A PROJECT REPORT ON

RESUME ANALYSIS AND CANDIDATE RANKING SYSTEM

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA

For Partial Fulfilment of Award of the Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE
AND MACHINE LEARNING)

Submitted By

K. CHANDRASEKHAR (20KN1A4223)

Under the esteemed guidance of

Dr SEVA SREEDHAR BABU
Professor, Department of CSE (AI&ML)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

NRI INSTITUTE OF TECHNOLOGY
Autonomous
(Approved by AICTE, Permanently Affiliated to JNTUK, Kakinada) Accredited by NBA (CSE, ECE & EEE), Accredited by NAAC with 'A' GradeISO 9001: 2015 Certified Institution
Pothavarappadu (V), (Via) Nunna, Agiripalli (M), Krishna Dist., PIN: 521212, A.P, India.

2023-2024

A PROJECT REPORT ON

RESUME ANALYSIS AND CANDIDATE RANKING SYSTEM

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA

For Partial Fulfilment of Award of the Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE
AND MACHINE LEARNING)

Submitted By

K. CHANDRASEKHAR (20KN1A4223)

Under the esteemed guidance of

Dr. SEVA SREEDHAR BABU
Professor, Department of CSE (AI&ML)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

NRI INSTITUTE OF TECHNOLOGY

Autonomous

(Approved by AICTE, Permanently Affiliated to JNTUK, Kakinada) Accredited by NBA (CSE, ECE & EEE), Accredited by NAAC with 'A' GradeISO 9001: 2015 Certified Institution

Pothavarappadu (V), (Via) Nunna, Agiripalli (M), Krishna Dist., PIN: 521212, A.P, India.

2023-2024

NRI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Approved by AICTE, Permanently Affiliated to JNTUK, Kakinada)

Accredited by NBA (CSE, ECE & EEE), Accredited by NAAC with 'A' GradeISO 9001: 2015 Certified Institution Pothavarappadu (V), (Via) Nunna, Agiripalli (M), Krishna Dist., PIN: 521212, A.P, India.

CERTIFICATE

This is to certify that this project report entitled "**RESUME ANALYSIS AND CANDITATE RANKING SYSTEM** " is the bonafide work of **Mr. K. CHANDRASEKHAR (20KN1A4223)** in partial fulfilment of the requirements for the award of the graduate degree of Bachelor of Technology in Computer Science and Engineering (Artificial Intelligence and Machine Learning) during the academic year **2023-2024**.

(Dr S. Sreedhar Babu)
Signature of the Guide

(Dr B. H. Dasaradhram)
Signature of the HOD

Signature of the External Examiner

DECLARATION

I, **K. CHANDRASEKHAR** hereby declare that the project report entitled "**RESUME ANALYSIS AND CANDITATE RANKING SYSTEM"** is an original work done in the Department of Computer Science & Engineering (AI&ML), NRI Institute of Technology, Pothavarappadu(V), Agiripalli (M), Eluru District during the academic year **2023-2024**, in partial fulfilment for the award of the Degree of **Bachelor of Technology** in **Computer Science & Engineering (AI&ML)**. I assure that this project is not submitted to any other College or University.

Roll No	Name of the Student	Signature
(20KN1A4223)	**K. CHANDRASEKHAR**	

ACKNOWLEDGEMENT

Firstly, I would like to convey our heartful thanks to the Almighty for the blessings on us to carry out this project work without any disruption.

I am extremely thankful to **Dr S. Sreedhar Babu**, my guide throughout project. I am also thanking him for most independence and freedom of throughout given to me during various phases of the project.

I am also thankful for my project coordinator **P. Bhagya Madhuri** for her valuable guidance which helped me to bring this project successfully.

I am very much grateful to **Dr B. H. Dasaradha Ram**, H.O.D of C.S.E (AI&ML) Department, for his valuable guidance which helped me to bring out this project successfully. His wise approach made me learn the minute details of the subject. His matured and patient guidance paved away for completing my project with the sense of satisfaction and pleasure.

I am greatly thankful to our principal **Dr C. Naga Bhaskar** for his kind support and facilities provided at my campus which helped me to bring out this project successfully.

Finally, I would like to convey my heartful thanks to my Technical Staff, for their guidance and support in every step of this project. I convey my sincere thanks to all the faculty, and friends who directly or indirectly helped me for the successful completion of this project.

PROJECT ASSOCIATES

K. CHANDRASEKHAR (20KN1A4223)

ABSTRACT

In the contemporary job market, the task of screening resumes and identifying suitable candidates for specific job roles can be time-consuming and labor-intensive. To address this challenge, we propose an Automated Resume Screening and Candidate Ranking System. This system leverages natural language processing (NLP) and machine learning techniques to analyze job descriptions and candidate resumes, extract relevant keywords and features, and rank candidates based on their suitability for the given job role.

The primary objective of our project is to streamline the recruitment process by automating the initial screening of resumes and identifying top candidates efficiently. By automating these tasks, we aim to save time for recruiters and hiring managers, reduce human bias, and improve the overall efficiency of the recruitment process.

The system is built using the Flask framework in Python, allowing for easy deployment and integration with web-based interfaces. Upon receiving a zip file containing resumes and a job description PDF file, the system extracts text data from these files and preprocesses it using techniques such as tokenization and TF-IDF vectorization. The job description and resumes are then analyzed to extract relevant keywords and features.

The Automated Resume Screening and Candidate Ranking System offers a powerful solution to the challenges associated with manual resume screening. By automating the initial screening process and providing recruiters with a ranked list of candidates, the system enables more efficient and informed decision-making in the recruitment process.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Introduction to Project

The project, titled "Resume Analysis & Candidate Ranking System," aims to streamline the process of analyzing job descriptions and resumes to rank candidates based on their suitability for a given job role. In today's competitive job market, recruiters often face challenges in efficiently assessing numerous resumes against specific job requirements. This project offers a solution by automating the analysis process, allowing recruiters to focus their time and efforts on the most promising candidates.

1.2 Problem Definition

Recruiters and hiring managers are inundated with a large volume of resumes for each job opening, making it difficult and time-consuming to identify the most qualified candidates. Manual resume screening is not only labor-intensive but also prone to bias and inconsistency. Additionally, it can be challenging to objectively compare candidate qualifications against the job description, leading to suboptimal hiring decisions.

1.3 Solution for Problem Definition

The "Resume Analysis & Candidate Ranking System" addresses these challenges by leveraging natural language processing (NLP) techniques to extract and analyze relevant information from resumes and job descriptions. By applying algorithms for text preprocessing, keyword extraction, and similarity scoring, the system automates the evaluation process and provides recruiters with ranked lists of candidates based on their suitability for the role. This enables recruiters to make data-driven hiring decisions and significantly reduces the time and effort required for candidate screening.

1.4 Process Diagram

The following diagram illustrates the high-level process flow of the "Resume Analysis & Candidate Ranking System":

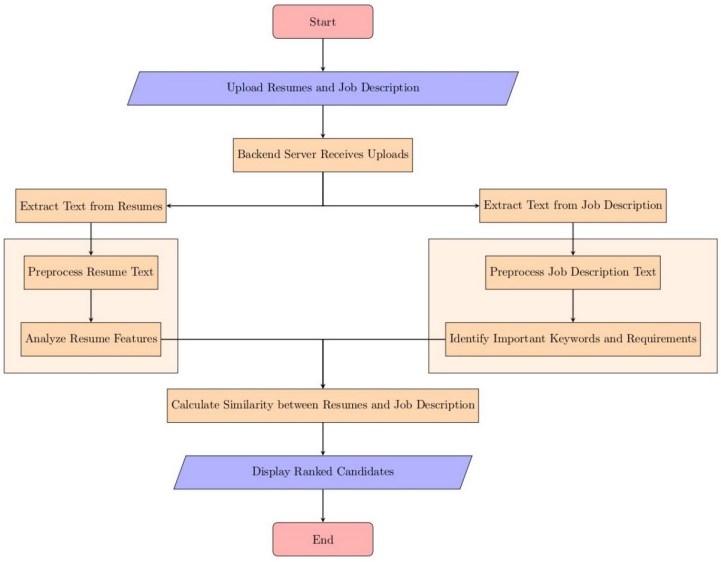


Figure-1.1: Flowchart of the system

The process begins with the upload of resumes and the job description. The backend server receives the uploads and proceeds to extract text from both resumes and the job description. After preprocessing the text data, the system analyzes the resumes and the job description to identify important keywords and requirements. Subsequently, the system calculates the similarity between resumes and the job description to determine candidate rankings. Finally, the ranked candidates are displayed for further review by recruiters.

1.5 Objectives

The primary objective of the Resume Analysis & Candidate Ranking System is to streamline the recruitment process by automating the analysis of resumes and ranking candidates based on their relevance to a given job description. The key objectives of the project include:

- Develop a system capable of extracting relevant information from resumes, including skills, experiences, and qualifications.
- Implement algorithms for evaluating the suitability of candidates based on the similarity between their resumes and a provided job description.
- Create a user-friendly interface that allows recruiters to upload resumes and job descriptions, view ranked candidates, and make informed hiring decisions.

2. LITERATURE REVIEW

The ability of automated resume screening and candidate ranking systems to expedite hiring procedures and enhance the effectiveness of decision-making has made them extremely popular in recent years. In order to extract relevant information from resumes, Rajeswari et al. (2020) support the integration of natural language processing (NLP) techniques; Wang, Zhang, and Liu (2019) suggest a deep learning strategy to improve candidate ratings. Chen et al. (2018) present a topic modeling-based approach and show how well it works to find latent themes in resumes, while Liu et al. (2021) investigate the use of machine learning algorithms for resume ranking.

Simultaneously, endeavors have been directed towards improving resume screening by the application of NLP approaches, as demonstrated by Smith and Jones (2019), who emphasize the significance of NLP in enhancing candidate assessments. Li et al. (2020) offer a thorough analysis of NLP-based screening techniques, outlining developments and difficulties in the field. Simultaneously, methods that rely on machine learning, like the automated hiring system Liu and Zhang (2018) suggested, highlight how machine learning can simplify hiring procedures while reducing the need for human participation.

Furthermore, to improve candidate ranking accuracy, it is critical to integrate text mining and semantic analysis approaches, according to current studies. While Patel and Patel (2021) provide insights into the changing environment of automated resume screening systems, Wang, Li, and Zhang (2018) describe an automated method utilizing semantic analysis. Together, these developments highlight the growing interest in applicant rating and automated resume screening, which provide enhanced recruiting effectiveness and decision- making precision by combining NLP, machine learning, and semantic analysis techniques.

S.no	Title of the paper	Name of Publisher	Published Year	Merits	Demerits
1	Automated Resume Screening Using NLP Techniques	International Journal of Engineering Research & Technology	2020	Utilizes NLP techniques for automated screening	Specific NLP knowledge required for implementation
2	A Deep Learning Approach for Automated Resume Screening	IEEE Access	2019	Deep learning provides high accuracy	Requires large datasets and computational resources
3	Resume Screening Based on Machine Learning Algorithms	Journal of Physics: Conference Series	2021	Diverse machi ne learning algorithms for screening	Performance highly depende nt on data quality

4	A Topic Modeling Based Approach for Resume Ranking	Proceedings of the 2nd International Conference on CSAE	2018	Efficiently ranks resumes using topic modeling	May not captur e nuanced information in resumes
5	Automated Resume Screening Using Topic Modeling	International Journal of Innovative Technology and Exploring Engineering	2020	Topic modeling enables efficient screening	May overlook specific skillsets or experiences
6	Improving Resume Screening Using Natural Language Processing	Journal of Computational Science	2019	NLP enhances screening accuracy	May struggle with unstructured or poorly formatted resumes
7	A Review of Resume Screening Methods Based on NLP Techniques	International Journal of Computer Applications	2020	Provides insights into various NLP-based methods	Lack of standardization across methods

3. SYSTEM ANALYSIS

3.1 Existing System

The AI-powered resume analysis and candidate ranking system is designed to streamline the hiring process. It parses resumes using natural language processing, employs machine learning for candidate ranking, and integrates with Applicant Tracking Systems (ATS). The system ensures a user-friendly interface, incorporates feedback for continuous improvement, and prioritizes security, scalability, and reporting. It offers a comprehensive solution for efficient and unbiased talent acquisition.

3.2 Proposed System

The proposed system aims to automate the resume screening process using Natural Language Processing (NLP) techniques and machine learning algorithms. By leveraging NLP, the system can extract relevant information from resumes and job descriptions, such as skills, experience, and qualifications. Machine learning models trained on historical data can then analyze and rank resumes based on their suitability for the job position.

The proposed system offers several advantages over the existing manual process, including:

1. **Efficiency:** Automation reduces the time and effort required for resume screening, allowing HR professionals to focus on more strategic tasks.
2. **Consistency:** The system applies standardized criteria for evaluating resumes, minimizing inconsistencies and biases in candidate selection.
3. **Scalability:** The automated system can handle large volumes of resumes, ensuring timely processing of applications even during peak recruitment periods.
4. **Accuracy:** NLP and machine learning algorithms enhance the accuracy of candidate evaluation by identifying relevant keywords and patterns in resumes.

3.3 Analysis Model

The analysis model for the proposed system involves several steps:

1. **Data Collection:** Resumes and job descriptions are collected from various sources, such as job portals and company databases.
2. **Preprocessing:** Text data undergoes preprocessing to remove noise, standardize formatting, and extract relevant features.
3. **Feature Extraction:** NLP techniques are used to extract features from resumes and job descriptions, such as keywords, skills, and qualifications.
4. **Model Training:** Machine learning models, such as classification or ranking algorithms, are trained on labeled data to predict the suitability of resumes for specific job positions.
5. **Evaluation:** The performance of the trained models is evaluated using metrics such as accuracy, precision, recall, and F1-score.
6. **Deployment:** The finalized model is deployed as part of the automated resume screening system, where it processes incoming resumes and provides ranked candidate lists to HR personnel.

4. FEASIBILITY ANALYSIS

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation.

- Technical Feasibility
- Operation Feasibility
- Economic Feasibility

4.1 Technical Feasiility

To determine whether the proposed system is technically feasible, we should take into consideration the technical issues involved behind the situation. Technical feasibility center on the existing computer system and to what extent it can support the proposed addition. Python and its libraries are technology software which are used to develop Data Analytics. So, there is no need for additional purchase of any software, and these are open-source software's which are freely available in Internet.

4.2 Operational Feasibility

Proposed projects are beneficial only if they can be turned out into information systems that will meet the user's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the application implementation. This system is operational feasible since the users are familiar with the technologies and hence there is no need to gear up the personnel to use the system. Also, the system is very friendly and easy to use.

4.3 Economic Feasibility

To decide whether a project is economically feasible, we must take into consideration various factors as:

- Cost benefit analysis
- Long-term returns
- Maintenance cost

The proposed system is computer based. It requires average computing capabilities which is very basic requirement and can be afforded by an organization; it does not incur additional economic overheads, which renders the system economically feasible.

5. SYSTEM REQUIREMENT SEPECIFICATION

5.1 Introduction

The System Requirement Specification (SRS) outlines the functional and non-functional requirements of the automated resume screening system. It defines the features and capabilities of the system, as well as the hardware and software requirements necessary for its implementation.

