Candidate Background Verification Using Machine Learning and Fuzzy Matching

Himanshu Suman, Harsh Tamkiya
Computer Science Department, SVIIT
Indore, India

Anurag Singh Kushwah
Techrefic Technologies Pvt Ltd
Indore, India

Abstract— Recruiters usually spend less than a minute looking at each resume when deciding whether it's worth continuing the recruitment process with the candidate. They focus on the keywords, and it is impossible to guarantee a fair process of candidate selection [5]. They have to go through thousands of resumes before deciding which candidate to select within a short time. Thus, making it nearly difficult to perform a thorough background check of the candidates. Recruiters may often encounter cases where candidates provide false information on their resume regarding their education, experience or even skills. The main scope of this paper is to introduce a data-driven approach to verify any candidate's resume by using social media platforms automatically and give recruiters time to only examine promising candidates. Once, the candidate profile is validated, the ranking score is calculated. The score describes how well the candidate's profile matches with the information retrieved from his/her social media accounts. For example, we check a candidate's educational background, work experience and skills from LinkedIn. Later, this paper illustrates a prototype application which can increase the productivity of recruiters. This application shows how to improve the hiring process by giving an unbiased hiring decision support.

Keywords— web-scrapping, machine learning, data analysis, data management

I. INTRODUCTION

With the increasing demand for technical talent, it's no surprise that the recruiting arms race the companies to create new ways to attract the very best individuals using one of the newest weapons in the arsenal in machine learning. Especially, the war for the talents and the number of applications for open positions lead to a new dimension in processing candidate profiles and finding the best match. Recruiters and hiring managers can easily be biased without looking into the candidate's background. Hiring candidates with the right education and experience background are important for the growth of the company. It's usually challenging for hiring managers to go through each and every profiles when they have their inboxes flooded with applicants. Such a situation demands an effective solution to the problem.

In this research prototype, we leveraged current state-of-the-art technology in machine learning to demonstrate how our research can significantly improve the quality and speed of the recruitment process.

II. WHY MACHINE LEARNING?

Machine learning is giving computers the ability to learn without being explicitly programmed [1]. This is primarily accomplished through various recognition processes.

Machine learning usually finds the best correlation and patterns that a human would overlook, leading to higher quality candidate matches [6]. Presently, recruiters have to go through candidate profiles manually searching for the best match for a particular position. But, machine learning is a tool that can add an intelligence layer on top of job filters that can improve the entire recruitment process. We have witnessed a growth in technical population in the previous decade. This working population regularly applies for open positions in various companies. An enormous number of applicants flood the recruiter's inboxes with their resume. Thus, the hiring manager spends a long time deciding and selecting the best candidates for open positions. Time equals money, and machine learning can save recruiters an unimaginable amount of time.

Fig. 1. explains an overview of the project. This diagram simply explains how we have leveraged machine learning to recognize filtered and best fitting resumes among various candidates.

Fig. 1. Represents an overview of the prototype

III. DATA EXTRACTION

Here, the purpose is to acquire the information from the resumes of the applicants and to transform them into a useable format. Data can easily be extracted from the raw formats such as PDF using a variety of tools available in the market. After extracting the data, we have converted this data into JSON format for the specific use of the project [4]. Now, that we have the required data, we can process this data according to our plan.

IV. FROM DATA TO INFORMATION

Here, the purpose is to acquire a comprehensive picture of the candidate's profile by verifying it from social platforms such as LinkedIn, Opportunity, Job case, etc. Thus, we need to connect multiple dimensions of information such as education, experience and skills of the candidate. In this section, we will show how we measure fit on each dimension and how we can merge them to get a final ranking score to check a candidate's profile.

A. Scoring:

In our prototype, each candidate is assigned a score between 0 and 9 indicating it's matching with a profile on a professional networking platform, let's consider LinkedIn. This score is calculated for each data point and also for a complete profile. Education and experience scores are preferred more over any other data points.

