2023 International Conference on Advances in Electronics, Communication, Computing and Intelligent Information Systems (ICAECIS) | 979-8-3503-4805-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICAECIS58353.2023.10170518

Screening and Ranking resume's using Stacked Model

D.Kavitha
Assistant Professor, Dept of Computer Science &Engineering Easwari Engineering College,TN, India dkavithavijay@gmail.com

Padmavathi B
Assistant Professor, Dept of Computer Science& Engineering , Easwari Engineering College,TN, India bpadma_cse@yahoo.com

Sushanth V
Dept of Computer science & Engineering Easwari Engineering College, TN, India sushanthsush185@gmail.com

Yashwanth P
Dept. of Computer Science & Engineering Easwari Engineering College, TN, India imyash505@gmail.com

Yuvaraj R Dept of Computer science & Engineering Easwari Engineering College, TN, India singhyuvaraj12@gmail.com

Abstract: Finding the top candidates for a position is the aim of the resume screening process. The application must make use of machine learning methodologies as well as natural language processing to rate candidates in real time. Natural language processing (NLP) and machine learning methods are used to rate resumes. The output would be the resume of a top candidate, with resumes and job descriptions serving as the input. The output's results are obtained instantly. String matching, Cosine Similarity, and TF-IDF will all be done with Mon. Existing systems are straightforward and efficient, but they lack precision, efficiency, and processing capacity.

Keywords: Machine Learning, Artificial Intelligence, Deep Learning, NLP, TD-IDF

1. Iintroduction

The global recruitment industry is valued at $200 billion. The process involves sifting through a wide pool of applicants to find those most qualified for a certain job description. When a firm advertises a job opportunity, they will receive dozens of applications for the vacancy. Every hiring manager starts with a thorough review of candidates' resumes. If there is a position vacant at a firm, HR is likely to get thousands of resumes and cover letters per day.

At now, this approach takes the longest and is most prone to mistakes because of the need for human interaction. Humans, too, have limits that prevent them from functioning nonstop. Consequently, decreased efficiency is also an issue. As a result, we have suggested a system which could quickly identify the necessary skill set by analyzing a document or even a resume and sorting the results based on the skill sets, which is a limitation imposed by the company. Machine learning is a strategy we plan to employ. An up-and-coming field in HR technology, technology for recruitment aims to automate or greatly streamline formerly manual processes like resume screening.

Countless institutions conduct brand-new studies across a wide range of disciplines every day. From the world of information technology to that of medicine, daily life benefits from a plethora of studies and innovations. There are now more jobs available in these industries than ever before because to these developments. We may safely assume that recruitment practices elsewhere follow the same patterns. The candidate, therefore, is the one who must initially create the resume, as this document serves as the cornerstone of the individual's application.

Most modern businesses request that applicants submit their resumes online. Once they have received the email, the following step is to categorize the responses based on the need. Now typically they sort this manually, but it takes a long time. On the other hand, that might be the cause of a lot of mistakes. This kind of sorting is quite inefficient. This might lead to hiring someone whose qualifications aren't sufficient for the position, or it could cause the manual sorting process to overlook an exceptional candidate.

The most time-consuming part of a recruiter's work is narrowing down a vast pool of applicants to a select few who are best suited for the open position. When a firm receives hundreds of resumes, it can become overwhelming for human recruiters to sort through them to find the best possible prospects. For any particular position, the organization may receive hundreds of applicants. Advertisement, around 75 percent do not demonstrate the appropriate expertise needed for the position described in the advertising.

A single position can have thousands of applicants because of LinkedIn, Facebook, and online talent databases. In today's HR departments, automated systems are becoming the norm. Resume screening aids in streamlining and improving the field. As a result, human resources professionals have more time to spend on meaningful interactions with both applicants and employees. Our proposed model is a digital solution for HR departments that uses NLP technology to efficiently screen applicants' resumes. Token-based similarity and latent Dirichlet allocation are the two best techniques for comparing the content of resumes in the field to that of the employer's job description. The recruiter is then given the top-ranked comparable resumes.