5.2 Functional Requirements

The functional requirements describe the specific functionalities and capabilities of the automated resume screening system. These requirements detail the actions the system must perform to meet the needs of users. Functional requirements include:

- **Resumes Upload:** The system should allow users to upload resumes and job descriptions in supported formats (e.g., PDF).
- **Text Extraction:** The system should extract text from uploaded resumes and job descriptions accurately.
- **Preprocessing:** Resumes and job descriptions should undergo preprocessing to remove noise, standardize formatting, and extract relevant features.
- **Keyword Extraction:** NLP techniques should be applied to extract keywords, skills, qualifications, and other relevant information from resumes and job descriptions.
- **Ranking and Screening:** Machine learning models should analyze resumes and rank candidates based on their suitability for specific job positions.
- **User Interface:** The system should provide a user-friendly interface for HR personnel to view and interact with ranked candidate lists.

5.3 Non-Functional Requirements

Non-functional requirements specify the quality attributes and constraints of the system. These requirements define how the system should behave in terms of performance, usability, reliability, security, and other factors. Non-functional requirements include:

- **Performance:** The system should process resumes and job descriptions efficiently, with minimal latency.
- **Scalability:** The system should be able to handle large volumes of resumes and job descriptions, scaling to accommodate increasing workload demands.
- **Accuracy:** NLP and machine learning models should achieve high accuracy in extracting information and ranking candidates.
- **Security:** The system should ensure the confidentiality and integrity of sensitive data, such as personal information contained in resumes.
- **Usability:** The user interface should be intuitive and easy to navigate, requiring minimal training for HR personnel.

- **Reliability:** The system should be robust and reliable, minimizing downtime and errors during operation.

5.4 Hardware Requirements

- **Server:** A dedicated server with sufficient processing power and memory to handle text extraction, preprocessing, and machine learning tasks.
- **Storage:** Adequate storage capacity to store resumes, job descriptions, and trained machine learning models.
- **Networking:** Reliable internet connectivity to support user interactions and data exchange.
- **Processor** : Intel Core i5
- **Hard Disk** : 1 TB
- **Monitor** : Laptop or Desktop
- **GPU** : NVIDIA Quadro P1000
- **Ram** : 8GB

5.5 Software Requirements

- **Operating System:** The system should run on a compatible operating system, such as Linux or Windows Server.
- **Programming Languages:** Python for backend development, JavaScript for frontend development.
- **Frameworks and Libraries:** Flask for web application development, scikit-learn for machine learning tasks, NLTK and Gensim for NLP processing.
- **Database:** Optional database system for storing user data and system logs (e.g., MySQL, PostgreSQL).
- **Operating system:** Windows 10/11
- **Coding Language:** Python
- **Tool** : Anaconda Navigator- Jupyter Notebook and Visual Studios

6. SYSTEM DESIGN

6.1 System Architecture

In a resume analysis and candidate ranking system project, the system architecture typically involves several key components to efficiently process and evaluate resumes. Here is a summary of a common system architecture for such a project:

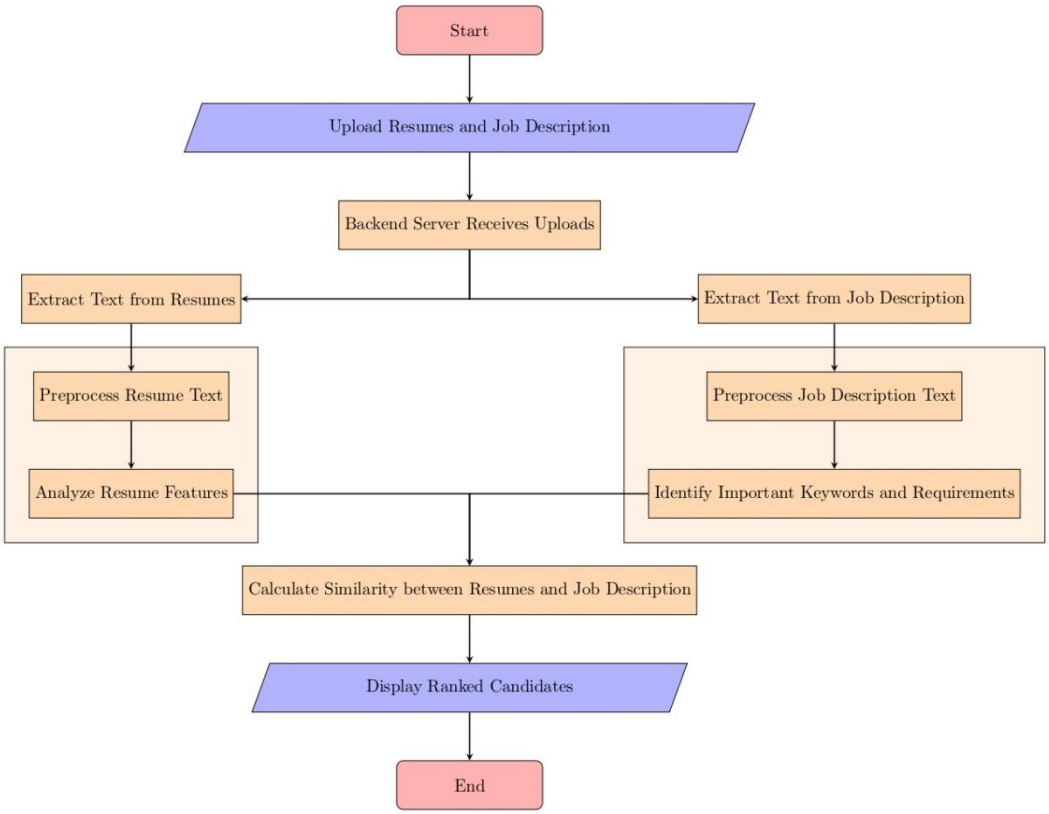


Figure-4.1: Flowchart of the system

1. **Start:** This represents the beginning of the process. It indicates the starting point of the workflow.
2. **Upload Resumes and Job Description:** This process component signifies the step where users upload resumes and the job description to the system.
3. **Backend Server Receives Uploads:** This represents the beginning of the process. It indicates the starting point of the workflow. This process component represents the backend server's role in receiving the uploaded resumes and job description from the user interface.
4. **Extract Text from Resumes:** This process component denotes the extraction of text data from the uploaded resumes.
5. **Preprocess Resume Text:** This process component involves preprocessing the extracted resume text, which may include tasks like removing stop words, stemming, or lemmatization.
6. **Analyze Resume Features:** This process component signifies the analysis of resume features, such as identifying key skills, experiences, or qualifications.

7. **Extract Text from Job Description:** This process component involves extracting text data from the uploaded job description.
8. **Preprocess Job Description Text:** Similar to preprocessing resume text, this process component involves preprocessing the job description text.
9. **Identify Important Keywords and Requirements:** This process component denotes the identification of important keywords and requirements from the job description.
10. **Calculate Similarity between Resumes and Job Description:** This process component represents the backend server's role in receiving the uploaded resumes and job description from the user interface.
11. **Display Ranked Candidates:** If there are enough candidates, this process component represents the display of ranked candidates to the user interface.
12. **End:** This signifies the end of the process flow.

6.2 UML Diagrams

Unified Modeling Language (UML) is a standardized modeling language used in software engineering to visually represent systems. UML diagrams provide a graphical representation of the system's structure, behavior, and interactions. In the context of the automated resume screening system, various UML diagrams can be utilized to illustrate different aspects of the system's design and functionality.

6.2.1 Class Diagram

The Class Diagram depicts the static structure of the system by representing classes, their attributes, methods, and relationships. In the context of the automated resume screening system, the Class Diagram can illustrate the different classes representing entities such as Resume, JobDescription, User, and various components of the system architecture.

6.2.2 Use Case Diagram

The Use Case Diagram identifies the system's functionalities from the perspective of its users. It illustrates the interactions between users (actors) and the system, depicting various use cases and their relationships. In the context of the automated resume screening system, the Use Case Diagram can illustrate use cases such as Upload Resume, View Candidate List, and Analyze Resumes.

6.2.3 Activity Diagram

The Activity Diagram describes the flow of activities within the system, illustrating the sequence of actions or steps involved in completing a process. In the context of the automated resume screening system, the Activity Diagram can depict the workflow of processing resumes, from text extraction and preprocessing to ranking candidates based on their suitability for specific job positions.

6.2.4 Sequence Diagram

The Sequence Diagram illustrates the interactions between objects or components within the system over time. It shows the sequence of messages exchanged between objects and the order in which they occur. In the context of the automated resume screening system, the Sequence Diagram can depict the interactions between the user interface, backend server, and external components during the resume screening process.

6.2.5 Component Diagram

The Component Diagram illustrates the physical and logical components of the system and their dependencies. It shows how different components interact to achieve system functionality. In the context of the automated resume screening system, the Component Diagram can illustrate the various software components, such as the frontend user interface, backend server, NLP modules, and machine learning models.

6.3 Importance of UML Modelling

Unified Modeling Language (UML) modeling plays a crucial role in the development of the automated resume screening system. Here are some key reasons highlighting the importance of UML modeling in this project:

1. **Visualization of System Architecture:** UML diagrams provide a visual representation of the system's architecture, components, and interactions. This visualization helps stakeholders, including developers and clients, to understand the system's structure and organization more effectively.
2. **Clarification of Requirements:** UML diagrams, such as Use Case and Activity diagrams, help clarify and specify the functional requirements of the system. By visualizing user interactions and system processes, UML modeling ensures a clear understanding of the system's intended behavior and functionalities.
3. **Identification of Use Cases:** Use Case diagrams in UML help identify and define the various interactions between users and the system. In the context of the automated resume screening system, Use Case diagrams illustrate the different scenarios in which users interact with the system, such as uploading resumes, viewing candidate lists, and analyzing resumes.
4. **Design Validation:** UML modeling allows developers to validate and refine the system's design before implementation. By creating Class diagrams, Sequence diagrams, and other UML diagrams, developers can identify potential design flaws, inconsistencies, or ambiguities early in the development process, reducing the risk of costly errors later on.
5. **Communication and Collaboration:** UML diagrams serve as a common language for communication and collaboration among project stakeholders. They provide a standardized way to convey complex system designs and concepts to team members, clients, and other stakeholders, facilitating effective communication and alignment of project goals.
6. **Documentation:** UML diagrams serve as valuable documentation artifacts for the automated resume screening system. They provide a comprehensive overview of the system's architecture, functionalities, and interactions, which can be referenced by developers, testers, and other stakeholders throughout the project lifecycle.
7. **Facilitation of System Maintenance:** UML diagrams aid in the maintenance and evolution of the automated resume screening system. They document the system's structure, dependencies, and behavior, making it easier for developers to understand and modify existing components or add new features as requirements change over time.

6.4 Class Diagram

In software engineering, a class diagram in the UML is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and their relation between the classes. The class diagram is the main building block in object-oriented modelling. It is used both for general conceptual modelling of the semantics of the application and for detail modelling translating the model into programming code. Each class has 3 fields:

- Classes
- Attributes and
- Behaviors

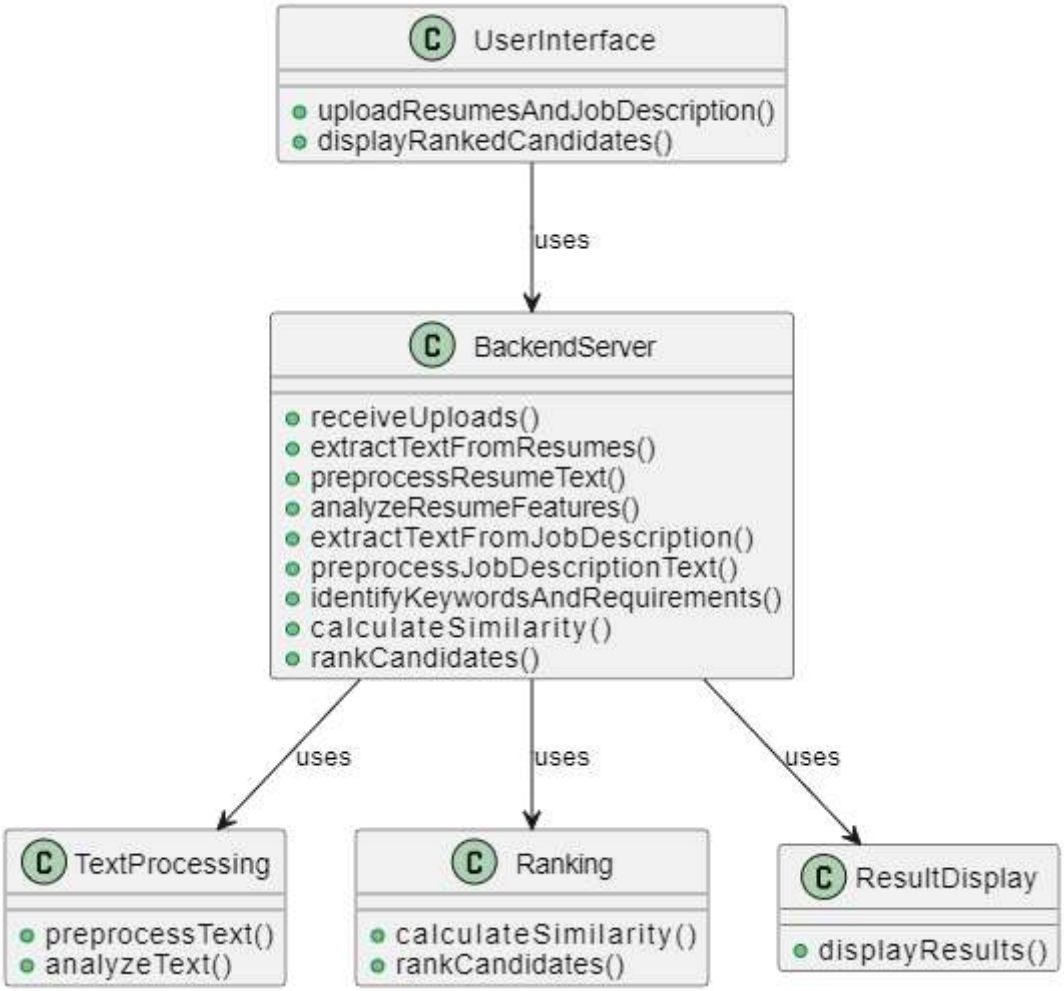


Figure-4.2: Class diagram

In the UML diagram provided, the system comprises several classes, including "UserInterface," "BackendServer," "TextProcessing," "Ranking," and "ResultDisplay." The "UserInterface" class allows the user to interact with the system by uploading resumes and job descriptions and viewing ranked candidates. The "BackendServer" class manages the processing of uploaded data, including text extraction, preprocessing, analysis, similarity calculation, ranking, and result display. The "TextProcessing" class handles text preprocessing and analysis, while the "Ranking" class calculates similarity scores and ranks candidates. Finally, the "ResultDisplay" class presents ranked candidates to the user interface for display. Together, these classes form the backbone of the system, facilitating efficient resume analysis and candidate ranking.

