JOB TITLE RECOMMENDATION SYSTEM

Dhana Veeramachaneni, Srinagh Chalasani

***Abstract*—We concentrated our research on profiles unique to the IT industry due to the abundance of data on profiles of diverse persons available online. The goal is to collect data from employees in various IT roles, as well as their associated skill sets, experience, and company, and use data augmentation methods to improve the amount and quality of this data. Collaborative filtering algorithms are popular with this type of recommendation system, according to extant research on the subject. After preprocessing the data and obtaining the appropriate data set structure, this project focuses on training the data set using several machine learning methods. A comparison of the performance, primarily accuracy, of these trained and tested models was undertaken, and two models that performed pretty well were picked as the best fit for the data set. Support Vector Machines, Random Forest classifiers, K Nearest Neighbors, Decision trees, Gaussian Naive Bayes, Multinomial Naive Bayes, and Multi-layer perceptron are among the methods that were taught. Support Vector Machines and Random Forest Classifier were found to be the most appropriate classifiers for the dataset out of these. We also scanned a handful of resumes, collected the skill set, ran both of these classifiers through them, and forecasted the roles for each of these resumes.**

I. Introduction

"There is an abundance of data available in the digital world and if it is harnessed effectively and correctly it can provide terrific insights" -Gian Filgoni.

We decided to work on a job recommendation system after starting our thought process at this point and given our shared exposure to the IT business. As part of this project, we looked at a data set that included an individual's role, such as web developer or data scientist, skills, including such various programming languages, experience, and company, and used various machine learning algorithms to compare accuracy and other performance metrics, and choose an appropriate model that predicts a suitable role based on current qualifications.

Furthermore, we attempted to develop a script that goes through resumes, collects the skills set, and feeds it to our trained model, which predicts a significant roles.

II. Related Work

There are numerous research papers and projects available online, but the paper "Implementation of an Automated Job Recommendation System Based on Candidate Profiles" piqued our interest. In this paper, the authors aim to implement a recommendation system for job hunting, for which they compared two collaborative filtering algorithms given a data set. We took up this challenge and started to train and assess a few more algorithms to see whether they could outperform the original. Because data is such an important aspect of this, we wanted to see if we could collect a large amount of real-time data utilizing web crawling services like LinkedIn and Indeed. Furthermore, whether we can correctly anticipate work position or future company based on finite attributes. We limited this study to data linked to the IT industry; nonetheless, we did have some issues with the dataset, which was distinct from prior studies performed online, and we tried various classification algorithms that haven't been thoroughly examined with recommendation systems. Collaborative filtering techniques are more prevalent with these types of issue statements in general.

Advisor: Yuzhen, Ye

III. Data Collection

Web crawling with Python's wonderful soap and lxml modules was our first method of data collection. This was tested on job search engines such as LinkedIn and Indeed. However, because many public sites had added security elements, this original technique did not generate a lot of data. As a result, we had to get around the CAPTCHA in order to access these sites, which proved to be difficult. Moreover, if the number of pull requests exceeded a specified threshold in a set amount of time, the session would time out. We were looking for a dataset that included attributes such as Skills, JobTitle, Experience, and CompanyName. The data sets that were available on the internet lacked the necessary features. As a result, we did data augmentation and generated extra data based on the minimal data we had gathered via web crawling (skills and company names).

In addition, there is a method called 'makeRegression' from the Scikit learn library. It will generate data that has a correlation with a set of informative features given a set of features. Rather than using the built-in method, we wanted to try data augmentation and see what kind of outcomes we could get. This enriched data was used to train and evaluate our models.

IV. Data Pre-Processing

We cleaned this data collection and turned it from a raw format into a role with a corresponding list of abilities, experience, and location. We gathered a unique set of these skills and inserted them as columns, with a value of one if the talent is present for that record and 0 otherwise. We next thoroughly examined it, visualizing the data as a percentage of each of the data set's jobs, the top 10 skills for each of these roles, the proportion of a skill in each role, and so on. The data collection now comprises roughly 22,000 records, 47 columns, 44 skills, and 7 profiles.

V. Approaches

To train and analyze the accuracy and evaluation matrices, we identified a set of supervised classification methods.

1) **Naive Bayes:** Given the job title, a Gaussian Naive Bayes classifier was trained using the pre-processed data, assuming skills are independent of one another. Because there are many missing prior probabilities in the distribution, the findings may not be obvious. Due to the lesser quality of our data collection, it's possible that it doesn't strictly follow the Gaussian distribution (36%).