2. LITERATURE SURVEY

In (1), The top candidates are graded in accordance with the job description using K-NN, Content-based Advice: use the cosine similarity measure to find resumes that are a good fit for the position you're advertising.

In (2), the suggested system, JARO, is a virtual assistant who conducts interviews by evaluating candidates' resumés (CVs). This will then generate a list of questions to ask the requestor. This expedites the interview process to ensure an impartial decision is made.

In (3), a neural-network model and a few text processing methods are employed to know the practicality of each resume.

We provide many model variants, each making use of the pair/triplet-based loss or a semi-supervised learning approach, to deal with the label deficiency problem in the dataset.

Multiple regression was used within (4) to test the theory. The results show a significant beneficial relationship between two criteria, illustrating how increased use of AI in work environments boosts HR functional effectiveness.

The study by the authors of a hybrid strategy for curriculum vitae screening was described in (5)Using a real-world recruiting dataset, they demonstrate their ability to classify resumes using an integrated knowledge base and show that their findings are promising when compared to those obtained using standard machine learning-based resume classification approaches..

In (6), Machine Learning methods such as KNN, Linear SVC, and XGBoost are used in this system. All of these algorithms are combined into a two-level stacked model that aids in properly predicting certain job profiles from a text description. This framework might be useful for employers to waitlist rivals, as well as for applicants to verify if their resume is well-structured enough for the system to recognize appropriate job profiles.

The interactive web application in (7) allows job seekers to post their resumes and apply for any open openings. The corporate recruiter uses NLP and machine learning to match applicant resumes to job criteria.

In (8), it web tool assists us in screening and ranking resumes. Using a technology called Natural Language Processing, the submitted resumes are evaluated to the job description to find the best profiled resumes (NLP). The resumes are then scored and ranked from best match to worst match. Only corporate recruiters looking to choose the top prospects from a vast pool of applications are given access to this ranking.

In (9), The authors hope to automate this screening process by analyzing NLP in depth to change help streamline all aspects of the hiring process for businesses, and we can provide a numerical score to each application by comparing the number of required fields to those included on each resume. Their categorization model is also rather precise, with an accuracy of 74%.

Finally, in (10), In order to provide material that recruiters may use, the writers gather data from resumes and conduct the appropriate analysis on it. As a result, the Resume Parser would assist recruiters in quickly identifying the most qualified prospects, saving them time and effort.

System Flow Diagram

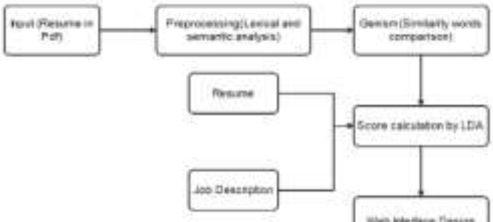


Fig.1. System flow diagram of resume screening system

The system flow diagram(Fig.1.) describes the flow of the proposed model, at first the input as the research in pdf is given then the preprocessing of Lexical and semantic analysis is processed. Genism which compares the similarity words and the score calculation by LDA is carried out which gives the Resume and Job Description. This overall process helps to result the web interface design.

Architecture diagram

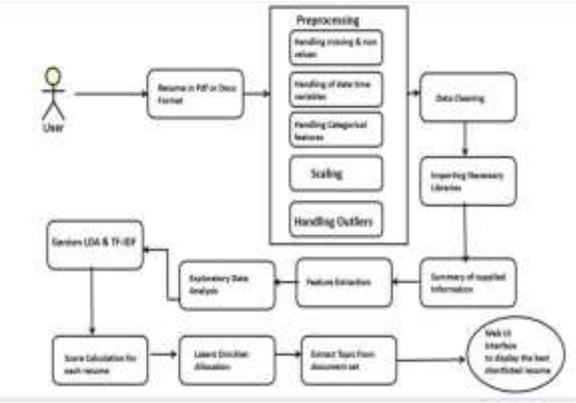

User
Resume in Pdf or Docx Format
Preprocessing
Handling missing & non values
Handling of date time variables
Handling Categorical features
Scaling
Handling Outliers
Data Cleaning
Importing Necessary Libraries
Gensim LDA & TF-IDF
Explanatory Data Analysis
Feature Extraction
Summary of supplied Information
Score Calculation for each resume
Latent Dirichlet Allocation
Extract Topic From document set
Web UI interface to display the best shortlisted resume