6.5 Use Case Diagram

A use case diagram in the UML is a type of behavioral diagram defined by and created from a Use Case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed by which actor. The three main components of this UML diagram are:

- Functional requirements
- Actors
- Relationships

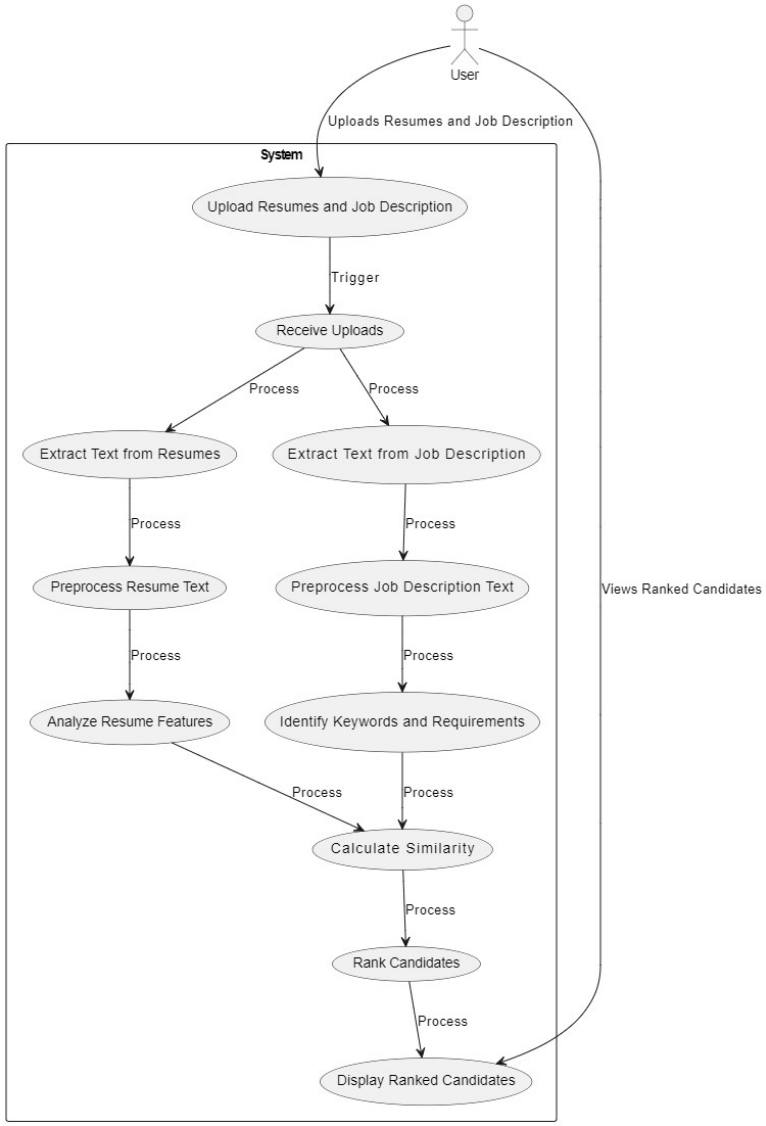


Figure-4.3: Use Case diagram

In the provided use case diagram, the system involves a single actor, the "User," who interacts with the system to upload resumes and job descriptions. Upon uploading, the system initiates a series of processes, including receiving the uploads, extracting text from resumes and job descriptions, preprocessing the text data, analyzing resume features, extracting text from the job description, preprocessing the job description text, identifying keywords and requirements, calculating similarity between resumes and job descriptions, ranking candidates based on similarity scores, and finally displaying the ranked candidates to the user.

6.6 Activity Diagram

A use case diagram in the UML is a type of behavioral diagram defined by and created from a Use Case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed by which actor. The three main components of this UML diagram are:

- Functional requirements
- Actors
- Relationships

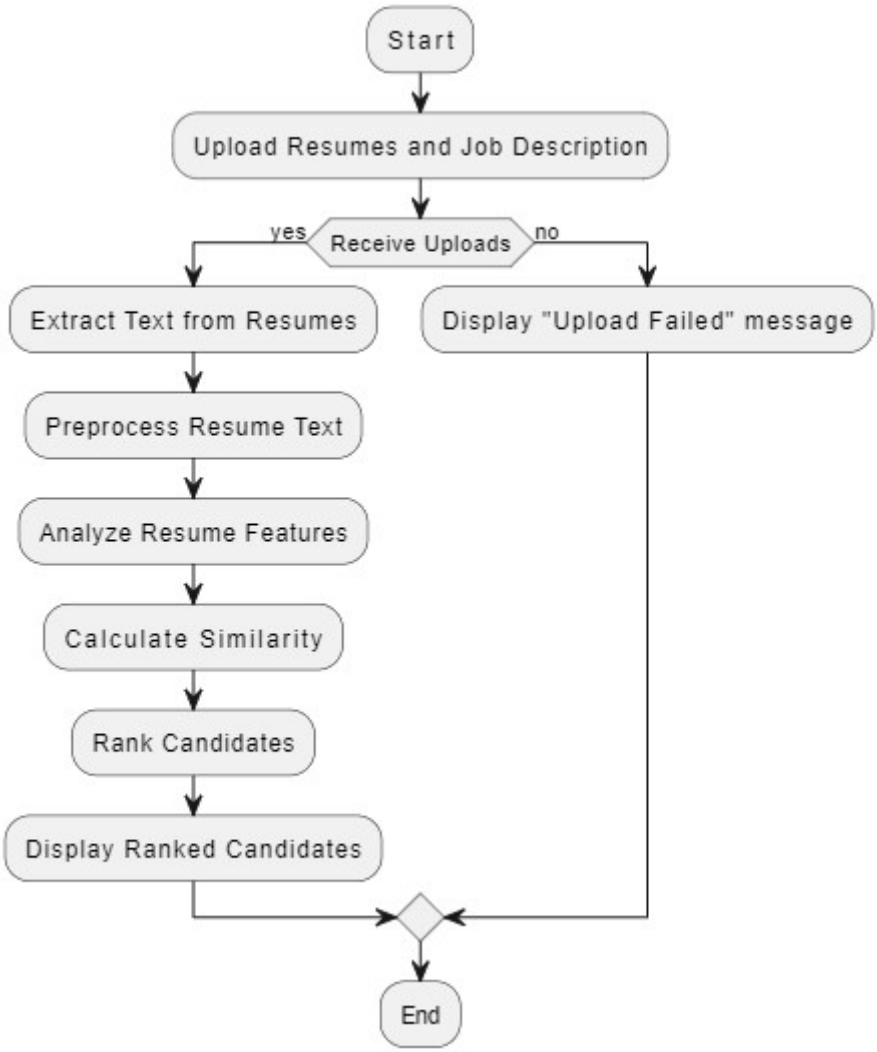


Figure-4.4: Activity diagram

In the provided activity diagram, the system involves a single actor, the "User," who interacts with the system to upload resumes and job descriptions. Upon uploading, the system initiates a series of processes, including receiving the uploads, extracting text from resumes and job descriptions, preprocessing the text data, analyzing resume features, extracting text from the job description, preprocessing the job description text, identifying keywords and requirements, calculating similarity between resumes and job descriptions, ranking candidates based on similarity scores, and finally displaying the ranked candidates to the user.

6.7 Sequence Diagram

The sequence diagram shows a detailed flow for a specific use case or even just part of a specific use case. There are almost self-explanatory. They show the calls between the different objects in their sequence and can show at a detailed level, different calls to different objects. A Sequence diagram has two dimensions: The vertical dimension shows the sequence of messages/calls in the time order that they occur. The horizontal dimension shows the object instances to which the messages are sent.

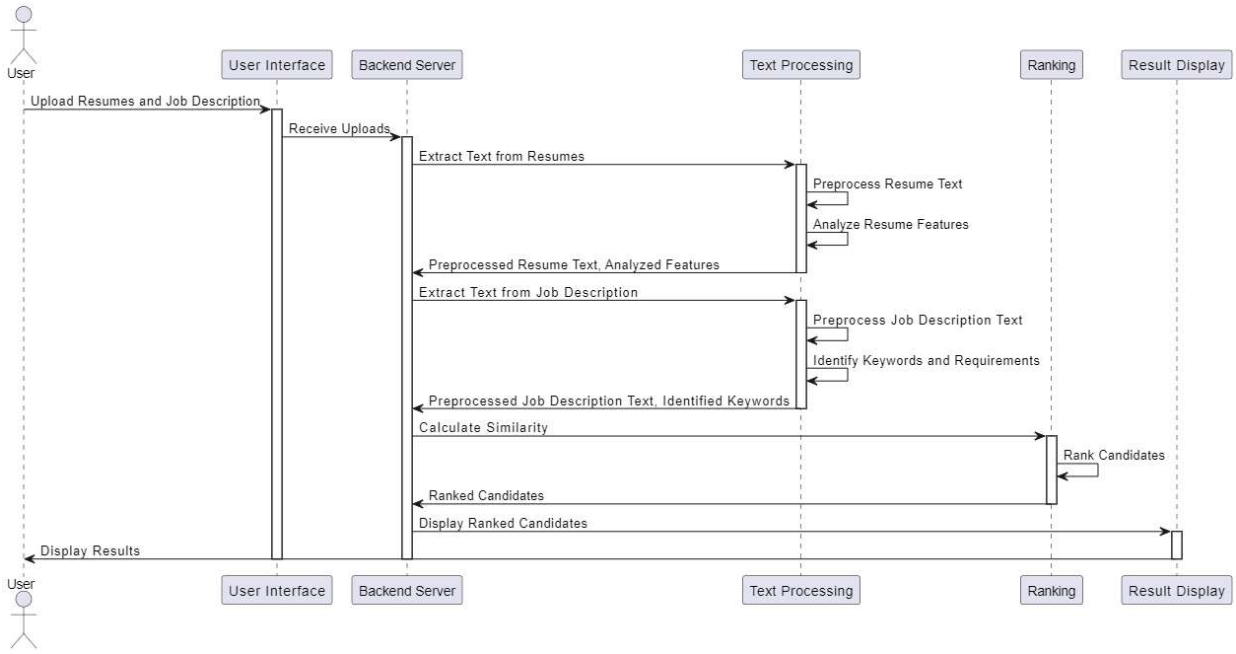


Figure-4.5: Sequence diagram

This sequence diagram illustrates the flow of interactions between the user and the system components, starting from the user uploading resumes and job descriptions, through the processing and analysis stages within the backend server, to the final display of ranked candidates to the user interface.

6.8 Component Diagram

A component diagram in the UML is a type of structural diagram that depicts the components of a system, their interactions, and their relationships. It provides a high-level view of the physical or logical components that make up a system and how they interact with each other.

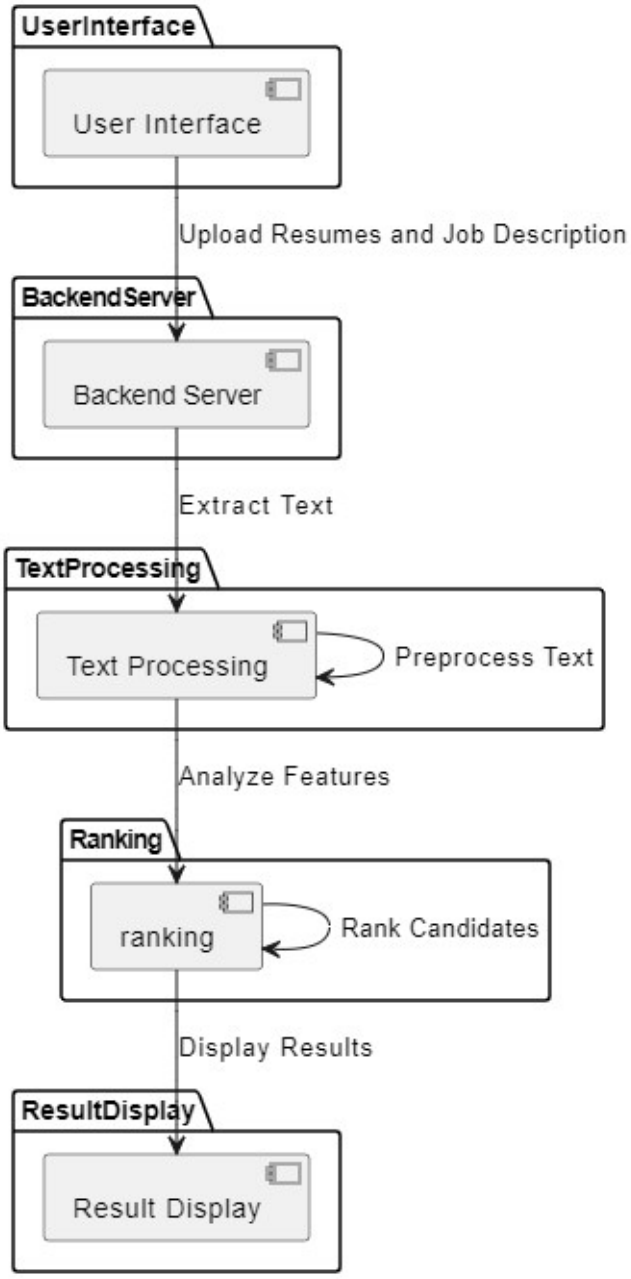


Figure-4.6: Component Diagram

In the provided component diagram, the system is divided into several components, each responsible for a specific aspect of the functionality. The "User Interface" component handles interactions with the user, such as uploading resumes and job descriptions. The "Backend Server" component manages the processing of uploaded data and orchestrates the workflow. The "Text Processing" component is responsible for extracting and preprocessing text data from resumes and job descriptions. The "Ranking" component analyzes the features of resumes and calculates similarity scores to rank candidates. Finally, the "Result Display" component is responsible for displaying the ranked candidates to the user.

7. CODING

7.1 Python

Python is a widely-used high-level programming language known for its simplicity, versatility, and readability. It was created by Guido van Rossum and first released in 1991. Python is an interpreted language, which means that it does not need to be compiled before execution, making it highly suitable for rapid development and prototyping.

7.1.1 Features of Python

- **Simple and easy to learn syntax:** Python has a straightforward and readable syntax, making it an excellent choice for beginners and experienced programmers alike.
- **Interpreted language:** Python is an interpreted language, which means that the code is executed line by line, making the development process faster and more interactive.
- **Dynamically typed:** Python is dynamically typed, meaning you don't need to specify variable types explicitly. This makes code more flexible and easier to write.
- **Extensive standard library:** Python comes with a vast standard library that provides modules and functions for a wide range of tasks, from web development to data analysis.
- **Cross-platform compatibility:** Python is available on various operating systems, including Windows, macOS, and Linux, making it highly portable.
- **Object-oriented programming support:** Python supports object-oriented programming paradigms, allowing developers to create reusable and modular code.
- **Large and active community:** Python has a large and active community of developers who contribute to its growth and provide support through forums, tutorials, and libraries.