B. Searching Candidate Profile:

Modern search engines support Boolean Searching, which encourages the use of various Boolean operators to create conditions so that we can search for the correct profile. A Boolean search, in the context of a search engine, is a type of search where you can use special words or symbols to limit, widen, or define your search [4]. This is possible through Boolean operators such as AND, OR, NOT, and NEAR, as well as the symbols + (add) and - (subtract) . When you include an operator in a Boolean search, you're either introducing flexibility to get a wider range of results, or you're defining limitations to reduce the number of unrelated results. In the context of this paper, we prefer using Boolean operators along with the information present in the candidate's resume. For example, let's assume a candidate named Thomas Shelby attended Cambridge for his undergraduate degree and worked at Microsoft and Google. So, for searching for his profile we will apply-

"Thomas and Shelby"

"Thomas and Shelby and Cambridge"

"Thomas and Shelby and Facebook"

"Thomas and Shelby and Google"

Moreover, we will collect the profile links for each search and combine the links together, say in a list and calculate the occurrence of links. The link with the most occurrence is our target profile.

C. Scraping Data from Professional Networking Platform:

Firstly, web scraping is a technique for extracting information from the internet automatically using software that simulates human web surfing [4]. Web scraping helps to extract large volumes of data about customers, products, people, stock markets, etc. It is usually difficult to get this kind of information on a large scale using traditional data collection methods. We can utilize the data collected from a website such as e-commerce portal, social media channels to understand customer behaviors and sentiments, buying patterns, and brand attribute associations which are critical insights for any business [6].

For matching a resume, we scraped and stored the data from various professional networking platforms. We primarily use selenium and beautiful-soup to scrape and gather information such as a candidate's educational background, experience and skills from these websites [4]. Selenium is a portable framework for testing web applications. Beautiful Soup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping. Scrapping correct data in a proper format is necessary to perform further actions. Furthermore, the scraped data is stored into MongoDB so that we can maintain a record of the profiles. This will help to reduce the processing time as if the same profile is encountered in the future, we can directly look up into the database and continue the process from here.

D. Matching Data:

1) Fuzzy matching: Fuzzy String Matching, also called Approximate String Matching, is the process of finding strings that approximately match a given pattern [9]. The closeness of a match is often measured in terms of edit distance, which is the number of primitive operations necessary to convert the string into an exact match.

2) FuzzyWuzzy Module: FuzzyWuzzy (python module) has, just like the Levenshtein package, a ratio function that computes the standard Levenshtein distance similarity ratio between two sequences or strings [9]. In other words, fuzzy string matching is a type of search that will find matches even when users misspell words or enter only partial words for the search.

3) What is the Levenshtein Distance?

The Levenshtein distance is a metric to measure how apart are two sequences of words. In other words, it measures the minimum number of edits that you need to do to change a one-word sequence into the other [9]. These edits can be insertions, deletions or substitutions. This metric was named after Vladimir Levenshtein, who originally considered it in 1965. The formal definition of the Levenshtein distance between two strings a and b is shown in expression (1):

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j), & \text{if } \min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1 \end{cases} & , otherwise \end{cases} \quad (1)$$

Where 1(ai ≠ bj) denotes 0 when a=b and 1 otherwise. It is important to note that the rows on the minimum above corresponds to a deletion, insertion, and a substitution in that order.

It is also possible to calculate the Levenshtein similarity ratio based on the Levenshtein distance [2]. This can be done using the following expression (2):

$$\frac{(|a| + |b|) - lev_{a,b}(i,j)}{|a| + |b|} \quad (2)$$

Our use case: In our prototype, we have used fuzzy matching to match the resume and scraped data. Assuming the candidate's resume data is available as JSON document and scraped data is stored in the MongoDB database. We match the targeted resume with the searched profile from the MongoDB through fuzzy matching. From the results we obtained, we assign a score to the data point that has been matched. This is then stored for an overall evaluation.

E. Education:

The education score is based on the academic degree, university name, description and start-end-date of the course. These elements have a score between 0 to 9, with 9 being the best. We use each individual score to determine whether the education of a candidate is valid or not. If any of these

elements are unsatisfied in the candidate's resume or on the networking platform, we by default assign a score of 0 to that particular element. We understand that this measure might be unfair, but we have encountered this issue in a few cases. All these elements are then matched using fuzzy matching as explained above. In the case of description, we match the description by using two different approaches. Firstly, we match the entire description using fuzzy-matching and obtain a score. Secondly, we match the description without the stop words by using NLKT module in python [1]. Practically, there is a pretty low probability of getting the same description on both the resume and website. So, we have taken this step to strengthen our chances of matching the resume.