Fig.2. Architecture diagram of resume screening system

The architecture diagram(Fig.2.) shows the interaction between the user and system. User inputs the resume in pdf Or docx Format and data preprocessing is carried out. Data cleaning is perform and necessary libraries are imported. The information is summarized and the feature is extracted. Score calculation is done which results in the web UI interface to display the shortlisted resume.

3. EXISTING SYSTEM

The manual review of applications is the existing method for resume screening. The employment market in India is massive, and with millions of people looking for work, it is physically difficult to scan the resumes and discover someone who is a good fit. Because of this, the entire process of employing people is sluggish and inefficient, which costs the companies resources. In order to locate a candidate whose qualifications best match those outlined in the job posting, HR must manually search through all of the resumes. This requires a lot of resources and is prone to error, therefore there is a possibility that the ideal candidate for the position will be overlooked in the process.

4. PROPOSED SYSTEM

Our approach uses machine learning and natural language processing (NLP) to rank resumes. With the help of artificial intelligence, this tool for contextually assessing resumes goes beyond keywords. After reviewing resumes, the software evaluates prospects in real-time in accordance with the hiring needs of the recruiter. The online application seeks to organize the resumes by comparing the resumes that fit the specified Job Descriptions and intelligently reading job descriptions as input. After filtering, it offers a ranking and suggests the best resume for a specified text job description. The software uses natural language processing to instantly match and rate applicants. When it comes to string matching, cosine similarity, overlap coefficient, and natural language processing, this software uses Mong rather than general methods.

We employ a different methodology for our work because it largely concentrates on the content of the resumes, from which we extract skills and other pertinent information to match people with job descriptions.

5. PROPOSED ALGORITHM

Module 1: Data Collection and Pre-Processing

a.Data Collection

Data collection refers to the process of gathering information from various sources, which is then used to develop deep learning. The information should be saved in a way that is appropriate for the challenge. We collect data from Kaggle that contains the resume dataset in the csv format. We train and test our model using this dataset.

b.Data Pre-Processing

Data preprocessing is a data mining technique used to turn the raw data into a format that is both practical and effective. Data preprocessing changes the data into a format that can be processed in data mining, machine learning, and other data science tasks more quickly and efficiently.

To ensure reliable findings, the techniques are typically applied at the very beginning of the machine learning and AI development pipeline. Although there are many various tools and strategies for preparing data, we have chosen to concentrate on the following measures for our dataset:

c.Data Cleaning

Data cleaning is the process of removing erroneous, damaged, badly structured, duplicate, or incomplete data from a dataset. When combining different data sources, we have checked this in a variety of methods to see if any data is duplicated or improperly categorized.

This module cleans the data by getting rid of duplication, hateful or irrelevant content, fixing structural issues, and dealing with missing data. We eliminate observations with missing values as a last option, but keep in mind that doing so will remove or damage data.

We make up the missing figures based on other observations, but this approach again runs the risk of tainting the data's objectivity because you can be working with conjectures rather than concrete evidence. Here are the following steps we have used to clean our data:

- **Removing HTML Tags:** Emails, websites, and job connections are commonly included in the data we retrieve from various resumes. It's possible, then, that we'll come across several HTML tags—header, body, paragraph, strong, and so on—during the data extraction process. Through the use of regular expressions, we are able to quickly and easily strip the text of all HTML tags. The plan is to write a regular expression that will get rid of all tags. For this reason, we employ the following syntax:

- without html = re.sub(pattern=r", repl=' ', string=text), print(f"without html").