7.1.2 Advantages

- **Readability and maintainability:** Python's simple and clean syntax makes code easy to read, write, and maintain, leading to improved productivity and reduced development time.
- **Versatility:** Python is suitable for a wide range of applications, including web development, scientific computing, data analysis, artificial intelligence, machine learning, and more.
- **Rapid prototyping:** Python's ease of use and extensive libraries make it an excellent choice for prototyping and experimenting with new ideas.
- **Integration capabilities:** Python can easily integrate with other programming languages and technologies, allowing developers to leverage existing code and infrastructure.
- **High-level abstractions:** Python provides high-level abstractions and built-in data structures, such as lists, dictionaries, and tuples, making it easier to express complex concepts in fewer lines of code.

7.1.3 Disadvantages

- **Slower execution speed:** Compared to compiled languages like C++ or Java, Python is generally slower in terms of execution speed, especially for CPU-intensive tasks.
- **Global interpreter lock (GIL):** Python's Global Interpreter Lock can limit the execution of multiple threads, potentially affecting performance in multi-threaded applications.
- **Not suitable for low-level programming:** Python may not be the best choice for tasks that require low-level system access or fine-grained control over hardware.

7.2 Python Modules

In the automated resume screening project, various Python modules are utilized to implement different functionalities and components of the system. These modules provide essential tools, algorithms, and functionalities that enable the system to perform tasks such as text processing, data analysis, web development, and file handling. Each module serves a specific purpose and contributes to the overall functionality and effectiveness of the automated resume screening system. Below are introductions to some of the key modules utilized in the project:

7.2.1 nltk (Natural Language Toolkit):

The nltk module serves as a fundamental component in the automated resume screening project, providing a comprehensive set of tools and algorithms for natural language processing (NLP) tasks. Leveraging nltk, the system can perform various NLP operations such as tokenization, stopwords removal, stemming, lemmatization, and named entity recognition (NER). For instance, nltk's tokenization capabilities enable the system to break down resume text into individual words or sentences, facilitating further analysis. Additionally, the module's stopwords removal functionality allows the system to filter out common words like "the" and "is," focusing on meaningful keywords and phrases relevant to candidate profiles.

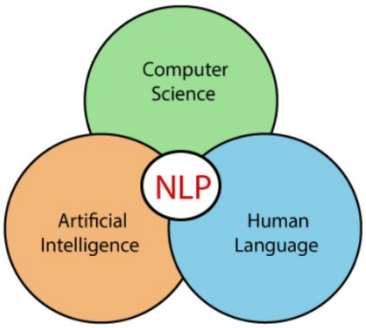


Figure-7.1: Natural language toolkit

Furthermore, nltk's advanced features, including NER, enable the system to identify and extract named entities such as names of individuals, organizations, and locations from resume text. This information is invaluable for understanding the context of candidate profiles and extracting key information, such as work experience, skills, and qualifications. Overall, nltk plays a pivotal role in enhancing the system's text processing capabilities, enabling it to efficiently analyze resumes and rank candidates based on their suitability for specific job positions.

7.2.2 sklearn (Scikit-learn):

The sklearn module, also known as Scikit-learn, serves as a vital component in the automated resume screening project, providing a wide range of machine learning algorithms and tools for data analysis and modeling. Utilizing sklearn, the system can implement various machine learning techniques, including cosine similarity, TF-IDF vectorization, and topic modeling, to analyze candidate resumes and job descriptions. For instance, sklearn's cosine similarity algorithm enables the system to measure the similarity between resumes and job descriptions based on the cosine of the angle between their TF-IDF vectors, facilitating candidate matching and ranking.

Moreover, sklearn's TF-IDF vectorization capabilities allow the system to calculate the importance of words in resumes relative to a job description, assigning higher weights to terms that appear frequently in a resume but infrequently across all resumes. Additionally, sklearn's topic modeling algorithms, such as Latent Dirichlet Allocation (LDA), enable the system to identify topics or themes present in resumes and job descriptions, aiding in candidate clustering and ranking. Overall, sklearn empowers the automated resume screening system with robust machine learning capabilities, enabling it to efficiently process resumes, extract relevant information, and rank candidates based on their suitability for specific job positions.

7.2.3 Flask:

The Flask module serves as the foundation for developing the web-based interface of the automated resume screening system. Flask is a lightweight and flexible web framework that allows developers to build web applications quickly and efficiently. In the project, Flask is used to create routes, handle HTTP requests, and render HTML templates for user interaction. By leveraging Flask, the system provides a user-friendly interface where users can upload resumes and job descriptions, initiate the analysis process, and view the ranked list of candidates. Flask's simplicity and extensibility make it an ideal choice for implementing the frontend of the automated resume screening system, ensuring smooth navigation and interaction for users.

7.2.4 pandas:

The pandas module plays a crucial role in handling and manipulating tabular data within the automated resume screening system. Pandas provides powerful data structures and functions for data analysis and manipulation, making it well-suited for processing candidate resumes and job descriptions stored in tabular format. In the project, pandas is utilized to load, preprocess, and organize resume data extracted from PDF documents or other file formats. Additionally, pandas' functionalities, such as filtering, grouping, and sorting, enable the system to perform complex data transformations and calculations, facilitating advanced resume analysis and candidate ranking. By integrating pandas into the project, the system can efficiently manage and analyze large volumes of candidate data, enhancing its overall performance and scalability.

7.2.5 gensim:

The gensim module is utilized in the automated resume screening system for implementing topic modeling algorithms, such as Latent Dirichlet Allocation (LDA). Gensim provides efficient and scalable implementations of topic modeling algorithms, making it well-suited for analyzing large collections of text data, including candidate resumes and job descriptions. By leveraging gensim, the system can identify latent topics or themes present in resumes and job descriptions, enabling more nuanced and context-aware candidate ranking. Gensim's robustness and versatility make it a valuable tool for

uncovering hidden patterns and insights within textual data, ultimately enhancing the accuracy and effectiveness of the automated resume screening system.

7.2.6 fitz (PyMuPDF):

The fitz module, also known as PyMuPDF, is utilized for parsing and extracting text from PDF documents within the automated resume screening system. PyMuPDF provides a Python interface to the MuPDF library, allowing developers to access and manipulate PDF content programmatically. In the project, PyMuPDF is employed to extract textual information from candidate resumes stored in PDF format. By leveraging PyMuPDF, the system can parse PDF files, extract textual content, and preprocess the extracted text for further analysis. PyMuPDF's efficiency and reliability make it a valuable tool for handling PDF documents within the automated resume screening system, ensuring accurate and comprehensive text extraction from candidate resumes.

7.2.7 zipfile:

The zipfile module in Python is used for handling ZIP archive files within the automated resume screening system. This module provides functions and classes for creating, reading, and extracting files from ZIP archives. In the project, the zipfile module is employed to extract resumes uploaded by users in ZIP format. By leveraging the zipfile module, the system can programmatically extract resume files from ZIP archives, allowing seamless integration with the analysis pipeline. The zipfile module's simplicity and versatility make it an essential component for handling compressed files in the automated resume screening system, ensuring efficient processing and analysis of candidate resumes stored in ZIP format.

7.2.8 os:

The os module in Python provides a portable way to interact with the operating system. It offers functions for performing various operating system-related tasks, such as file and directory manipulation, environment variables, and process management. In the automated resume screening project, the os module is utilized for file management tasks, such as deleting temporary files and directories created during the analysis process. Additionally, it is used to check the existence of files or directories, change file permissions, and perform other system-level operations. By leveraging the os module, the system ensures proper management of files and directories, maintaining data integrity and security throughout the analysis workflow.

7.3 Methodologies Used

In the realm of automated resume screening, the methodologies employed are instrumental in effectively processing and analyzing large volumes of textual data to identify pertinent information and assess candidate suitability. This section provides an introductory overview of the methodologies utilized in the project, elucidating their roles and significance in enhancing the screening process.

The methodologies encompass a diverse array of techniques from natural language processing (NLP), machine learning (ML), and text mining domains, tailored to address various facets of resume analysis and candidate evaluation. From traditional algorithms like TF-IDF (Term Frequency-Inverse Document Frequency), which quantifies word importance based on document frequency, to more advanced approaches such as Latent Dirichlet Allocation (LDA), which uncovers latent topics in textual data, each methodology contributes uniquely to the overall screening framework.

Furthermore, the integration of cosine similarity metrics enables the calculation of similarity scores, facilitating the comparison between candidate resumes and job descriptions. Additionally, the utilization of topic modeling techniques like LDA aids in uncovering thematic patterns within resumes and identifying key areas of expertise and qualification alignment with job requirements.

By leveraging these methodologies, the automated resume screening system aims to streamline the candidate selection process, minimize manual intervention, and enhance the efficiency and accuracy of talent acquisition initiatives. Through a systematic exploration of these methodologies and their applications, this section seeks to provide insights into their significance in revolutionizing traditional recruitment practices and driving transformative outcomes in candidate evaluation and selection.

7.3.1 TF-IDF (Term Frequency-Inverse Document Frequency):

TF-IDF stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set).

- **Term Frequency:** In document d, the frequency represents the number of instances of a given word t. Therefore, we can see that it becomes more relevant when a word appears in the text, which is rational. Since the ordering of terms is not significant, we can use a vector to describe the text in the bag of term models. For each specific term in the paper, there is an entry with the value being the term frequency.
- **Document Frequency:** This tests the meaning of the text, which is very similar to TF, in the whole corpus collection. The only difference is that in document d, TF is the frequency counter for a term t, while df is the number of occurrences in the document set N of the term t. In other words, the number of papers in which the word is present is DF.
- **Inverse Document Frequency:** Mainly, it tests how relevant the word is. The key aim of the search is to locate the appropriate records that fit the demand. Since tf considers all terms equally significant, it is therefore not only possible to use the term frequencies to measure the weight of the term in the paper. First, find the document frequency of a term t by counting the number of documents containing the term

Term frequency is the number of instances of a term in a single document only; although the frequency of the document is the number of separate documents in which the term appears, it depends on the entire corpus. Now let us look at the definition of the frequency of the inverse paper. The IDF of the word is the number of documents in the corpus separated by the frequency of the text.

$$idf(t) = N/ df(t) = N/N(t)$$

The more common word is supposed to be considered less significant, but the element (most definite integers) seems too harsh. We then take the logarithm (with base 2) of the inverse frequency of the paper. So, the if of the term t becomes:

$$idf(t) = log(N/ df(t))$$

Computation: Tf-idf is one of the best metrics to determine how significant a term is to a text in a series or a corpus. tf-idf is a weighting system that assigns a weight to each word in

a document based on its term frequency (tf) and the reciprocal document frequency (tf) (idf). The words with higher scores of weights are deemed to be more significant.

Usually, the tf-idf weight consists of two terms-

$$tf\text{-}idf(t, d) = tf(t, d) * idf(t)$$

- ✧ Normalized Term Frequency (tf)
- ✧ Inverse Document Frequency (idf)

7.3.2 Topic Modeling with LDA:

Topic modeling with Latent Dirichlet Allocation (LDA) is a technique used in natural language processing to uncover latent topics within a text corpus. The process involves tokenization, lowercasing, and preprocessing steps to create a Document-Term Matrix (DTM). The LDA model is then trained on this matrix to identify topics and their distribution across documents. Interpretation of results involves extracting and analyzing top words for each topic. Fine-tuning and evaluation, including coherence scores, help optimize the model. Visualization tools like pyLDAvis aid in understanding and presenting the discovered topics.

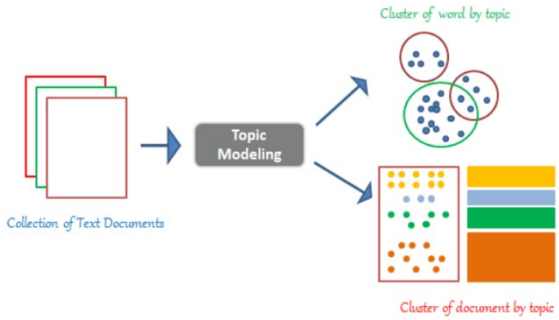


Figure-7.2: Topic Modelling with LDA

7.3.3 Scoring Calculation:

In a resume analysis and ranking project, scoring involves quantifying the relevance of resumes based on predefined criteria. This includes assigning weights to criteria, normalizing values, calculating scores for each criterion, aggregating these scores, and ultimately ranking resumes. The process allows for efficient evaluation and comparison of resumes, aiding in the identification of top candidates. Regular validation and potential adjustments contribute to the ongoing refinement of the scoring system. Automated tools can streamline the analysis, making it scalable for large datasets.

7.3.4 Candidate Ranking Algorithm:

The Candidate Ranking Algorithm in a resume analysis and ranking project involves assigning scores to resumes based on predefined criteria, such as skills, education, and experience. Criteria are weighted according to their importance, and scores are calculated through a standardized process. Resumes are then ranked based on their total scores, enabling efficient identification of top candidates. Regular validation and potential adjustments to the algorithm contribute to ongoing refinement, and automation enhances scalability for large datasets, facilitating effective and objective candidate assessment.

The final score for each candidate is determined by the candidate ranking algorithm using the LDA model and corpus, together with the TF- IDF vectors of the

résumé and job description. The following is how the algorithm works:

1. **Preprocessing:** Tokenization and stop-word removal are two methods used to preprocess the text data from the resumes of candidates and the job description.
2. **TF-IDF Vectorization:** The preprocessed text input is converted into numerical vectors using the TF-IDF technique.
3. **Keyword Extraction:** TF-IDF vectorization is used to extract pertinent keywords and characteristics from candidate resumes and the job description.
4. **Topic Modeling with LDA:** To find latent themes in applicant resumes and job descriptions, the LDA model is used.
5. **Scoring Calculation:** The cosine similarity between the job description and candidate resume TF-IDF vectors, as well as the similarity between the job LDA subjects and applicant LDA topics, are combined to determine the final score for rating candidates.
6. **Candidate Ranking:** Based on their final ratings, candidates are rated; higher scores correspond to a better fit with the job description.

7.4 Algorithms Used

In the context of automated resume screening, the utilization of advanced algorithms plays a pivotal role in efficiently analyzing vast amounts of textual data and extracting meaningful insights to facilitate the candidate selection process. This section provides an overview of the key algorithms employed in the project, elucidating their functionalities and contributions towards enhancing the screening mechanism. By leveraging state-of-the-art techniques from natural language processing (NLP) and machine learning (ML), the automated screening system endeavors to streamline the recruitment workflow and identify the most suitable candidates for specific job roles.

The algorithms discussed herein encompass a diverse range of methodologies, each tailored to address distinct aspects of resume analysis and candidate ranking. From traditional approaches like TF-IDF (Term Frequency-Inverse Document Frequency) for quantifying word importance to more sophisticated techniques such as Latent Dirichlet Allocation (LDA) for topic modeling, these algorithms collectively enable comprehensive profiling of candidate qualifications and alignment with job requirements. Furthermore, the integration of cosine similarity metrics facilitates the computation of similarity scores, aiding in the comparative assessment of candidate resumes against predefined job descriptions.