F. Work Experience:

The experience score is based on the duration and place of employment i.e. company name, job title, job description and start-end-date. These elements have a score between 0 to 9, with 9 being the best. We use each individual score to determine whether the experience of a candidate is valid or not. If any of these elements are unsatisfied in the candidate's resume or on the networking platform, we by default assign a score of 0 to that particular element. We understand that this measure might be unfair, but we have encountered this issue in a few cases. All these elements are then matched using fuzzy matching as explained above. Additionally, we eliminate words that are frequently common in company names to enhance company matching. In the case of the job description, we match the description by using two different approaches. Firstly, we match the entire description using fuzzy-matching and obtain a score. Secondly, we match the description without the stop words by using NLKT module in python [1]. Practically, there is a pretty low probability of getting the same description on both the resume and website. So, we have taken this step to strengthen our chances of matching the resume.

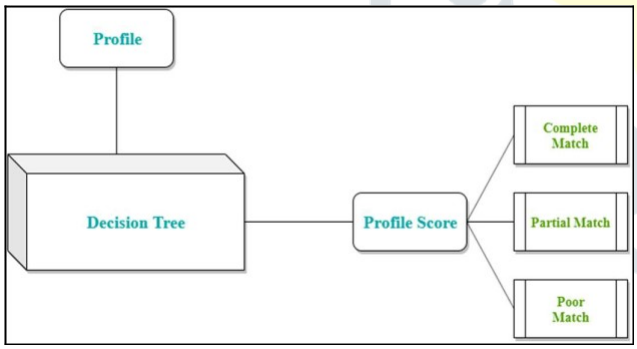

Fig. 2. Represents a walk-through of machine learning model

V. MACHINE LEARNING MODEL

Here, we have used Decision Tree Classifier as our preferred machine learning model. A decision tree is an acyclic graph that can be used to make decisions. In each branching node of the graph, a specific feature of j of the feature vector is examined [1][3]. If the value of the feature is below a specific threshold, then the left branch is followed; otherwise, the right branch is followed. As the leaf node is reached, the decision is made about the class to which the example belongs. Decision Trees are versatile Machine learning algorithms that can perform both classification and regression tasks. They are a very powerful algorithm capable of fitting complex datasets [7].

A. Decision Tree Terminologies:

1) Root Node: It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2) Splitting: It is a process of dividing a node into two or more sub-nodes.
3) Decision Node: When a sub-node splits into further sub-nodes, then it is called a decision node.
4) Leaf / Terminal Node: Nodes do not split is called Leaf or Terminal node.
5) Pruning: When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
6) Branch / Sub-Tree: A subsection of the entire tree is called branch or sub-tree.
7) Parent and Child Node: A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of the parent node.

B. Types of Decision Tree:

1) Categorical Variable Decision Tree: Decision Tree which has a categorical target variable then it called a categorical variable decision tree.
2) Continuous Variable Decision Tree: Decision Tree has continuous target variable then it is called as Continuous Variable Decision Tree.

C. Why we chose Decision Tree?

1) Easy to Understand: Decision tree output is very easy to understand even for people from the non-analytical background. It does not require any statistical knowledge to read and interpret them. Its graphical representation is very intuitive and users can easily relate their hypothesis [7].
2) Less data cleaning required: It requires less data cleaning compared to some other modelling techniques. It is not influenced by outliers and missing values to a fair degree.
3) Non-Parametric Method: Decision tree is considered to be a non-parametric method. This means that decision trees have no assumptions about space distribution and the classifier structure [7].
4) Non-linear relationships between parameters do not affect tree performance.

D. Problems we may face with the Decision Tree:

1) Overfitting: Decision-tree learners can create over-complex trees that do not generalize the data well. Overfitting is one of the most practical difficulties for decision tree models.
2) Not fit for continuous variables: While working with continuous numerical variables, decision tree loses information, when it categorizes variables in different categories.

3) **Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the data set prior to fitting with the decision tree.**

E. Information Gain:

Less impure node requires less information to describe it. And, the more impure node requires more information. Information theory is a measure to define this degree of disorganization in a system known as Entropy. If the sample is completely homogeneous, then the entropy is zero and if the sample is equally divided (50% — 50%), it has an entropy of one [2]. Entropy controls how a Decision Tree decides to split the data. It actually affects how a Decision Tree draws its boundaries.