- **Removing Hashtags:** To prevent hashtags from obscuring our procedure, we must remove them. We are aware that hashtags always contain letters, digits, and the # symbol but never any whitespace. The text will then be split at whitespace matches to produce the tokens. We use the re.sub function for the same. The Python Regular Expressions (re) package contains the re.sub() method. All instances of the supplied pattern that match are replaced by the replace string in the returned string.

a.Removing Punctuations: When we are dealing with the texts contained in the dataset, we frequently run into issues that need us to remove specific characters from the text. Regex is used for the portion that involves replacing characters with punctuation. Using a certain regex, we remove all of the punctuation and replace it with a string that is empty. First, the string is initialized, and then the starting content is printed with any necessary punctuation added. After that, we construct a function that removes all of the punctuation by using the re.sub command. At this point, the output is printed down without any punctuation.

b. Removing Extra Whitespace

When working with text datasets, we may run across scenarios in which we have more than one gap between words in texts, which is, for the most part, undesirable. These situations might arise at any time. In web development, issues like these can arise, and when they do, they typically require repair.

This task can be accomplished by combining the split() function with the strip() function and the join() function. This is accomplished in two distinct stages. In the first step, we convert the string into a list of words, and then we use the join function to link the words together with a single space between them.

c. Removing Stop Words

In both the indexing process and the process of receiving results, search engines are instructed to exclude instances of regularly used words known as "stop words."

These words should not be stored in our database or processed at this time. By maintaining a database of "stop words," we can quickly and simply get rid of them. The NLTK (Natural Language Toolkit) module for Python includes a dictionary of "stop words" for 16 languages. You may locate them in the nltk data folder. Thus, we use the same process to eliminate stop words from our dataset. First, we check the list of stop words in English using the following command:
bring in nltk

'English' is a stop word, and we can get it by using the nltk.

corpus.import.stopwords.print (stop words.words('English'))

Now, in a new code, we import stop words from ntlk. corpus and import word tokenize from nltk. Tokenize. We then set the stop words in English in the file using the command:
Assert stop words = set(stop words. words('English')).
open file1 ("text.txt")

Then, the file content is read as a stream and append the file so that there are no stop words in the resume information we have gathered from the dataset.

d. Tokenization

Tokenization is the process of dividing text into a list of tokens from a string of text. Tokens can be viewed as components, similar to how a word functions as a token in a phrase and how a sentence functions as a token in a paragraph.

The NLTK tokenize module makes it simple to tokenize the text's sentences and words. We're going to import the pertinent NLTK library functions first. However, import nltk first. Then we import word tokenize from nltk.tokenize. We print the outcome last.

e. Plotting The Graph

We use Matplotlib to plot categories vs counting graph. In Python, Matplotlib is the go-to library for plotting data. We can develop graphs and charts, and generate static raster or vector files with little to no effort, all without resorting to clunky graphical user interfaces.

We run "pip install matplotlib" to get matplotlib set up. Finally, in our notebook, we bring in matplotlib. pyplot as plt. One simple command in Matplotlib allows us to generate a plot with little effort.

To finish off our plot, we'll use the.xlabel(),.ylabel(), and.title() methods to give our axes meaningful names and give our diagram a proper heading (). Matplotlib's plt.show() is added to the end of your Python script to show off your plot.

Matplotlib's fig size and dpi options make it possible to generate plots with precisely the dimensions and dots per inch (DPI) that we require. The dpi is the number of dots per inch, while the fig size is the product of the figure's width and height (in inches) (pixel-per-inch).