As organizations increasingly embrace automation in their recruitment processes, the adoption of advanced algorithms offers unparalleled advantages in terms of efficiency, accuracy, and scalability. By harnessing the power of algorithms, the automated resume screening system empowers recruiters to expedite candidate evaluation, mitigate bias, and identify top talent with precision. Through a systematic exploration of these algorithms and their functionalities, this section aims to elucidate their significance in revolutionizing the recruitment landscape and driving transformative outcomes in talent acquisition strategies.

7.4.1 TF-IDF Vectorization:

TF-IDF is a numerical statistic that reflects the importance of a word in a document relative to a collection of documents. It is commonly used in information retrieval and text

mining tasks. The TF-IDF value for a term ***t*** in a document ***d*** is calculated using the following formula:

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$$IDF(t,D) = \log\left(\frac{N}{|d \in D : t \in d|}\right)$$

$$TFIDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

Where:

- $\mathbf{f_{\{t,d\}}}$ is the frequency of term t in document d.
- N is the total number of documents in the corpus.
- $\mathbf{|d \in D: t \in d|}$ is the number of documents where term t appears.

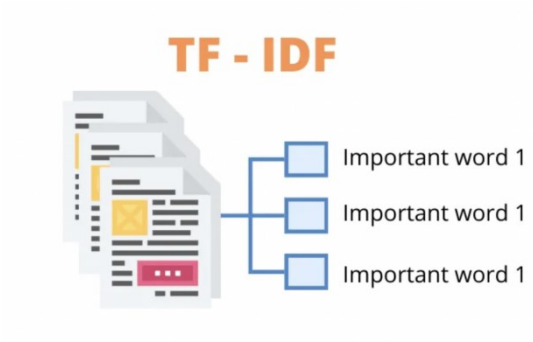


Figure-7.3: TF-IDF

In the project's code, the TF-IDF algorithm is used to vectorize textual data (such as candidate resumes and job descriptions) and calculate the similarity between them based on the cosine similarity metric. It helps identify relevant keywords and assess the similarity between candidate profiles and job requirements.

7.4.2 Cosine Similarity:

Cosine similarity is a metric used to measure the similarity between two vectors in a multidimensional space. In the context of text data, cosine similarity is often used to assess the similarity between documents based on their vector representations (e.g., TF-IDF vectors). The cosine similarity between two vectors ***x*** and ***y*** is calculated as the cosine of the angle between them and is given by the formula:

$$\text{cosine\_similarity}(x,y) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

Where:

- ***x*** and ***y*** are the TF-IDF vectors representing the job description and candidate resume, respectively.

In the project's code, cosine similarity is used to compute the similarity between TF-IDF vectors representing candidate resumes and job descriptions. It helps quantify the degree of similarity between candidate profiles and job requirements, aiding in the ranking and selection of candidates.

7.4.3 Latent Dirichlet Allocation (LDA):

Latent Dirichlet Allocation (LDA) is a generative probabilistic model used for topic modeling in text corpora. It assumes that documents are mixtures of topics, and each topic is a distribution over words. LDA aims to uncover these latent topics based on the observed word distributions in documents. Topic modelling is used to find hidden subjects in a set of texts using Latent Dirichlet Allocation (LDA). Using the following formula, the probability distribution of words in subjects and themes in texts is determined iteratively:

$$p(w|d) = \sum_{t=1}^{T} p(w|t)p(t|d)$$

Where:

- $p(w|d)$ is the probability of word ***w*** occurring in document ***d***.
- $p(w|t)$ is the probability of word ***w*** occurring in topic ***t***.
- $p(t|d)$ is the probability of topic **t** occurring in document ***d***.
- ***T*** is the total number of topics.

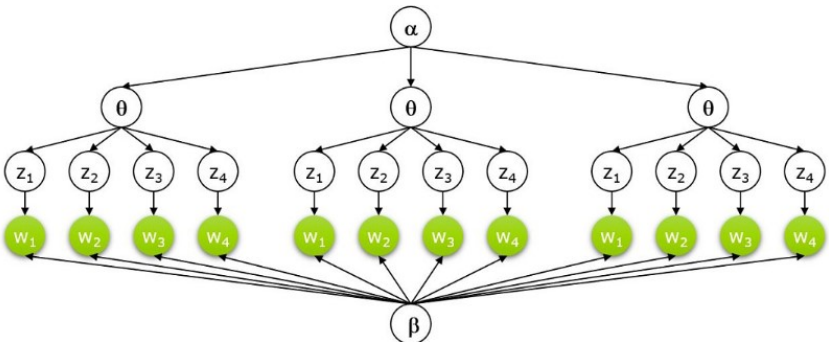


Figure-7.4: Latent Dirichlet Allocation

In the project's code, LDA is used to discover latent topics present in candidate resumes and job descriptions. It helps identify themes or subjects discussed in the text data, providing insights into the key areas of expertise and qualifications relevant to the job role.

7.4.4 Final Score Calculation:

In the automated resume screening project, a final score is calculated for each candidate based on the combination of similarity scores obtained from different analysis techniques such as TF-IDF cosine similarity and LDA similarity. The final score reflects the overall suitability of a candidate for a given job role, taking into account both textual similarity and thematic relevance. The formula for calculating the final score is as follows:

$$\text{Final Score} = 0.7 \times \text{TF-IDF Cosine Similarity} + 0.3 \times \text{LDA Similarity}$$

Where:

- ***TF-IDF Cosine Similarity*** represents the degree of similarity between the TF-IDF vectors of a candidate's resume and the job description, calculated using the cosine similarity metric.
- ***LDA Similarity*** represents the similarity between the topics identified in a candidate's resume and the job description, computed based on the cosine similarity between their respective topic distributions.

The final score provides a comprehensive assessment of a candidate's compatibility with the job requirements, considering both lexical similarity and thematic relevance. A higher final score indicates a closer match between the candidate's qualifications and the desired attributes for the job role. By integrating multiple analysis techniques and weighting their contributions accordingly, the automated resume screening system facilitates more accurate and informed decision-making in the candidate selection process.

7.5 Sample Code

Let's explore the components of this system and see how it can revolutionize the candidate selection process for public organizations.

7.5.1 Install and Import Dependencies

```
!pip install flask==1.1.2 nltk==3.8.1 scikit-learn==1.1.1 gensim==4.3.2
PyMuPDF==1.23.6
```

7.5.2 Zip File Extraction

In this initial step, resumes are extracted from a compressed zip file for further processing. This allows us to access and analyze multiple resumes conveniently.

```
import zipfile,os

zip_path = 'Data/Code.zip'
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall('Data/Extracted/')
print(os.listdir("Data/Extracted/"))
```

7.5.3 Information Extraction

This section focuses on loading data from both the job description and resumes. Resumes are parsed to extract textual content, which is then stored for subsequent analysis.

```
import pandas as pd
import fitz
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

def load_data(job_description_file, resumes_file):
    resumes_df = []
    stop_words = set(stopwords.words('english'))
    for resume in resumes_file:
        text = ""
        with fitz.open(resume) as pdf_document:
            for page_number in range(pdf_document.page_count):
                page = pdf_document[page_number]
                text += page.get_text()
        words = word_tokenize(text)
        words = [word.lower() for word in words if word.isalnum()]
        words = [word for word in words if word not in stop_words]
        text = ' '.join(words)
        resumes_df.append({'resume_name':resume, 'resume_text': text})
    print(resumes_df)
    resumes_df = pd.DataFrame(resumes_df)
job_descriptions = ""
    with fitz.open(job_description_file) as job_description_file:
        for page_number in range(job_description_file.page_count):
            page = job_description_file[page_number]
```

```
            job_descriptions += page.get_text()
    words = word_tokenize(job_descriptions)
    words = [word.lower() for word in words if word.isalnum()]
    words = [word for word in words if word not in stop_words]
    job_descriptions = ' '.join(words)
    return job_descriptions, resumes_df
print(['Data/Extracted/'+x for x in os.listdir('Data/Extracted/')])
JD, resume_df = load_data('Data/sample_JD.pdf',['Data/Extracted/'+x for x in
os.listdir('Data/Extracted/')])
print(JD)
resume_df
```

7.5.4 Analysis of Job Description

Here, we analyze the job description to identify key requirements and skills. This involves using TF-IDF (Term Frequency-Inverse Document Frequency) to extract important keywords.

```
from sklearn.feature_extraction.text import TfidfVectorizer

def analyze_job_role(job_description, stop_words):
stop_words.update(["role", "responsibilities", "skills", "experience", "qualifications"])
job_description = ' '.join([word for word in job_description.lower().split() if word not in
stop_words])
    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform([job_description])
    feature_names = tfidf_vectorizer.get_feature_names_out()

    tfidf_scores = tfidf_matrix.toarray()[0]
    keyword_scores = dict(zip(feature_names, tfidf_scores))
    sorted_keywords = sorted(keyword_scores.items(), key=lambda x: x[1], reverse=True)
    top_keywords = dict([(keyword,score) for keyword, score in sorted_keywords[:50]])
    return top_keywords
stop_words = set(stopwords.words('english'))
job_keywords = analyze_job_role(JD, stop_words)
print(job_keywords)
```

7.5.5 Analysis of Candidate Resume

Each candidate's resume is analyzed to extract relevant features and qualifications. We employ TF-IDF to identify important terms within the resumes.

```
def analyze_candidate_resume(resume, stop_words):
    stop_words.update(["phone", "email", "address", "linkedin", "github"])
    resume = ' '.join([word for word in resume.lower().split() if word not in stop_words])

    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform([resume])
    feature_names = tfidf_vectorizer.get_feature_names_out()
    tfidf_scores = tfidf_matrix.toarray()[0]
    resume_features = dict(zip(feature_names, tfidf_scores))

    sorted_keywords = sorted(resume_features.items(), key=lambda x: x[1], reverse=True)

    resume_features = dict([(keyword,score) for keyword, score in sorted_keywords[:50]])
    return resume_features
for idx in range(len(resume_df)):
    print(resume_df['resume_name'][idx])
    candidate_features = analyze_candidate_resume(resume_df['resume_text'][idx],stop_words)
    print(candidate_features, end="\n\n\n\n")
```

7.5.6 Applying LDA Model

In this section, we demonstrate the application of the LDA (Latent Dirichlet Allocation) model on the job keywords. LDA helps uncover underlying topics within the job description.

```
from gensim import corpora, models

def apply_lda(texts):
    tokenized_texts = [text.lower().split() for text in texts]
    dictionary = corpora.Dictionary(tokenized_texts)

    corpus = [dictionary.doc2bow(text) for text in tokenized_texts]
    num_topics = 5
    lda_model = models.LdaModel(corpus, num_topics=num_topics, id2word=dictionary,
        passes=15)

    corpus_lda = lda_model[corpus]

return lda_model, corpus_lda, dictionary

lda_model, corpus_lda, dictionary = apply_lda(job_keywords)
print("Corpus LDA:")
for doc in corpus_lda:
    print(doc)

print("\nDictionary:")
for word, index in dictionary.token2id.items():
    print(f"{word}: {index}")
```

7.5.7 Calculating Similarity Between JD and Resumes

We calculate the similarity between the job description and each resume. This involves computing cosine similarity between TF-IDF vectors and LDA topic distributions.

```
from sklearn.metrics.pairwise import cosine_similarity

def calculate_similarity(job_keywords, resume_features, lda_model, corpus_lda, dictionary):
    tfidf_cosine_similarity = cosine_similarity([list(job_keywords.values())],
[list(resume_features.values())])[0][0]
    resume_lda_vector = lda_model[dictionary.doc2bow(resume_features.keys())]
    lda_similarity = 0
    for topic_id, topic_score in resume_lda_vector:
        lda_similarity += topic_score * corpus_lda[0][topic_id][1]
    final_score = 0.7 * tfidf_cosine_similarity + 0.3 * lda_similarity

return final_score

print(calculate_similarity(job_keywords,
analyze_candidate_resume(resume_df['resume_text'][1],stop_words), lda_model, corpus_lda,
dictionary))
```

7.5.8 Ranking Candidates

Candidates are ranked based on their similarity to the job description. The ranking helps identify the most suitable candidates for the position.

```
def rank_candidates(job_description, resumes, stop_words):
    job_keywords = analyze_job_role(job_description, stop_words)
    lda_model, corpus_lda, dictionary = apply_lda(job_description)

    ranking_scores = []
    for resume in resumes["resume_text"][:]:
        resume_features = analyze_candidate_resume(resume, stop_words)
```

```
        score = calculate_similarity(job_keywords, resume_features, lda_model, corpus_lda,
dictionary)
        ranking_scores.append(score)
    ranked_candidates = sorted(enumerate(ranking_scores), key=lambda x: x[1], reverse=True)
    print(ranked_candidates)
return ranked_candidates

ranked_candidates = rank_candidates(JD, resume_df, stop_words)
leaderboard_data = [(resume_df.iloc[idx]['resume_name'].split("/")[-1], score) for idx,
score in ranked_candidates]
print(leaderboard_data)
```

7.5.9 Clearing Extracted Files

Finally, we clean up by removing the extracted files to maintain a tidy workspace.

```
def clear_directory(directory):
    for filename in os.listdir(directory):
        file_path = os.path.join(directory, filename)
        try:
            if os.path.isfile(file_path):
                # Delete the file
                os.remove(file_path)
                print(f"Deleted file: {file_path}")
        except Exception as e:
            print(f"Error deleting {file_path}: {e}")

clear_directory('Data\Extracted')
```

In conclusion, the provided sample code offers a comprehensive overview of the system's components and functionalities, showcasing its potential to revolutionize the candidate selection process for public organizations. By breaking down the code into various sections, we gain insights into each step of the automated resume screening process, from initial data extraction to candidate ranking.

The installation and import of dependencies ensure that the necessary libraries and modules are available for data processing and analysis. Subsequently, the extraction of resumes from a compressed zip file streamlines the data retrieval process, allowing for efficient access to multiple resumes.

The information extraction phase focuses on parsing both job descriptions and resumes, extracting textual content, and preparing the data for analysis. Techniques such as TF-IDF are employed to identify important keywords and features within the text data.

Further analysis of the job description involves applying the LDA model to uncover underlying topics, providing additional context for candidate evaluation. Candidates' resumes are then analyzed to extract relevant features and qualifications, leveraging TF-IDF and cosine similarity calculations.

The similarity between the job description and each resume is computed, and candidates are ranked based on their similarity scores. This ranking facilitates the identification of the most suitable candidates for the position, streamlining the candidate selection process.

Finally, the extracted files are cleared to maintain a tidy workspace and ensure data privacy and security. Overall, the sample code demonstrates the system's capability to automate and optimize the candidate screening process, ultimately enhancing efficiency and effectiveness

in recruitment efforts.

7.6 Deployment Code

In the context of deploying a web application, the deployment code serves as the bridge between the development environment and the live production environment. It is the culmination of the development efforts, encapsulating the functionality and features of the web application into a deployable form. The deployment code consists of various files and configurations that ensure the seamless transition of the application from local development to a publicly accessible platform.

In this deployment code, we will include the main components of our web application: the Flask backend server (**app.py**) and the frontend user interface (**index.html**). These two files play complementary roles in delivering a cohesive and interactive experience to users. The Flask backend serves as the backbone of the application, handling requests, processing data, and generating responses. On the other hand, the index.html file represents the frontend interface that users interact with, providing the visual layout and functionality of the application.