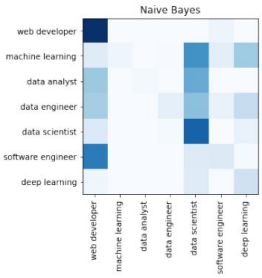

Fig. 1. Naive Bayes Classifier - Confusion Matrix

	precision	recall	f1-score	support
web developer	0.71	0.02	0.03	662
machine learning	0.81	0.08	0.15	713
data analyst	0.31	0.77	0.44	614
data engineer	0.22	0.19	0.21	137
data scientist	0.83	0.04	0.08	629
software engineer	0.37	0.12	0.18	832
deep learning	0.37	0.96	0.53	967
accuracy			0.36	4554
macro avg	0.52	0.31	0.23	4554
weighted avg	0.54	0.36	0.25	4554

Table 1. Naive Bayes Classifier - Evaluation Matrices

a) *Naive Bayes with Laplace Smoothing:* Laplace smoothing was employed to overcome the classification error caused by missing prior probabilities, and the overall accuracy was enhanced to 50%. Multinomial Naive Bayes takes into account a feature vector in which each term indicates a frequency. Every probability estimate will include a small-sample correction, or pseudocount. As a result, no probability will ever be zero. The pseudo count is 0 in this case.

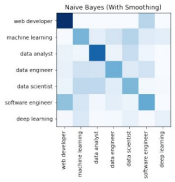

Fig. 2. Naive Bayes Classifier with Laplace Smoothing - Confusion Matrix

2) **K Nearest Neighbors:** Even though the calculation complexity is involved in the prediction that it has to traverse through the entire training set to find the K nearest neighbors by calculating the distance against each

	precision	recall	f1-score	support
web developer	0.58	0.64	0.61	662
machine learning	0.52	0.39	0.45	713
data analyst	0.37	0.36	0.37	614
data engineer	0.40	0.06	0.10	137
data scientist	0.37	0.38	0.37	629
software engineer	0.44	0.41	0.43	832
deep learning	0.62	0.80	0.70	967
accuracy			0.50	4554
macro avg	0.47	0.43	0.43	4554
weighted avg	0.49	0.50	0.49	4554

Table 2. Naive Bayes Classifier with Laplace Smoothing - Evaluation Matrices

training data point, the K Nearest Neighbors Classifier was able to achieve an overall accuracy of 67%, which is better than Naive Bayes. The distance measure is Minkowski, and the number of neighbors is n which is 5. We set the value of K to an odd number to avoid misinterpretation.

Fig. 3. K Nearest Neighbors Classifier - Confusion Matrix

	precision	recall	f1-score	support
web developer	0.77	0.67	0.72	662
machine learning	0.49	0.72	0.58	713
data analyst	0.50	0.44	0.47	614
data engineer	0.48	0.21	0.29	137
data scientist	0.60	0.55	0.58	629
software engineer	0.69	0.72	0.70	832
deep learning	0.88	0.80	0.84	967
accuracy			0.65	4554
macro avg	0.63	0.59	0.60	4554
weighted avg	0.67	0.65	0.65	4554

Table 3. K Nearest Neighbors Classifier - Evaluation Matrices

3) **Decision Trees:** The decision tree classifier was able to achieve a 66% overall accuracy. However, a decision tree may not be the optimal model for this situation because slight changes in the training data can cause the tree and classification results to diverge significantly.

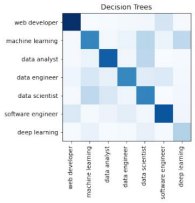

Fig. 4. Decision Trees - Confusion Matrix

4) **Neural Networks:** Over 1000 iterations, a multi-layer perceptron with two hidden layers of size 100 and 200

	precision	recall	f1-score	support
web developer	0.73	0.70	0.71	662
machine learning	0.72	0.56	0.63	713
data analyst	0.45	0.57	0.51	614
data engineer	0.46	0.26	0.33	137
data scientist	0.54	0.57	0.55	629
software engineer	0.70	0.73	0.71	832
deep learning	0.82	0.85	0.84	967
accuracy			0.66	4554
macro avg	0.63	0.60	0.61	4554
weighted avg	0.67	0.66	0.66	4554

Table 4. Decision Trees - Evaluation Matrices

was trained. From a batch size of 100 and a learning rate of 0.01 the model parameters were optimized using stochastic gradient descent. We're employing the ReLU activation function because it doesn't saturate when the weighted total of inputs is positive. This resulted in a 53% accuracy.