Without passing in a fig size or dpi, Matplotlib will use the defaults. Matplotlib allows us to create professional-grade graphics in a variety of file formats (png, jpg, svg, pdf, etc.). The figure will be saved using the. save fig() method.

f. Creating Word Cloud

In data visualization, a "Word Cloud" is a method of displaying textual information in which the relative size of individual words represents their relative frequency or significance in the data. An effective way to draw attention to key information in a document is to utilize a word cloud. Social network data is often analyzed using word clouds. Python modules matplotlib, pandas, and word cloud are needed to generate a word cloud. We begin by installing the necessary packages using the lines pip install matplotlib, pip install pandas, and pip install wordcloud.
Then, we import matplotlib. pyplot as plt, pandas as pd. Furthermore, from wordcloud, we import word Cloud and stop words. Then, we read our pre-processed csv file using df= pd.read_csv(). The following the first action is to set up a text variable to store the aggregated reviews. This may be done with Python's join function. We've arrived at the crucial stage of creating the wordcloud! We select the key features for our wordcloud
In our model, we use the command to create wordcloud and specify the height and width of the Word Cloud, maximum font size for the most significant words, and eliminate the stop words. Stop words are the words like that, was, is, the, etc., a which do not hold any emphasis in our dataset. Moreover, we have also set the background color from the choice of colors. In default settings, it is set to black. Apart from this, we set the color theme using "colormap" code. The next step is to use matplotlib's imshow() function to visually represent the created wordcloud. As a result, we have a complete list of keywords from the resume collection.

Module 2: Model Creation And Training

a. Creating The LDA Model

We will create a LDA model to train and TF-IDF to extract feature from and the job description as well to score against each other and find the best resumes for the job.

The significance of a word is measured using a metric known as TF-IDF (Term Frequency - Inverse Document Frequency). It provides a TF-IDF value that reflects the importance of the word in the context of the dataset by calculating its frequency use.

Calculating the frequency of a word in a given phrase is as simple as dividing the number of times that word appears in the phrase by the total number of words in the phrase. The term frequency formula is as follows. For the inverse document frequency, we divide the log of the total number of sentences by the number of sentences that contain the phrase in question. After multiplying the TF formula by the IDF formula (1) , we obtain a vector whose value we were seeking. On the one hand, this worth will be represented by a graph of relative significance, and on the other, by a list of words.

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

Based on the graph of TF-IDF vectorization, the most significant term is placed first, in descending order of its relevance, yielding a growth curve with the slope backwards. The greatest term frequency value will be assigned to the most essential groups of terms relevant to occupations and skill sets; they may be retrieved and utilized to train LDA.

A generative statistical model called latent Dirichlet allocation (LDA) is used in natural language processing to explain groups of observations by anomalous clusters that can't be seen but are responsible for apparent correlations across datasets. If, for the sake of argument, observations are words that are assembled into papers, then it is reasonable to assume that the documents contain a variety of subject matter, and that the occurrence of each word is associated with a particular topic. In this sense, LDA is illustrative of the topic model type.

Using Dirichlet distributions and processes, Latent Dirichlet Allocation (LDA) may classify or categories both the text inside a document and the words per topic. The programmed then assigns a score to the resume depending on the requirements of the advertised position. We'll be assigning points to each applicant's resume and prioritizing those with the highest totals.

b. Training Our Model

The model is first divided into a training set and a testing set before it is trained. Algorithms and applications that rely on predictions can have their efficacy estimated by using the train-test split for machine learning. Our own machine learning model's output may be compared to that of an automated system using this straightforward way. A typical split is 70-30, with the training set consisting of more generic examples and the test set including more specific data.

Separating a dataset into train and test sets allows us to evaluate the efficacy of our machine learning model. The model is adjusted with the aid of the train set, whose components are recognized. The second batch, dubbed "test data," is utilized only for the sake of making projections. To divide our dataset into train and test, we use the following approach. The pandas and sklearn packages are imported. Sklearn is the best machine learning library for Python. The model selection module of the scikit-learn library includes the splitter function train_test_split (). Further, the read csv() method is then used to import the CSV file. The data frame is now stored in the variable df.

Then, we employ a test size of 0.3, which allocates 30% of the dataset to evaluations and retains the remaining 70% for training purposes. We also set random state=0 to ensure a fair distribution of information across the two groups.