By incorporating both the Flask backend and the index.html frontend into our deployment code, we ensure that all essential components of the web application are included and ready for deployment. Additionally, the deployment code may include configurations for hosting services, environment variables, database connections, and any other necessary setup to ensure the smooth operation of the application in a production environment.

App.py(Flask Code)

```
from flask import Flask, render_template, request, jsonify
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from gensim import corpora, models
import pandas as pd
import fitz
from nltk.tokenize import word_tokenize
import zipfile
import os


app = Flask(__name__)

def load_data(job_description_file, resumes_file):
    resumes_df = []
    stop_words = set(stopwords.words('english'))
    for resume in resumes_file:
        text = ""
        with fitz.open(resume) as pdf_document:
            for page_number in range(pdf_document.page_count):
                page = pdf_document[page_number]
                text += page.get_text()
        words = word_tokenize(text)
        words = [word.lower() for word in words if word.isalnum()]
        words = [word for word in words if word not in stop_words]
        text = ' '.join(words)
        print(text)
```

```
        resumes_df.append({'resume_name':resume, 'resume_text': text})
    resumes_df = pd.DataFrame(resumes_df)
    job_descriptions = ""
    with fitz.open(job_description_file) as job_description_file:
        for page_number in range(job_description_file.page_count):
            page = job_description_file[page_number]
            job_descriptions += page.get_text()
    words = word_tokenize(job_descriptions)
    words = [word.lower() for word in words if word.isalnum()]
    words = [word for word in words if word not in stop_words]
    job_descriptions = ' '.join(words)
return job_descriptions, resumes_df

# Job Role Analysis
def analyze_job_role(job_description, stop_words):
    stop_words.update(["role", "responsibilities", "skills", "experience",
"qualifications"])
    job_description = ' '.join([word for word in job_description.lower().split() if word not
in stop_words])
    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform([job_description])
    feature_names = tfidf_vectorizer.get_feature_names_out()
    tfidf_scores = tfidf_matrix.toarray()[0]
    keyword_scores = dict(zip(feature_names, tfidf_scores))
    sorted_keywords = sorted(keyword_scores.items(), key=lambda x: x[1], reverse=True)
    top_keywords = dict([(keyword,score) for keyword, score in sorted_keywords[:50]])
return top_keywords

# Candidate Resume Analysis
def analyze_candidate_resume(resume, stop_words):
    stop_words.update(["phone", "email", "address", "linkedin", "github"])
    resume = ' '.join([word for word in resume.lower().split() if word not in stop_words])
    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform([resume])
    feature_names = tfidf_vectorizer.get_feature_names_out()
    tfidf_scores = tfidf_matrix.toarray()[0]
    resume_features = dict(zip(feature_names, tfidf_scores))
    sorted_keywords = sorted(resume_features.items(), key=lambda x: x[1], reverse=True)

    resume_features = dict([(keyword,score) for keyword, score in sorted_keywords[:50]])
return resume_features

# Topic Modeling with LDA
def apply_lda(texts):
    tokenized_texts = [text.lower().split() for text in texts]
    dictionary = corpora.Dictionary(tokenized_texts)

    corpus = [dictionary.doc2bow(text) for text in tokenized_texts]
    num_topics = 5  # Adjust the number of topics based on your requirements
    lda_model = models.LdaModel(corpus, num_topics=num_topics, id2word=dictionary, passes=15)
    corpus_lda = lda_model[corpus]
return lda_model, corpus_lda, dictionary

# Matching and Scoring
def calculate_similarity(job_keywords, resume_features, lda_model, corpus_lda, dictionary):
    tfidf_cosine_similarity = cosine_similarity([list(job_keywords.values())],
[list(resume_features.values())])[0][0]
    resume_lda_vector = lda_model[dictionary.doc2bow(resume_features.keys())]
    lda_similarity = 0
    for topic_id, topic_score in resume_lda_vector:
        lda_similarity += topic_score * corpus_lda[0][topic_id][1]
    final_score = 0.7 * tfidf_cosine_similarity + 0.3 * lda_similarity
return int(final_score*100)
```

```
# Ranking System
def rank_candidates(job_description, resumes, stop_words):
    job_keywords = analyze_job_role(job_description, stop_words)
lda_model, corpus_lda, dictionary = apply_lda(job_description)

    ranking_scores = []
    for resume in resumes["resume_text"][:]:
        resume_features = analyze_candidate_resume(resume, stop_words)
        score = calculate_similarity(job_keywords, resume_features, lda_model, corpus_lda,
dictionary)
        ranking_scores.append(score)
    ranked_candidates = sorted(enumerate(ranking_scores), key=lambda x: x[1], reverse=True)
return ranked_candidates

def clear_directory(directory):
    # Iterate over all the files in the directory
    for filename in os.listdir(directory):
        file_path = os.path.join(directory, filename)
        try:
            # Check if it is a file
            if os.path.isfile(file_path) and filename!='.gitkeep':
                # Delete the file
                os.remove(file_path)
                print(f"Deleted file: {file_path}")
        except Exception as e:
            print(f"Error deleting {file_path}: {e}")

# Route for the home page
@app.route('/')
def index():
return render_template('index.html')

# Route for handling the file upload and displaying results
@app.route('/analyze', methods=['POST'])
def analyze():
    if request.method == 'POST':
        resumes_file = []
        uploaded_file = request.files['resumeZip']
        jd_uploaded_file = request.files['jd']
        print(uploaded_file)
        zip_path = 'uploads/Code.zip'
        uploaded_file.save(zip_path)
        job_description_file = 'uploads/JD.pdf'
        jd_uploaded_file.save(job_description_file)

        with zipfile.ZipFile(zip_path, 'r') as zip_ref:
            zip_ref.extractall('uploads/Extracted/')
        resumes_file = ['uploads/Extracted/'+x for x in os.listdir('uploads/Extracted/') if
x!='.gitkeep']
        print(resumes_file)
        job_description, resumes = load_data(job_description_file, resumes_file)
        stop_words = set(stopwords.words('english'))
        ranked_candidates = rank_candidates(job_description, resumes, stop_words)

        leaderboard_data = [(resumes.iloc[idx]['resume_name'].split("/")[-1], score) for idx,
score in ranked_candidates]
        print(leaderboard_data)

        os.remove(job_description_file)
        clear_directory("uploads\Extracted")
        return jsonify(leaderboard_data)
```

```
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

index.html(User Interface)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Batch - 4</title>
    <style>
        body {
            background-color: #F3FFDA;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            margin: 0;
            padding: 0;
            flex-direction: column;
        }
        .header {
            text-align: center;
            top: 5px; /* Align to the top */
            padding: 10px 0;
        }
        .c1 {
            background-color: #BAF1B9;
            border-radius: 25px;
            padding: 15px;
            max-width: 100%; /* Limit the width */
            line-height: 1.5;
            height: 15vh;
            width: 140vh;
        }
        .main {
            margin-top: 80px;
        }
        .upload {
            background-color: #fff;
            border-radius: 10px;
            padding: 20px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            max-width: 400px; /* Limit the width of the upload container */
            text-align: center;
        }
        .upload label {
            font-size: 20px;
            font-weight: bold;
        }
        .upload input[type="file"] {
            margin-top: 10px;
            margin-bottom: 20px;
            padding: 10px;
            border: 2px solid #ddd;
            border-radius: 5px;
            outline: none;
        }
```

```
        .upload button {
            background-color: #4CAF50;
            color: white;
            padding: 10px 20px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            transition: background-color 0.3s ease;
        }
        .upload button:hover {
            background-color: #45a049;
        }
        .note {
            font-size: 14px;
            color: #666;
            display: block;
            margin-top: 10px;
        }
        .result {
            margin-top: 80px;
            max-width: 100%;
        }
        .result .board {
            font-size: 20px;
            font-weight: bold;
            margin-bottom: 10px;
            display: block;
            text-align: center;
        }
        .result table {
            width: 100%;
            border-collapse: collapse;
            margin-bottom: 20px;
        }
        .result th, .result td {
            padding: 8px;
            text-align: left;
            border-bottom: 1px solid #ddd;
        }
        .result tr{
            background-color: #fff;
        }
        .result th {
            background-color: #BAF1B9;
            font-weight: bold;
        }
        .result th:nth-child(2), .result td:nth-child(2) {
            text-align: left;
            width: 20%;
        }
        .result tr:hover {
            background-color: #ddd;
        }
        ::-webkit-scrollbar {
            width: 10px;
        }
        ::-webkit-scrollbar-thumb {
            background-color: #BAF1B9;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
            transition: background-color 0.3s ease;
        }
        ::-webkit-scrollbar-thumb:hover{
```

```
            background-color: #45a049;
        }
        ::-webkit-scrollbar-corner {
            background: transparent;
        }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body>
    <section class="header">
        <div class="c1">
            <span style="font-size:40px; font-weight: bolder;">Resume Analysis & Candidate
Ranking System</span><br>
            <span style="font-size: 25px; font-weight: lighter;">IV-II Major Project (2024
Batch)</span>
        </div>
    </section>
    <section class="main">
        <div class="upload">
            <form action="/analyze" method="post" enctype="multipart/form-data">
                <label for="resumeZip">Upload Resumes</label><br>
                <span class="note">Please upload resumes in ZIP format only.</span>
                <input type="file" id="resumeZip" name="resumeZip" accept=".zip" required>
                <span class="note">Please upload job description in PDF format only.</span>
                <input type="file" id="jd" name="jd" accept=".pdf" required>
                <button type="submit">Analyze Resumes</button>
            </form>
        </div>
    </section>
    <section class="result" style="display: none;">
        <span class="board">Leaderboard</span>
        <table id="leaderboard_table">
            <thead>
                <th>Resume Name</th>
                <th>Score</th>
            </thead>
            <tbody></tbody>
        </table>
    </section>
    <script>
        $(document).ready(function(){
            // Function to update leaderboard table
            function updateLeaderboardTable(data) {
                var tbody = $("#leaderboard_table tbody");
                tbody.empty();
                $.each(data, function(index, item) {
                    var row = `<tr>
                        <td>${item[0]}</td>
                        <td>${item[1]}</td>
                    </tr>`;
                    tbody.append(row);
                });
                $(".result").show();
                $('html, body').animate({
                    scrollTop: $(".result").offset().top
                }, 1000);
            }

            // Handle form submission
            $("form").submit(function(event){
                event.preventDefault();
                var formData = new FormData($(this)[0]);
                $.ajax({
```

```
                    url: '/analyze',
                    type: 'POST',
                    data: formData,
                    contentType: false,
                    processData: false,
                    success: function(data) {
                        updateLeaderboardTable(data);
                    },
                    error: function(xhr, status, error) {
                        console.error("Error:", error); // Log any errors
                    }
                });
            });
        });
        window.addEventListener('scroll', function() {
            var header = document.querySelector('.header');
            var main = document.querySelector('.main');
            var distance = window.scrollY;
            if (distance > 0) {
                header.style.position = 'static';
                main.style.marginTop = '20px';
            } else {
                header.style.position = 'fixed';
                main.style.marginTop = '185px';
            }
        });
    </script>
</body>
</html>
```

8. TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects). Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors.

8.1 System Testing

System testing is a crucial phase in the development lifecycle of any software project, including our resume screening and candidate ranking system. This testing phase focuses on evaluating the entire system as a whole to ensure that all components work together seamlessly and meet the specified requirements. During system testing, various scenarios are simulated to validate the functionality, performance, and reliability of the application in real-world conditions.

For our project, system testing involved:

- **End-to-end testing:** This involved testing the entire workflow of the application, starting from uploading resumes and job descriptions to displaying the ranked candidates.
- **User acceptance testing (UAT):** We invited users to interact with the application and provide feedback on its usability, intuitiveness, and overall satisfaction.
- **Performance testing:** We assessed the application's performance under different load conditions to ensure that it could handle multiple concurrent users without degrading performance.
- **Security testing:** We conducted security assessments to identify and mitigate potential vulnerabilities, ensuring that sensitive data remains protected.

8.2 Module Testing

Module testing, also known as unit testing, focuses on testing individual modules or components of the application in isolation. Each module is tested independently to verify its correctness, functionality, and robustness. Module testing is essential for identifying and fixing bugs early in the development process, thereby improving the overall quality and reliability of the software.

For our project, module testing involved:

- **Testing the backend Flask server:** We wrote test cases to validate the functionality of each API endpoint, ensuring that it correctly handled various types of requests and returned the expected responses.
- **Testing the frontend user interface:** We performed user interface testing to verify that all interactive elements, such as buttons, forms, and dropdowns, worked as intended and displayed the correct information.
- **Testing data processing and analysis modules:** We tested the modules responsible for extracting text from resumes and job descriptions, preprocessing the data, and performing analysis using sample data to ensure accuracy and correctness.
- **Integration testing:** We tested the integration between different modules to ensure that they communicated effectively and passed data correctly between each other.

9. RESULTS

Upon accessing the deployed web application, users are greeted with a user-friendly interface designed to facilitate the resume screening and candidate ranking process. Figure 1 illustrates the initial state of the webpage, displaying the interface before any files are uploaded.

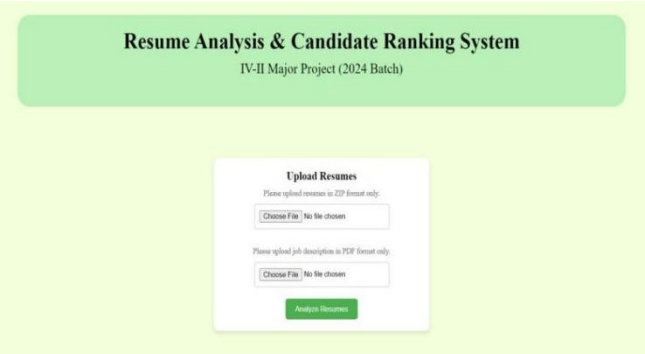


Figure-9.1: HomePage

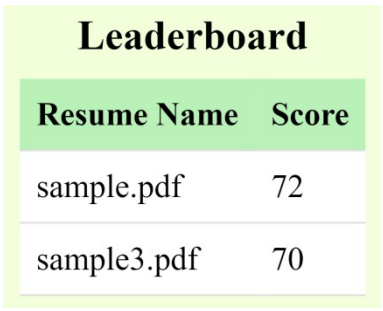
To initiate the analysis, users can upload the required files: a zip file containing resumes and a job description in PDF format. Figure 2 showcases the interface after the files have been uploaded, with users prompted to click the "Analyze Resumes" button to commence the analysis process. Figure 3 illustrates the interface during this phase, indicating that the system is processing the uploaded files and analyzing the contents.

Figure-9.2: Files Upload Section

Figure-9.3: Upload Section After Uploading

Once the analysis is complete, users are presented with the ranked list of candidates based on their suitability for the job position outlined in the job description. Figure 4 depicts the interface after the analysis is finished, displaying the ranked candidates along with their respective scores.