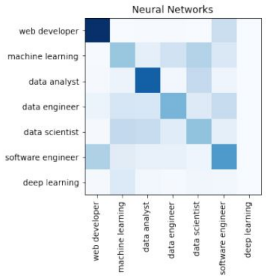

Fig. 5. Neural Networks - Confusion Matrix

	precision	recall	f1-score	support
web developer	0.58	0.68	0.62	662
machine learning	0.53	0.39	0.45	713
data analyst	0.35	0.34	0.34	614
data engineer	0.00	0.00	0.00	137
data scientist	0.35	0.32	0.33	629
software engineer	0.43	0.49	0.46	832
deep learning	0.71	0.83	0.77	967
accuracy			0.52	4554
macro avg	0.42	0.44	0.42	4554
weighted avg	0.49	0.52	0.50	4554

Table 5. Neural Networks - Evaluation Matrices

5) **Support Vector Machines:** Support Vector Machines can be considered a suitable method in terms of accuracy, given the nature of feature space, which is a set of abilities. In other words, support vector machines reduce a multi-class classification issue to a set of binary classification problems in which each class is assigned a support vector that is linearly separated from every other class in a higher-dimensional feature space. Overall, support vector machines were able to reach a 67% accuracy. However, because training time is proportional to the number of lessons, it took significantly more time.

6) **Random Forest Classifier:** Given the multiclass classification capabilities of the random forest classifier over the support vector machine, where the multiclass

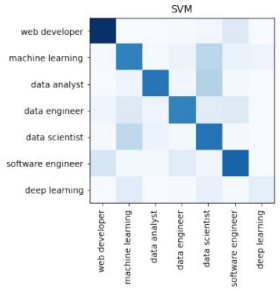

Fig. 6. Support Vector Machines - Confusion Matrix

	precision	recall	f1-score	support
web developer	0.84	0.64	0.73	662
machine learning	0.74	0.60	0.66	713
data analyst	0.47	0.64	0.54	614
data engineer	0.65	0.08	0.14	137
data scientist	0.54	0.60	0.57	629
software engineer	0.68	0.70	0.69	832
deep learning	0.81	0.88	0.84	967
accuracy			0.67	4554
macro avg	0.67	0.59	0.60	4554
weighted avg	0.69	0.67	0.67	4554

Table 6. Support Vector Machines - Evaluation Matrices

classification is broken down into numerous binary classification problems, this can be regarded a good option for the challenge. The classifier, as expected, was able to attain an overall accuracy of 67%.

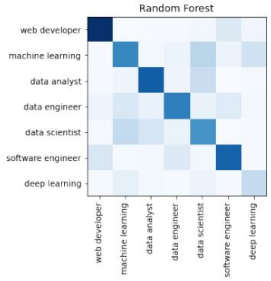

Fig. 7. Random Forest Classifier - Confusion Matrix

	precision	recall	f1-score	support
web developer	0.73	0.72	0.72	662
machine learning	0.68	0.61	0.64	713
data analyst	0.47	0.53	0.50	614
data engineer	0.48	0.21	0.29	137
data scientist	0.55	0.58	0.56	629
software engineer	0.72	0.70	0.71	832
deep learning	0.82	0.87	0.84	967
accuracy			0.67	4554
macro avg	0.63	0.60	0.61	4554
weighted avg	0.67	0.67	0.67	4554

Table 7. Random Forest Classifier - Evaluation Matrices

VI. Summary of Results

We can observe from our results that SVM, RFC, and K Nearest Neighbors perform better than Decision Trees. The

least accurate method is Naive Bayes, which can be improved slightly by using Laplace Smoothing. Multilayer perceptron neural networks have somewhat better accuracy than Naive Bayes with smoothing.

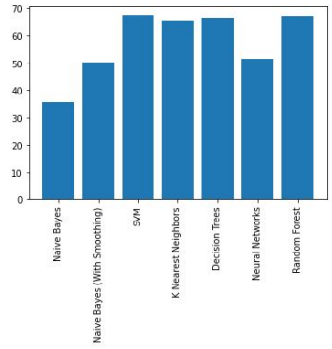

Fig. 8. Summary of the Accuracy

Because it is innately suitable for multi-class classification problems, random forest classification is more suited for this situation. While SVM has a high level of accuracy, it is better suited to binary classification. We're essentially splitting down this multi-class problem into many binary classification problems because it's a multi-class problem.