In this step, we teach our dataset. After dividing the data into training and test sets, we next employ a machine learning algorithm to assess how well the training data has been put to use. Scikit was used to do data fitting for this study. One of the best Python libraries for machine learning is scikit-learn. Clustering, classification, regression, and dimensionality reduction are just some of the machine learning and statistical modelling capabilities available in the sklearn toolkit.

One of these tools is the scikit learn 'fit' method. After the model has been initialized, the 'fit' method trains the algorithm on the training data. To be honest, that's about all it does. The sklearn fit procedure thus use the training data to educate the ML model. After that, we may proceed with the machine learning process using additional scikit learn techniques, such as predict and score.

This is how we use fit method on the training dataset. First, we define the instance of the model, then use the fit method as mentioned in the figure below. Finally, Then, within the parenthesis, we provide the training dataset's features and label vector. These datasets are also known as X_train and Y_train. X_train, the X-input to the fit() method, is a 2-dimensional format for the perfect fitting of the model. Otherwise, our model shows an error.

Finally, we evaluate the accuracy of our LDA model. The accuracy, which goes from 0 to 1, is the proportion of data that are correctly classified. We compare the predicted class to the actual class in this measure very instinctively since we want the model to appropriately classify the data. To do so, we utilize the accuracy score function in the Sklearn library. For the same, we employ the following command: print(accuracy score(y test, y pred)).

Module 3: Creating A Web App

Now that we have a completely trained model and have saved the weights, we can develop a web application using stream lit to give the model a user interface (UI). Web applications for machine learning and data science are developed using stream light, an open-source Python framework.

Stream light allows us to build and release web apps rapidly. The stream lit web software streamlines the iterative process of coding and viewing its results on the screen. stream lit is the platform of choice for developing Python-based apps.

First, we install stream lit using the pip install stream lit command on python. Following this, we generate a python script containing streamlit commands, then run such script using the command streamlit run <ourscript.py>. Streamlit provides a large number of widgets that we made use of, such as st.selectbox, st.checkbox, st.slider, and others.

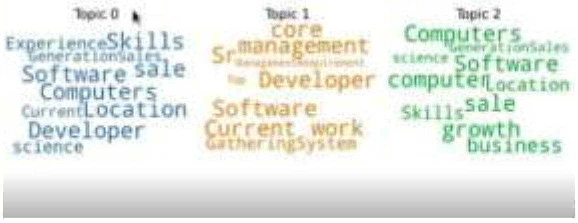
Using the st.sidebar function, we can easily arrange our widgets or data. This strategy assisted us in aligning data in the sidebar on the left panel. To display a select box in the left panel, we simply used st.sidebar.selectbox.

Then, in the web app, we use stream lit to create a title and a Data Frame. Now, in our console, we type the following command: stream lit run stream lit app.py. After the command is executed, we get an URL. Clicking on the URL takes us to our webpage or web app.

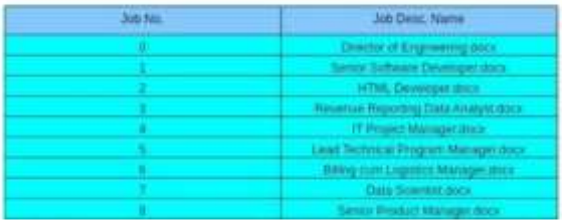
In our web application, user can add the multiple resumes and the job title we are considering for and get the output in return as the score for resume so that we can pick top score resumes for the next process of job application.

6. OUTPUT SCREENSHOTS

Output 1: The Word Cloud shows the Top Keywords that are seen in the resume screening model. The words Computers, Software, Skills, management, current work are used most.



Output 2: It shows the descriptions of various job profiles that are added in our model.