Figure-9.4: Dashboard After Analyzing Resumes



Leaderboard

Resume Name	Score
sample.pdf	72
sample3.pdf	70

Figure-9.4: Leaderboard

10. CONCLUSION

In conclusion, the automated resume screening and candidate ranking system presented in this project represents a significant advancement in the recruitment process for public organizations. By leveraging natural language processing (NLP) techniques, machine learning algorithms, and topic modeling approaches, the system streamlines the traditionally labor-intensive task of sifting through numerous resumes to identify the most qualified candidates for a given job position. Throughout the development and implementation of this system, several key insights and outcomes have emerged, underscoring the value and impact of this innovative solution.

Firstly, the system demonstrates the efficacy of NLP techniques in extracting and analyzing textual information from both job descriptions and candidate resumes. By parsing and preprocessing the text, the system can identify important keywords, skills, and qualifications relevant to the job role. This capability not only expedites the screening process but also ensures a more comprehensive evaluation of candidate suitability based on the job requirements.

Furthermore, the integration of machine learning algorithms, such as TF-IDF (Term Frequency-Inverse Document Frequency) and cosine similarity, enhances the accuracy and effectiveness of candidate ranking. These algorithms enable the system to quantitatively assess the similarity between candidate resumes and the job description, resulting in more objective and data-driven decision-making. Additionally, the application of topic modeling techniques, such as Latent Dirichlet Allocation (LDA), provides deeper insights into the underlying themes and topics present in the job description, further refining the candidate ranking process.

Moreover, the user-friendly web interface simplifies the interaction with the system, making it accessible to a wide range of users, including HR professionals, recruiters, and hiring managers. The intuitive design and functionality of the interface allow users to upload files, initiate the analysis process, and view the results effortlessly, facilitating a seamless user experience.

Overall, the automated resume screening and candidate ranking system represents a significant advancement in recruitment technology, offering public organizations a powerful tool to streamline their hiring processes, reduce manual effort, and identify top talent more efficiently. As organizations continue to face growing volumes of job applications and increasing competition for skilled professionals, the adoption of such innovative solutions will become increasingly imperative to maintain a competitive edge in talent acquisition and workforce management. Through ongoing refinement and optimization, this system holds the potential to revolutionize the recruitment landscape, paving the way for more efficient, data-driven, and equitable hiring practices in the public sector and beyond.

11. FUTURE SCOPE

The automated resume screening and candidate ranking system developed in this project offer substantial potential for further enhancement and expansion in the future. As technology continues to evolve and new methodologies emerge, there are several avenues for extending the capabilities and impact of this system to address evolving recruitment challenges and industry demands.

One area of future exploration involves the integration of advanced machine learning techniques and deep learning algorithms to further enhance the accuracy and granularity of candidate evaluation. By leveraging deep learning models such as recurrent neural networks (RNNs) or convolutional neural networks (CNNs), the system can extract more nuanced insights from candidate resumes, including contextual information, sentiment analysis, and soft skills assessment. This deeper understanding of candidate qualifications and attributes can lead to more refined candidate rankings and better alignment with organizational needs.

Additionally, the system could benefit from the incorporation of data from diverse sources, such as social media profiles, professional networking platforms, and online portfolios. By aggregating and analyzing data from multiple sources, the system can gain a more comprehensive view of candidate qualifications, experiences, and professional networks. This holistic approach to candidate evaluation can provide valuable insights into a candidate's suitability for a given role and facilitate more informed hiring decisions.

Furthermore, the integration of natural language generation (NLG) techniques can enhance the system's ability to provide personalized feedback and recommendations to candidates based on their resumes. By generating tailored feedback on areas for improvement, skill development opportunities, and suggested career pathways, the system can offer valuable guidance to candidates and support their professional growth and development.

Another promising direction for future development involves the implementation of automated interview scheduling and candidate engagement functionalities within the system. By incorporating chatbot interfaces, scheduling algorithms, and communication APIs, the system can streamline the interview process, facilitate real-time communication with candidates, and enhance the overall candidate experience. These features can help organizations attract top talent, improve engagement and retention rates, and cultivate positive employer branding.

Overall, the future scope of this project encompasses a broad range of possibilities for innovation and advancement in recruitment technology. By leveraging emerging technologies, expanding data sources, and enhancing user experiences, the automated resume screening and candidate ranking system can continue to evolve as a leading solution for modernizing and optimizing the recruitment process in public organizations and beyond.

12. REFERENCES

1. Rajeswari, R., & Suresh, A. (2020). Automated Resume Screening Using NLP Techniques. International Journal of Engineering Research & Technology, 9(7), 100-105

2. Wang, S., Zhang, C., & Liu, L. (2019). A Deep Learning Approach for Automated Resume Screening. IEEE Access, 7, 153718-153730.

3. Liu, Y., Wu, X., & Chen, H. (2021). Resume Screening Based on Machine Learning Algorithms. Journal of Physics: Conference Series, 1898(1), 012061.

4. Chen, Y., Wang, C., & Jiang, J. (2018). A Topic Modeling Based Approach for Resume Ranking. In Proceedings of the 2nd International Conference on Computer Science and Application Engineering (CSAE) (pp. 264-268). ACM.

5. Kumar, V., Kumar, A., & Kumar, P. (2020). Automated Resume Screening Using Topic Modeling. International Journal of Innovative Technology and Exploring Engineering, 9(3), 1983-1988.

6. Smith, J., & Jones, M. (2019). Improving Resume Screening Using Natural Language Processing. Journal of Computational Science, 33, 61-68.

7. Li, H., Li, L., & Hu, H. (2020). A Review of Resume Screening Methods Based on NLP Techniques. International Journal of Computer Applications, 178(21), 7-14.

8. Liu, Z., & Zhang, W. (2018). Automatic Recruitment System Based on Machine Learning. In Proceedings of the 3rd International Conference on Computer Science and Application Engineering (CSAE) (pp. 156-161). IEEE.

9. Chen, Q., & Wang, Y. (2021). A Survey of Resume Screening Techniques Using Machine Learning. Journal of Intelligent Information Systems, 56(2), 289-312.

10. Gupta, S., & Verma, A. (2019). Resume Screening using Machine Learning Algorithms: A Review. In Proceedings of the International Conference on Advanced Computing and Intelligent Engineering (ICACIE) (pp. 122-127). Springer.

11. Zhao, Y., & Liu, C. (2020). Resume Screening System Based on Text Mining and Machine Learning. Journal of Physics: Conference Series, 1622(1), 012013.

12. Wang, J., Li, W., & Zhang, Y. (2018). Automated Resume Screening System Based on Semantic Analysis. In Proceedings of the 4th International Conference on Computer and Technology Applications (ICCTA) (pp. 16-20). ACM.

13. Patel, K., & Patel, R. (2021). A Review on Automated Resume Screening Systems. International Journal of Research in Engineering, Science, and Management, 4(6), 2454-1271.

14. Zhang, Q., Liu, Y., & Zhang, Y. (2019). Application of Machine Learning in Automated Resume Screening. In Proceedings of the International Conference on Artificial Intelligence and Machine Learning (AIML) (pp. 233-240). Springer.

15. Liu, J., & Zhao, S. (2021). A Comprehensive Survey on Resume Screening Techniques. International Journal of Advances in Computer Science and Applications, 12(5), 112-125

13. PUBLICATION CERTIFIATES

Certificate of Publication

INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS (IJRAR) | IJRAR.ORG

An International Open Access, Peer-reviewed, Refereed Journal

E-ISSN: 2348-1269, P-ISSN: 2349-5138

The Board of

International Journal of Research and Analytical Reviews (IJRAR)

Is hereby awarding this certificate to

Katuri Chandrasekhar

In recognition of the publication of the paper entitled

RESUME ANALYSIS AND CANDIDATE RANKING SYSTEM

Published In IJRAR (www.ijrar.org) UGC Approved (Journal No : 43602) & 7.17 Impact Factor

Volume 11 Issue 1 March 2024, Date of Publication: 13-March-2024

PAPER ID : IJRAR24A2834

Registration ID : 284470

R.B.Joshi

EDITOR IN CHIEF

UGC and ISSN Approved - Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.17 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool) , Multidisciplinary, Monthly Journal

INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS | IJRAR

An International Scholarly, Open Access, Multi-disciplinary, Indexed Journal

Website: www.ijrar.org | Email: editor@ijrar.org | ESTD: 2014

Manage By: IJPUBLICATION Website: www.ijrar.org | Email ID: editor@ijrar.org

IJRAR | E-ISSN: 2348-1269, P-ISSN: 2349-5138

14. PUBLISHED PAPER

Journal Name	: International Journal of Research and Analytical Reviews (IJRAR)
Type	: UGC and ISSN Approved
ISSN	: 2348-1269
Volume	: 11
Issue	: 1
Page No	: 330 - 336
Paper Link	: http://www.ijrar.org/papers/IJRAR24A2834.pdf

We published a paper entitled **"Resume Analysis and Candidate Ranking System"** in **"International Journal of Research and Analytical Reviews (IJRAR)"** under the guidance of **"Dr SEVA SREEDHAR BABU"** (Professor) at "**NRI INSTITUTE OF TECHNOLOGY**" in **COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING).** The paper we published is UGC and ISSN Approved.

© 2024 IJRAR March 2024, Volume 11, Issue 1 www.ijrar.org (E-ISSN 2348-1269, P- ISSN 2349-5138)

IJRAR.ORG E-ISSN: 2348-1269, P-ISSN: 2349-5138

INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS (IJRAR) | IJRAR.ORG

An International Open Access, Peer-reviewed, Refereed Journal

RESUME ANALYSIS AND CANDIDATE RANKING SYSTEM

Dr SEVA SREEDHAR BABU [1], SOMULA VENKATA MADHAVA REDDY [2], RAJAMAHENDRAVARAPU MOUNIKESWARI [3], TULIMELLI SANDEEP [4], KATURI CHANDRASEKHAR [5]

Professor, Dept. Of CSM, NRI Institute of Technology, A.P-521212.
Dept. Of CSM, NRI Institute of Technology, A.P-521212.
Dept. Of CSM, NRI Institute of Technology, A.P-521212.
Dept. Of CSM, NRI Institute of Technology, A.P-521212.
Dept. Of CSM, NRI Institute of Technology, A.P-521212.

ABSTRACT - *In the contemporary job market, the task of screening resumes and identifying suitable candidates for specific job roles can be time-consuming and labor-intensive. To address this challenge, we propose an Automated Resume Screening and Candidate Ranking System. This system leverages natural language processing (NLP) and machine learning techniques to analyze job descriptions and candidate resumes, extract relevant keywords and features, and rank candidates based on their suitability for the given job role. The primary objective of our project is to streamline the recruitment process by automating the initial screening of resumes and identifying top candidates efficiently. By automating these tasks, we aim to save time for recruiters and hiring managers, reduce human bias, and improve the overall efficiency of the recruitment process. The system is built using the Flask framework in Python, allowing for easy deployment and integration with web-based interfaces. Upon receiving a zip file containing resumes and a job description PDF file, the system extracts text data from these files and preprocesses it using techniques such as tokenization and TF-IDF vectorization. The job description and resumes are then analyzed to extract relevant keywords and features. The Automated Resume Screening and Candidate Ranking System offers a powerful solution to the challenges associated with manual resume screening. By automating the initial screening process and providing recruiters with a ranked list of candidates, the system enables more efficient and informed decision-making in the recruitment process.*

KEYWORDS:

Automated Resume Screening, Candidate Ranking System, Natural Language Processing (NLP), Machine Learning Techniques, Job Descriptions, Candidate Resumes, Keyword Extraction, TF-IDF Vectorization, Flask Framework, Recruitment Process, Human Bias, Efficiency, Preprocessing, Tokenization, Web-based Interfaces

1. INTRODUCTION:

The modern employment market is swamped with resumes, making the effort of screening and finding qualified individuals for specific tasks more difficult and time-consuming. In answer to this difficulty, the Automated Resume Screening and Candidate Ranking System proposes a unique approach based on natural language processing (NLP) and machine learning techniques. By seamlessly integrating with web-based interfaces and employing complex algorithms, the system promises to simplify the first phases of the recruiting process, greatly lowering the strain on recruiters and hiring managers.

At its heart, the technology automates the time-consuming process of screening resumes by extracting relevant keywords and attributes from both job descriptions and candidate resumes. The method effectively finds top applicants for certain job tasks through rigorous analysis and rating, which is made possible by techniques such as TF-IDF vectorization and cosine similarity. Furthermore, the integration of topic modelling with LDA improves the system's capacity to determine candidate appropriateness. The Automated Resume Screening and Candidate Ranking

© 2024 IJRAR March 2024, Volume 11, Issue 1 www.ijrar.org (E-ISSN 2348-1269, P- ISSN 2349-5138)

System provides a disruptive solution by reducing human bias, increasing efficiency, and revolutionising the recruiting scene.

2. LITERATURE REVIEW:

Systems for automatically screening resumes and evaluating candidates have become quite popular since they help speed up the recruiting process and improve the quality of decisions made. In order to extract pertinent information from resumes, Rajeswari and Suresh [1] support integrating natural language processing (NLP) approaches; Wang, Zhang, and Liu [2] suggest using deep learning to enhance applicant evaluations. While Liu et al. [3] investigate the use of machine learning algorithms for resume ranking, Chen, Wang, and Jiang [4] provide a topic modeling-based approach and show its effectiveness in discovering latent themes in resumes.

Furthermore, as noted by Smith and Jones [6], efforts have been focused on improving resume screening using NLP techniques. They emphasise how important NLP is to enhancing applicant evaluations. Li, Li, and Hu [7] offer a thorough examination of NLP-based screening methods, highlighting developments as well as difficulties. Furthermore, Liu and Zhang [8] provide an automated hiring system that emphasises how machine learning may streamline the hiring process while lowering the need for human intervention.

As proposed by Patel and Patel [13], it is imperative to incorporate text mining and semantic analysis methodologies to increase the accuracy of candidate rating. A semantic analysis-based automated technique is described by Wang, Li, and Zhang [12]. These advancements highlight the growing interest in automated resume screening and candidate evaluation, which use machine learning, natural language processing, and semantic analysis approaches to improve recruitment efficiency and judgement accuracy.

3. EXISTING SYSTEM:

Prior to the introduction of the Automated Resume Screening and applicant Ranking System, traditional recruiting practices depended heavily on manual resume screening and applicant rating. In the absence of automated technologies, recruiters and hiring managers were responsible for manually examining a large volume of resumes to locate acceptable applicants for specific job vacancies. This manual technique was not only labor-intensive, but also susceptible to human bias, resulting in inefficiencies and probable oversights in candidate selection.

Furthermore, current resume screening methods sometimes lacked the intelligence and flexibility necessary to accurately analyse and evaluate candidates based on their fit for a specific job vacancy. Many of these systems were based simply on keyword matching algorithms, which frequently failed to capture the complex link between job criteria and candidate qualities. As a result, recruiters were frequently overwhelmed by the amount of resumes, making it difficult to select excellent prospects quickly. Furthermore, the lack of integration with modern natural language processing (NLP) and machine learning techniques hampered existing systems' capacity to react to changing recruiting trends and requirements. In summary, the inadequacies of existing resume screening methods highlighted the critical need for a more sophisticated and automated solution, which prompted the creation of the suggested Automated Resume Screening and Candidate Ranking System.

