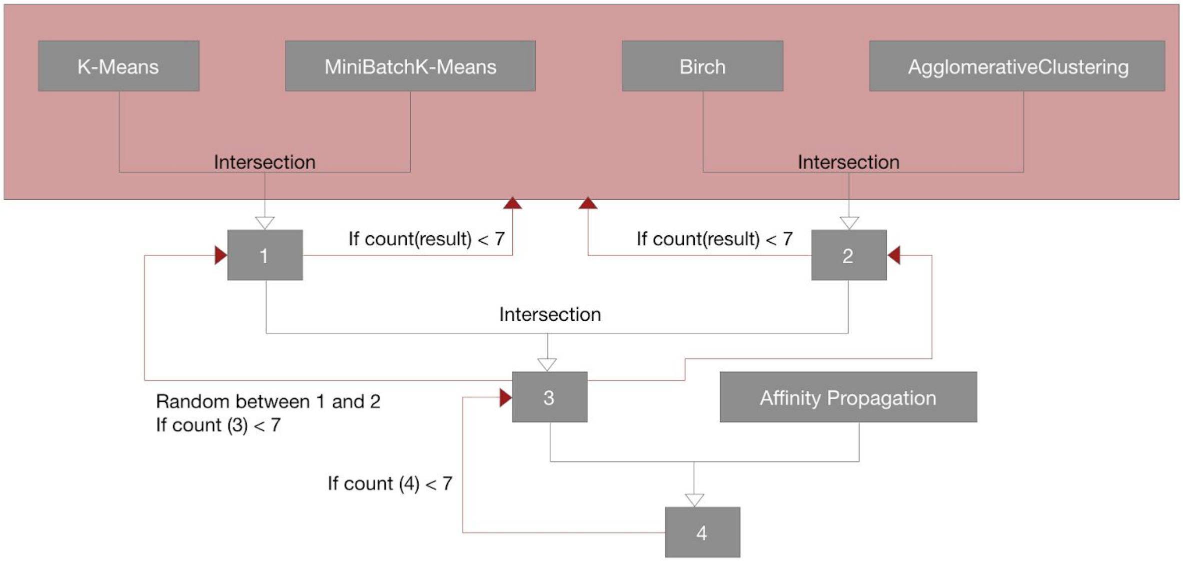
Combining clustering algorithms:

The results of these clustering algorithms were intersected to find a final result of matching jobs. In case the count of the resulting list was less than 7, a smaller intersection between the fewer clustering algorithms was randomly picked.

Experiment

For a better understanding of how the project is working, every row in the previous dataset was put through the algorithm and we found the final intersected list for each row.

The information was changed to a format that can be used in the models. Each model put the given row of information in a particular cluster. Inorder to find the best result, an intersection was taken of the results of clusters given by K-Means and Mini Batch K-Means (1). Similarly, an intersection of the values in the list was taken from results from Birch and Agglomerative clustering (2). Using the result from (1) and (2), an intersection list was formed (3). Then, the intersection between (3) and result from affinity propagation was found (4). This final intersection, (4) would be the final output. However, there were cases where no intersection was shown. In such cases, a minimum value of 7 was sought for. Therefore, result from (3) was selected in that instance. If the list was still smaller than 7, a random list between (1) and (2) was selected. If the list was still smaller than 7, then a random list between the results of the individual clustering algorithms was selected.



Note: The red arrows are followed only after 4 is reached and the count is less than 7.

Fig: How the clustering algorithms are combined

With this, we found a minimum cluster of 7 recommendations and maximum of 378 recommendations. A subjective view of these lists supports our project. However, a better metric will be necessary in the future.

Conclusions

The task of job search can be intimidating because of the overwhelming amount of information and vast requirements for an applicant. In order to ease this, it’s important to have a system so that job search can be made efficient specifically for college students. This project tries to find a model which helps in clustering jobs together so that applicants can find a job that’s best fit to what they are looking for.

With the help of K-Means, Mini Batch K-Means, Birch, Agglomerative clustering, and Affinity Propagation clustering algorithm, we were able to generate separate lists of recommended jobs based on the working mechanism and pros of each algorithm. But the key difference between our approach and the existing approaches is that we try to optimize the result by finding the maximum intersection between results i.e the resulting output will consist of the list of same decisions made from different clustering techniques.

For future work, more time and resources should be spent in finding quality data and data preparation so that it can lead to better and accurate results. For further work, we can use clustering techniques to generate clusters based on applicant’s data like age group, experience in the industry etc and integrate with our current approach, which might lead to even better result. This will help us generate better clusters and thus come up with better recommendations for the applicants. The dataset we used is primarily for Data Science, but this approach could be extended to other industries and job categories.

References

Elroy. 2018. Indeed Dataset - Data Scientist/Analyst/Engineer). https://www.kaggle.com/elroyggj/indeed-dataset-data-scientistanalystengineer.

Bo Li, Yibin Liao, and Zheng Qin. 2014. Precomputed Clustering for Movie Recommendation System in Real Time. *Journal of Applied Mathematics* 2014 (2014), 1–9. DOI:http://dx.doi.org/10.1155/2014/742341

Jain, H. and Kakkar, M. 2019. Job Recommendation System based on Machine Learning and Data Mining Techniques using RESTful API and Android IDE. 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (Jan. 2019). DOI:https://doi.org/10.1109/CONFLUENCE.2019.8776964

Zhou, Q. et al. 2019. Job recommendation algorithm for graduates based on personalized preference. CCF Transactions on Pervasive Computing and Interaction. 1, 4 (Nov. 2019), 260–274. DOI:https://doi.org/10.1007/s42486-019-00022-1.

Cintia Ganesha Putri, D. et al. 2020. Design of an Unsupervised Machine Learning-Based Movie Recommender System. Symmetry. 12, 2 (Jan. 2020), 185. DOI:https://doi.org/10.3390/sym12020185.

Nilashi M. et al. 2016. A New Method for Collaborative Filtering Recommender Systems: The Case of Yahoo! Movies and TripAdvisor Datasets (2016).

Jiawei Han, Micheline Kamber, and Jian Pei. 2012. *Data mining: concepts and techniques*. Elsevier/Morgan Kaufmann, Amsterdam.

Thavikulwat 2008. Affinity propagation: a clustering algorithm for computer-assisted business simulations and experimental exercises. In Developments in Business Simulation and Experiential Learning: Proceedings of the Annual ABSEL conference (Vol. 35).

Pedregosa et al. 2011. Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830. Retrieved May 7, 2020 from http://jmlr.org/papers/v12/pedregosa11a.html