Job No.	Job Desc. Name
0	Director of Engineering.docx
1	Senior Software Developer.docx
2	HTML Developer.docx
3	Revenue Reporting Data Analyst.docx
4	IT Project Manager.docx
5	Lead Technical Program Manager.docx
6	Billing cum Logistics Manager.docx
7	Data Scientist.docx
8	Senior Product Manager.docx

7. ALGORITHM COMPARISON

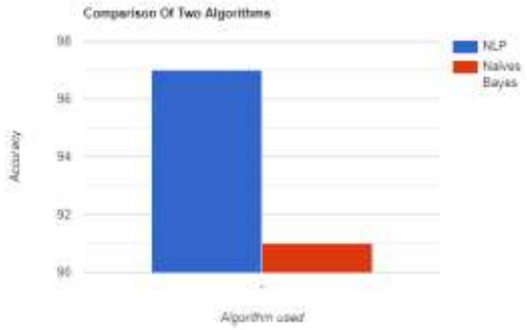


Fig.3. Comparison graph of two algorithms used

Here, we calculate the accuracy by rates and compare the two models' performance characteristics using the comparison graph(Fig.3.). It turns out that NLP performs better than Naive Bayes.

CONCLUSION

The model's outcomes are positive. The resume classifier algorithm does a good job of automating the human task of allocating tasks to new hires in accordance with their interests, work history, and declared competencies. The Resume Screening System eliminates the need for manual screening which is ineffective and ensures that no candidate is overlooked. The system's capacity to accept or reject a candidate for a job will depend on two factors: the company's requirements, and the results of an efficiency test designed to verify that the candidate's resume is accurate and that he or she possesses the requisite knowledge and abilities.

8. FUTURE SCOPE

With this prediction, the resume can be shortlisted without any manual check. It helps to increase the accuracy rate and in future this model itself trains the system to analyze and filter the required skills from the resume which is injected as the input. Thus enhancement in prediction is high and effective.

REFERENCES

1. Pradeep Kumar Roy; Sarabjeet Singh Chowdhary; Rocky Bhatia, "A Machine Learning technique for automation of Resume Recommendation system," Procedia Computer Science Volume 167, Pages 2318-2327, 2020

2. Jitendra Purohit; Aditya Bagwe; Rishabh Mehta; Ojaswini Mangaonkar; Elizabeth George, "Natural Language Processing based Jaro-The Interviewing Chatbot," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), August 2019

3. Yong Luo, Huaizheng Zhang, Yongjie Wang, Yonggang Wen, and Xinwen Zhang, "ResumeNet: A Learning-Based Framework for Automatic Resume Quality Assessment," 2018 IEEE International Conference on Data Mining (ICDM), December 2018 [ResumeNet: A Learning-Based Framework for Automatic Resume Quality Assessment]

4. 2020 International Conference on Computation, Automation, and Knowledge Management (ICCAKM): Garima Bhardwaj, S. Vikram Singh, and Vinay Kumar, "An Empirical Study of Artificial Intelligence and Its Impact on Human Resource Functions."

5. Book chapter by Abeer Zaroor, Mohammed Maree, and Muath Sabha, 2017; published in Smart Innovation, Systems and Technologies volume 72

6. Rasika Ransing Akshaya Mohan; Nikita Bhrugumaharshi Emberi; Kailas Mahavarkar "Screening and Raking resumes based on Stacked Model, 2021, IEEE

7. Amin Sujit Web Application For Screening Resumes, Nikita Jayakar, Sonia Sunny, Pheba Babu, M. Kiruthika, and Ambarish Gurjar, 2019 IEEE.

8. M. Alamelu D.Sathish Kumar; R. Sanjana; J.Subha Sree; A.Sangeerani Devi; D. Kavitha "Resume Validation and Filtration using NLP, 2021, IEEE IEMECON

9. M.F. Mridha; Rabeya Basri; Muhammad Mostafa Monowar; Md. Abdul Hamid "A Machine Learning Approach for Screening Individual's Job Profile Using Convolutional Neural Network", 2021 IEEE

10. Artificially Intelligent and Mechanics Voices: Anushka Sharma, Smiti Singhal, and Dhara Ajudia, 2021 AIMV