Entropy can be calculated using formula [2]:

$$Entropy = -p\log_2 p - q\log_2 q \quad (3)$$

Here p and q are the probability of success and failure respectively in that node. Entropy is also used with the categorical target variable. It chooses the split which has the lowest entropy compared to the parent node and other splits. The lesser the entropy, the better it is.

F. Steps to Calculate Entropy for a Split:

1) **Calculate the entropy of the parent node**
2) **Calculate entropy of each individual node of split and calculate the weighted average of all sub-nodes available in the split.**

We can derive information gain from entropy as 1- Entropy.

VI. OBSERVATIONS

After training the model, we have to evaluate the performance of the model using the convenient metrics. The confusion matrix is a table that summarizes how successful the classification model is at predicting examples belonging to various classes. It consists of the following terms - true positives (TP), true negatives (TN), false positives (FP) and false negatives (FP). The confusion matrix is used to calculate two other performance metrics: precision and recall. These two are most frequently used to assess the model [3].

Precision is the ratio of correct positive predictions to the overall number of positive predictions [3]. Recall is the ratio of correct positive predictions to the overall number of positive examples in the dataset (e.g. Fig. 3) [3]. To understand the meaning and importance of precision and recall for the model assessment it is often useful to think about the prediction problem as the problem of research of documents in the database using a query. The precision is the proportion of relevant documents in the list of all returned documents. The recall is the ratio of the relevant documents returned by the search engine to the total number of the relevant documents that could have been returned.

Accuracy is given by the number of correctly classified examples divided by the total number of classified examples.

[3] Accuracy is a useful metric when errors in predicting all classes are equally important (e.g. Fig. 3).

$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\text{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$

Fig. 3. Represents performance metrics for the model

In this research paper, we have compared various machine learning models such as Decision Tree, Random Forest, XG Boost, Logistic Regression and Support Vector Machine (SVM) to decide which machine learning model fits best for our purpose. Table I shows detailed comparison of machine learning models. And, fig. 4 depicts the result graph for the Decision Tree Algorithm.

TABLE I: Detailed Comparison of Machine Learning Models

ALGORITHM	ACCURACY	PRECISION	RECALL	TIME (IN SECONDS)
Decision Tree	94.72	92.80	93.70	0.0051
Random Forest	95.27	94.40	98.45	0.0550
Logistic Regression	96.90	97.40	96.80	0.0045
XG Boost	97.58	90.15	95.70	0.1431
SVM	96.59	94.40	91.70	0.0051