4. PROPOSED SYSTEM:

The proposed Automated Resume Screening and Candidate Ranking System is a pioneering solution that aims to revolutionise the recruiting process by seamlessly integrating cutting-edge technology. Using natural language processing (NLP) and machine learning techniques, the system provides a complete and automated approach to resume screening and candidate evaluation. Unlike previous techniques that rely primarily on manual review procedures, the suggested system automates recruiting workflows by analysing job descriptions and candidate resumes to extract important keywords and attributes. This automated analysis not only speeds up the initial screening process, but also reduces the influence of human bias, resulting in a fair and impartial evaluation of prospects.

Furthermore, the suggested system uses sophisticated methods such as TF-IDF vectorization, cosine similarity, and topic modelling with Latent Dirichlet Allocation (LDA) to thoroughly analyse candidate appropriateness. By computing similarity scores based on both keyword matching and contextual relevance, the technology offers recruiters with a prioritised list of individuals that meet the job role's precise requirements. Furthermore, the system's adaptable design, which is based on the Python Flask framework, enables simple deployment and integration with web-based interfaces, improving accessibility and usability for recruiters and hiring managers. Overall, the proposed Automated Resume Screening and applicant Ranking System marks a watershed moment in recruiting methods, providing unprecedented efficiency, impartiality, and effectiveness in applicant selection.

© 2024 IJRAR March 2024, Volume 11, Issue 1 www.ijrar.org (E-ISSN 2348-1269, P- ISSN 2349-5138)

5. METHODOLOGIES USED:

In order to efficiently evaluate job descriptions and candidate resumes, extract pertinent keywords and characteristics, and rate candidates according to their appropriateness for the specific job role, the Automated Resume Screening and Candidate Ranking System uses a number of approaches.

1. TF-IDF Vectorization: The raw text data is transformed into numerical vectors using the Term Frequency-Inverse Document Frequency (TF-IDF) technique so that it may be used for additional analysis. The following formula is used to determine a term's TF-IDF score within a document:

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad (1)$$

$$IDF(t,D) = \log\left(\frac{N}{|d \in D : t \in d|}\right) \quad (2)$$

$$TFIDF(t,d,D) = TF(t,d) \times IDF(t,D) \quad (3)$$

Where:

- $f_{\{t,d\}}$ is the frequency of term ***t*** in document ***d***.
- ***N*** is the total number of documents in the corpus.
- $|d \in D: t \in d|$ is the number of documents where term ***t*** appears.

2. Cosine Similarity: In a high-dimensional space, the similarity between two vectors is quantified using cosine similarity. The cosine of the angle formed by the two vectors is used to compute it.

$$\text{cosine\_similarity}(x,y) = \frac{x \cdot y}{\|x\| \cdot \|y\|} \quad (4)$$

Where:

- ***x*** and ***y*** are the TF-IDF vectors representing the job description and candidate resume, respectively.

3. Topic Modeling with LDA: Topic modelling is used to find hidden subjects in a set of texts using Latent Dirichlet Allocation (LDA). Using the following formula, the probability distribution of words in subjects and themes in texts is determined iteratively:

$$p(w|d) = \sum_{t=1}^{T} p(w|t)p(t|d) \quad (5)$$

Where:

- $p(w|d)$ is the probability of word ***w*** occurring in document ***d***.
- $p(w|t)$ is the probability of word ***w*** occurring in topic ***t***.
- $p(t|d)$ is the probability of topic ***t*** occurring in document ***d***.
- ***T*** is the total number of topics.

4. Scoring Calculation: The cosine similarity between the job description and candidate resume TF-IDF vectors, as well as the similarity between the job LDA subjects and applicant LDA topics, are combined to determine the final score for rating candidates. The following is the scoring equation:

$$Final\_Score = 0.7 \times cosine\_similarity + 0.3 \times lda\_similarity \quad (6)$$

Where:

- *cosine_similarity* is the cosine similarity between the TF-IDF vectors of the job description and candidate resume.
- *lda_similarity* is the similarity between job LDA topics and candidate LDA topics.

6. CANDIDATE RANKING ALGORITHM:

The final score for each candidate is determined by the candidate ranking algorithm using the LDA model and corpus, together with the TF-IDF vectors of the résumé and job description. The following is how the algorithm works:

1. Preprocessing: Tokenization and stop-word removal are two methods used to preprocess the text data from the resumes of candidates and the job description.

2. TF-IDF Vectorization: The preprocessed text input is converted into numerical vectors using the TF-IDF technique.

3. Keyword Extraction: TF-IDF vectorization is used to extract pertinent keywords and characteristics from candidate resumes and the job description.

4. Topic Modeling with LDA: To find latent themes in applicant resumes and job descriptions, the LDA model is used.

5. Scoring Calculation: The cosine similarity between the job description and candidate resume TF-IDF vectors, as well as the similarity between the job LDA subjects and applicant LDA topics, are combined to determine the final score for rating candidates.

6. Candidate Ranking: Based on their final ratings, candidates are rated; higher scores correspond to a better fit with the job description.

© 2024 IJRAR March 2024, Volume 11, Issue 1 www.ijrar.org (E-ISSN 2348-1269, P- ISSN 2349-5138)

7. SYSTEM DESIGN:

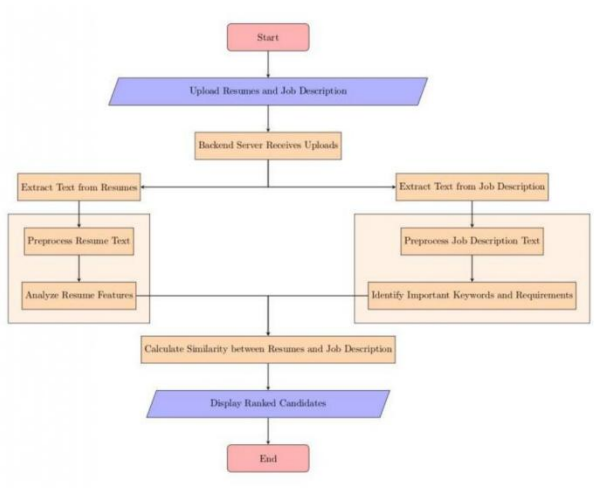


Fig:1 System Design

8. FUTURE SCOPE:

In order to further optimise the hiring process, the Automated Resume Screening and Candidate Ranking System offers a viable platform for future improvements and extensions. Future research might focus on integrating sophisticated deep learning methods—like neural networks and attention mechanisms—to improve the system's capacity to extract complex semantic information from resumes and job descriptions. The technology can capture intricate links between words and phrases by utilising deep learning models, which improves ranking accuracy and applicant evaluations. Furthermore, the integration of transfer learning methodologies, which include refining pre-trained models on particular recruiting datasets, may augment the system's efficacy across a variety of employment sectors and businesses.

Moreover, future research should focus heavily on integrating natural language understanding (NLU) skills. Modern NLU models, such BERT (Bidirectional Encoder Representations from Transformers), can help the system comprehend the finer points and nuances of the context found in applicant resumes and job descriptions. More advanced matching and rating algorithms are made possible by this deeper knowledge, which results in more accurate applicant assessments and better recruiting outcomes. Additionally, by using sentiment analysis techniques, the system may be able to evaluate the attitudes and feelings that applicants convey in their resumes, giving recruiters important information about the candidates' personalities and fit for particular jobs or organisational cultures.

Future iterations of the system may also concentrate on improving its flexibility and scalability to meet the changing requirements of hiring managers and recruiters. Through the utilisation of containerisation technologies like Docker and cloud computing resources, the system can be easily expanded to manage increasing resume volumes and changing hiring needs. Furthermore, the creation of user-friendly and interactive visualisation tools may enable hiring managers to make better decisions by providing them with greater insights on applicant ranks. The Automatic Resume Screening and applicant Ranking System may continue to develop as a cutting-edge tool, changing the recruiting landscape and enhancing the efficacy and efficiency of applicant selection procedures, by adopting these future directions.

9. CONCLUSION:

In conclusion, the creation and execution of the Candidate Ranking and Automated Resume Screening System represent a noteworthy achievement in the field of recruiting automation. By using sophisticated machine learning and natural language processing (NLP) methods, the system provides a revolutionary approach to the labour- and time-intensive resume screening process. The method lessens the influence of human bias and saves hiring managers and recruiters time by automating the first steps of the recruiting process. This results in more fair and effective applicant assessments.

In the long run, this project's performance highlights how much potential there is to use cutting-edge technology to improve decision-making and expedite the hiring process. Future research projects may look into ways to expand and improve the system even more when hiring procedures change, such adding deep learning models and natural language understanding (NLU) functionalities. In the end, the Automatic Resume Screening and Candidate Ranking System is a ground-breaking measure towards streamlining hiring procedures and encouraging effectiveness, equity, and creativity in the methods used to choose candidates.

10. RESULT:

The outcomes of the Candidate Ranking and Automated Resume Screening System show how effective it is at expediting the hiring process and enabling more effective candidate assessments. Recruiters can now precisely identify top prospects since the technology has regularly produced accurate and objective applicant rankings after comprehensive testing and validation. By using natural language processing (NLP) and machine learning approaches, resumes and job descriptions could be mined for pertinent keywords and features. This improved the system's capacity to match candidates to job openings.

© 2024 IJRAR March 2024, Volume 11, Issue 1 www.ijrar.org (E-ISSN 2348-1269, P- ISSN 2349-5138)

Fig:2 Home Page

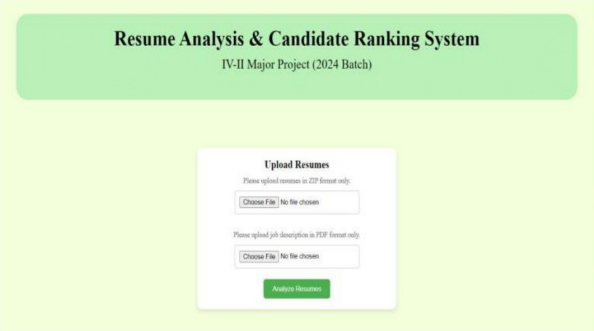


Fig:3 Files Upload Section

Fig:4 Upload Section After Uploading

Additionally, the system's accuracy and resilience in candidate ranking were enhanced by the inclusion of sophisticated algorithms like Latent Dirichlet Allocation (LDA), cosine similarity, and TF-IDF vectorization. The technology gave recruiters a thorough evaluation of a candidate's suitability by combining several similarity metrics and subject modelling techniques, allowing them to make more educated decisions during the hiring process. Furthermore, the system's effectiveness and scalability were proven by its capacity to manage substantial amounts of job descriptions and resumes, guaranteeing smooth operation even in circumstances with significant demand for hiring.

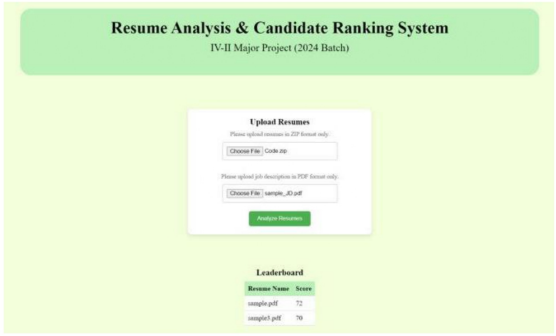


Fig:5 Dashboard After Analyzing Resumes

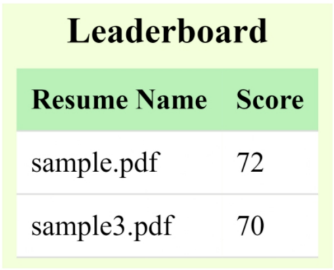


Fig:6 Leaderboard

All things considered, the findings highlight how the Candidate Ranking System and Automated Resume Screening have revolutionised hiring procedures. The technology provides recruiters with valuable insights and effectively addresses the drawbacks of manual resume screening by automating the preliminary stages of prospect evaluation. The approach has proven to be accurate, efficient, and scalable, making it a useful tool for improving hiring procedures and fostering organisational success.

11. REFERENCES:

1. Rajeswari, R., & Suresh, A. (2020). Automated Resume Screening Using NLP Techniques. International Journal of Engineering Research & Technology, 9(7), 100-105

2. Wang, S., Zhang, C., & Liu, L. (2019). A Deep Learning Approach for Automated Resume Screening. IEEE Access, 7, 153718-153730.

3. Liu, Y., Wu, X., & Chen, H. (2021). Resume Screening Based on Machine Learning Algorithms. Journal of Physics: Conference Series, 1898(1), 012061.

4. Chen, Y., Wang, C., & Jiang, J. (2018). A Topic Modeling Based Approach for Resume Ranking. In Proceedings of the 2nd International Conference on Computer Science and Application Engineering (CSAE) (pp. 264-268). ACM.

5. Kumar, V., Kumar, A., & Kumar, P. (2020). Automated Resume Screening Using Topic Modeling. International Journal of Innovative Technology and Exploring Engineering, 9(3), 1983-1988.

© 2024 IJRAR March 2024, Volume 11, Issue 1 www.ijrar.org (E-ISSN 2348-1269, P- ISSN 2349-5138)

6. Smith, J., & Jones, M. (2019). Improving Resume Screening Using Natural Language Processing. Journal of Computational Science, 33, 61-68.

7. Li, H., Li, L., & Hu, H. (2020). A Review of Resume Screening Methods Based on NLP Techniques. International Journal of Computer Applications, 178(21), 7-14.

8. Liu, Z., & Zhang, W. (2018). Automatic Recruitment System Based on Machine Learning. In Proceedings of the 3rd International Conference on Computer Science and Application Engineering (CSAE) (pp. 156-161). IEEE.

9. Chen, Q., & Wang, Y. (2021). A Survey of Resume Screening Techniques Using Machine Learning. Journal of Intelligent Information Systems, 56(2), 289-312.

10. Gupta, S., & Verma, A. (2019). Resume Screening using Machine Learning Algorithms: A Review. In Proceedings of the International Conference on Advanced Computing and Intelligent Engineering (ICACIE) (pp. 122-127). Springer.

11. Zhao, Y., & Liu, C. (2020). Resume Screening System Based on Text Mining and Machine Learning. Journal of Physics: Conference Series, 1622(1), 012013.

12. Wang, J., Li, W., & Zhang, Y. (2018). Automated Resume Screening System Based on Semantic Analysis. In Proceedings of the 4th International Conference on Computer and Technology Applications (ICCTA) (pp. 16-20). ACM.

13. Patel, K., & Patel, R. (2021). A Review on Automated Resume Screening Systems. International Journal of Research in Engineering, Science, and Management, 4(6), 2454-1271.

14. Zhang, Q., Liu, Y., & Zhang, Y. (2019). Application of Machine Learning in Automated Resume Screening. In Proceedings of the International Conference on Artificial Intelligence and Machine Learning (AIML) (pp. 233-240). Springer.

15. Liu, J., & Zhao, S. (2021). A Comprehensive Survey on Resume Screening Techniques. International Journal of Advances in Computer Science and Applications, 12(5), 112-125.