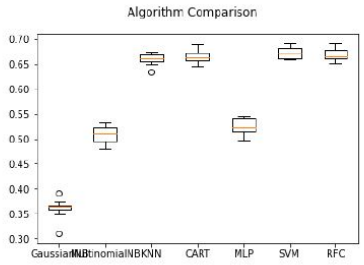

Fig. 9. Comparison of Algorithms

The above image shows the algorithm comparison in terms of accuracy. Although the overall accuracy is around 66 to 67 percent, each role classification has an accuracy of greater than or equal to 88 percent. This could be because several roles share a set of skills, causing overall accuracy to suffer. We were able to determine the prevalence of certain talents for a specific employment position based on our findings. Talents like python, R, SQL, and Hadoop, for example, are more important than other skills for the work of a data scientist. In addition, we trained SVM and RFC on a set of resumes that were considered as a bag of words. As an output, each résumé received a job position.

VII. Conclusion

We were interested in learning more about how job search engines work. We looked at a few sites, including LinkedIn, Indeed, and Glassdoor, to see how they work. We were able to extract some useful information from this problem statement by focusing on the skill set for the time being. What we loved about the project is that by incorporating location or studying the keywords in the resume objective, experience, prior organizations he has worked for, educational qualifications, and so on, we can extract more valuable information. We enjoyed learning about new data analysis and visualization techniques. We enjoyed working on web crawling despite the fact that it was a significant task for us. We learned how to compare and execute different categorization algorithms, as well as a few data analysis and visualization approaches. We also learned a little bit about data crawling.

Model	Accuracy
Naive Bayes	36
Naive Bayes with Laplace Smoothing	50
K Nearest Neighbors	67
Decision Trees	66
MLP	53
SVM	67
RFC	67

Table 8. Classification Accuracy of Algorithms

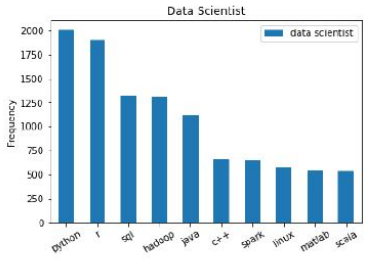

Fig. 10. Distribution of Skills

We first considered predicting the company names, but due to the limitations of web crawling, this proved difficult. As a result, we chose to preserve the job title as the target label. We continue to have reservations about the scope of web crawling. The data presented a significant challenge. We could have produced greater results if we had had a better dataset based on real-time profiles, and if we had had more time to train, we could have added more features.

VIII. Future Work

We'd want to include location, or research the keywords in the resume objective, experience, prior firms he's worked for, educational qualifications, and so on. This could be a nice research topic to investigate how we can look at all of the features at once, or look at them separately with the target label. We want to add more positions and talents before the project's deadline and assess how our model performs. We are looking forward to combining the abovementioned qualities and examining any currently or hybrid models that may be capable of studying a comprehensive résumé in the future scope of this project.

IX. Acknowledgements

We would like to express our gratitude to Professor Yuzhen Ye, as well as AIs Yu Luo and Zhuohuang Zhang, for their assistance throughout the course. We appreciate all of the technical help offered by the Luddy School of Informatics. We'd want to express our gratitude to our peers for establishing a competitive environment and providing us with adequate drive to perform better.

References

[1] Implementation of an Automated Job Recommendation System Based on Candidate Profiles, Vinay Desai, Dheeraj Bahl, Shreekumar Vibhandik,Isra Fatma. https://www.irjet.net/archives/V4/i5/IRJET-V4I5343
[2] Applying Classifications Techniques in Job Recommendation System for Matching of Candidates and Advertisements by Gozde Ozcan, Sule Gunduz Oguducu. https://infonomics-society.org/wp-content/uploads/
[3] Matching People and Jobs: A Bilateral Recommendation Approach, Jochen Malinowski, Tobias Keim. https://citeseerx.ist.psu.edu/viewdoc/download
[4] Recommendation Systems by Aditi and Manaswini CSCI 566 Machine Learning CourseWork
[5] Multi-class classification, https://en.wikipedia.org/wiki/Multiclassclassification
[6] Performance Comparison of Multi-Class Classification Algorithms, Gursev Pirge. https://gursev-pirge.medium.com/performance-comparison
[7] Introduction to Data Visualization in Python, Gilbert Tanner. https://towardsdatascience.com/introduction-to-data-visualization-in-python-89a54c97fbed
[8] Summary of web crawler technology research, Linxuan Yu, Yeli Li, Qingtao Zeng, Yanxiong Sun, Yuning Bian and Wei He. https://iopscience.iop.org/article/10.1088/1742