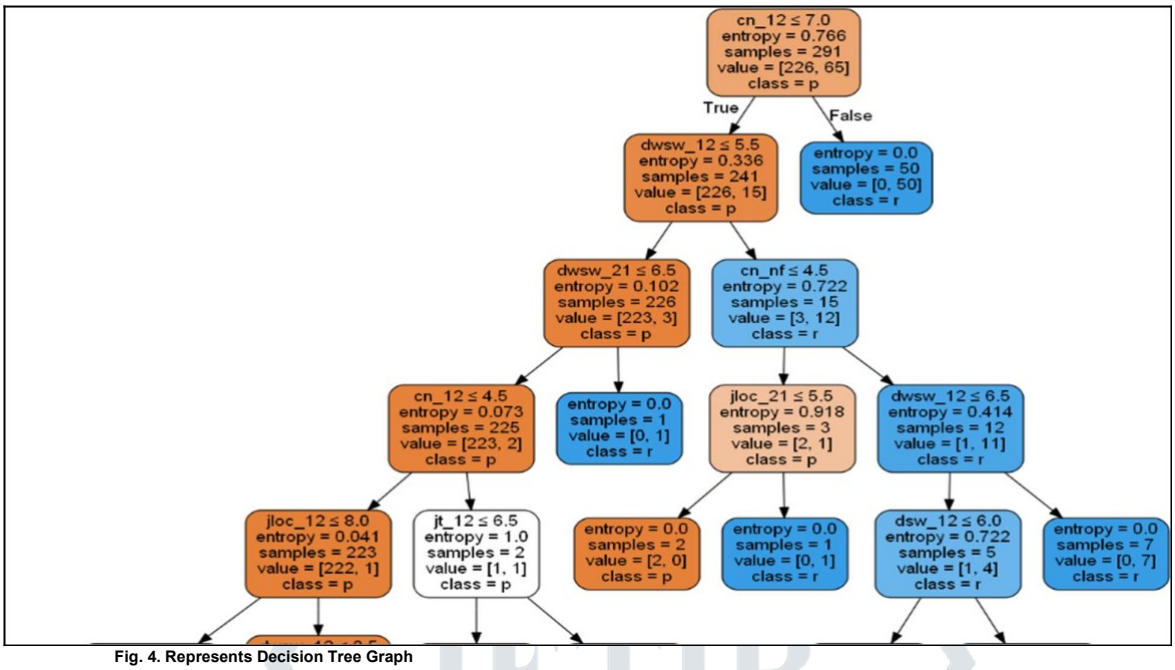

Fig. 4. Represents Decision Tree Graph

VII. CONCLUSION

As per our research paper, we have favored the decision tree algorithm because it requires less data cleaning and it is a non-parametric method. The data that will be used in the machine learning algorithm is not biased and the accuracy of the model is profoundly prioritized. We also need a machine learning algorithm that takes a minimal amount of time to predict the results. Therefore, according to the above observations, we can easily identify that decision tree successfully satisfies our needs. XG-Boost and Random Forest have shown different results on different candidate data and their computational performance is incompetent compared to other algorithms. As data is often not linearly separable, Linear Regression and Support Vector Machine (SVM) show reduced performance with large data points. [7] Decision Tree algorithm has shown significant results with respect to this research paper. Therefore, our approach can perform a back-ground check of candidates using social media platforms.

VIII. REFERENCES

[1] Geron, Aurelien. 2017. Hands-On Machine Learning with Scikit-Learn & TensorFlow. O'Reilly Media Inc, California. pp. 566.

[2] Hastie, Trevor., Tibshirani, Robert. and Friedman, Jerome. 2001. The Elements of Statistical Learning. Springer-Verlag New York Inc. New York. pp. 745.

[3] Burkov, Andriy. 2019. The Hundred-Page Machine Learning Book. Andriy Burkov. Canada. pp. 160.

[4] McKinney, Wes. 2012. Python for Data Analysis. O'Reilly Media Inc, California. pp. 522.

[5] Zimmerman, Tim., Kotschenreuther, Leo., Schmidt, Karsten. 2016. Data driven HR - Resume Analysis Based on Natural Language Processing and Machine Learning. CU. 2 pp.

[6] Faliagka, Evanthia., Ramantas, Kostas., Tsakalidis, Athanasios., Tzimas, Giannis. 2012. Application of Machine Learning Algorithms to an Online Recruitment System. ICIW. 6 pp.

[7] Caruana, Rich. 2006. An Empirical Comparison of Supervised Learning Algorithms. CU. 8 pp.

[8] Elsesser, Kim. Forbes. Is artificial intelligence the key to recruiting a diverse workforce. [https://www.forbes.com/sites/kimelsesser/2019/08/13/is-artificial-intelligence-the-key-to-recruiting-a-diverse workforce/#394d348b5f8b]. Assessed 20 November 2019.

[9] Javier Carrera Arias, Francisco. Datacamp. Fuzzy string matching python. [https://www.datacamp.com/community/tutorials/fuzzy-string-python]. Assessed 25 November 2019.

[10] Dietterich, Thomas., Kong, Eun., 1995. Machine Learning Bias, Statistical Bias, and Statistical Variance of Decision Tree Algorithms. OSU. 14 pp.

[11] Quinlan, J. R. (1987). Simplifying decision trees. International Journal of Man-Machine Studies, 27, pp.221–234.

[12] M. Saks, A. Wigderson, Probabilistic Boolean decision trees and the complexity of evaluating game trees, Proc. 27th FOCS, 1986, pp. 29–38.

[13] Sathyadevan S., Nair R.R. (2015) Comparative Analysis of Decision Tree Algorithms: ID3, C4.5 and Random Forest. Springer, New Delhi. Vol 31.

[14] Peng, X., Guo, H., Pang, J.: Performance analysis between different decision trees for uncertain data. In: Proceedings of the 2012 International CSSS, 12, pp.574–577.