

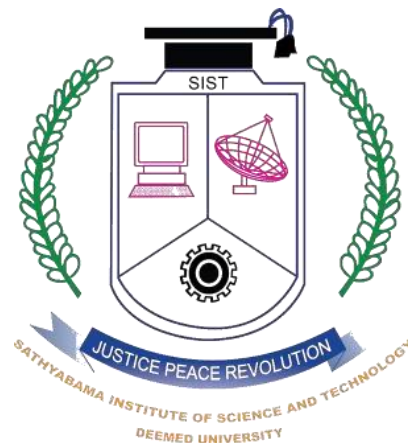
Resume screening Using NLP Embeddings,word search and AI with Streamlit WebApp

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Electronics and Communication Engineering

by

Juvvala Harshitha (Reg.No – 39130191)

Grandhi Iswarya (Reg.No – 39130174)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

**SCHOOL OF ELECTRICAL AND
ELECTRONICS**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119
APRIL – 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with —All grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Final Year Project Report is the bonafide work of **JUVVALA HARSHITHA (REG NO: 39130191)** and **GRANDHI ISWARYA (REG NO: 39130174)** who have done the project work as an individual who carried out the project entitled **“RESUME SCREENING USING NLP EMBEDDINGS, WORD SEARCH AND AI WITH STREAMLIT WEBAPP”** under my supervision from Jan 2023 to April 2023.

Internal Guide

Dr.G.RAJALAKSHMI, M.E., Ph.D.

Head of the Department

Dr. T. RAVI, M.E., Ph.D.

Submitted for Viva-voce Examination held on 26.04.23

Internal Examiner

External Examiner

DECLARATION

We **HARSHITHA.J** (39130191) and **GRANDHI ISWARYA** (REG NO: 39130174) hereby declare that the Project Report on “**RESUME SCREENING USING NLP EMBEDDINGS, WORD SEARCH AND AI WITH STREAMLIT WEBAPP**” done by us under the guidance of **Dr.G.RAJALAKSHMI, M.E., Ph.D.** at Sathyabama Institute of Science and technology is submitted in partial fulfilment of the requirements of the award of Bachelor of Engineering degree in **Electronics and Communication Engineering**.

DATE : 26.04.2023

PLACE: Chennai

1.



2.



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N.M. NANDHITHA M.E., Ph. D, Dean**, School of Electronics, **Dr. T.RAVI M.E., Ph.D.**, Head of the Department of Electronics and Communication Engineering for providing us with necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide, **Dr.G.RAJALAKSHMI,M.E.,Ph.D.** for her valuable guidance, suggestions, and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the **Department of Electronics and Communication Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The evolving technology is creating many chances of employment for many. Nowadays to apply for any job the most essential document is a resume. A resume tells a lot about the person's achievements and skill sets in all walks of life. The person applying for the job highlights the strong points and skill sets required for the company. Multinational organizations receive thousands of emails from such people who send their resumes for them to apply for a certain post. Now the real challenge is to know which resume is to be sorted and shortlisted according to the constraints. One method is to manually check and sort the resume. Now, this method is the most time-consuming and also can lead to a lot of errors because of human interventions. Also, humans cannot keep on working continuously. Hence there is a problem of less efficiency as well. The proposed system focusses on finding the find the required skill set by scanning the document or the resume and sort according to the skill sets which is a specified constraint of the organization. Machine learning for recruiting is an emerging category of HR technology, designed to reduce or even remove time-consuming activities like manual screening resumes.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ABSTRACT	V
	LIST OF TABLES	Vi
	LIST OF FIGURES	Vii
1.	INTRODUCTION	1
	1.1 INTRODUCTION	1
	1.2 DESCRIPTION	1
	1.3 OBJECTIVE	3
2.	LITERATURE SURVEY	4
3.	SYSTEM ANALYSIS	8
	3.1 EXISTING SYSTEM	8
	3.2 EXISTING SYSTEM DRAWBACKS	8
	3.3 PROPOSED SYSTEM	8
	3.4 REQUIREMENT SPECIFICATION	9
	3.4.1 HARDWARE REQUIREMENTS	9
	3.4.2 SOFTWARE REQUIREMENTS	10
	3.5 PLATFORM SPECIFICATION	10
4.	SYSTEM DESIGN	14
	4.1 SYSTEM ARCHITECTURE	14
	4.2 USE-CASE DIAGRAM	15
	4.3 ACTIVITY DIAGRAM	16
	4.4 SEQUENCE DIAGRAM	17
	4.5 DATA FLOW DIAGRAM 0	18
	4.6 DATA FLOW DIAGRAM 1	18
	4.7 ER DIAGRAM	19
5.	MODULE DESCRIPTION	20
	5.1 NATURAL LANGUAGE PROCESSING(NLP)	20
	5.2 TF-IDF	20
	5.3 LATENT DIRICHLET ALLOCATION(LDA)	21
6.	TESTING	22
	6.1 SYSTEM TESTING	22

6.2 TESTING TECHNIQUES	24
6.3 OUTPUT	26
7. CONCLUSION	28
7.1 CONCLUSION	28
7.2 FUTURE ENHANCEMENT	29
REFERENCES	30
APPENDIX	32

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
4.1	SYSTEM ARCHITECTURE	13
4.2	USE-CASE DIAGRAM	14
4.3	ACTIVITY DIAGRAM	15
4.4	SEQUENCE DIAGRAM	16
4.5	DATA FLOW DIAGRAM 0	17
4.6	DATA FLOW DIAGRAM 1	18
4.7	ER DIAGRAM	19
6.3.1	JOB DESCRIPTIONS	26
6.3.2	RANK ALIGNMENT	27
6.3.3	SCORES AND RANKS DISTRIBUTION	27

CHAPTER 1

INTRODUCTION

1.1 Introduction

Day by day there are so many new researches that are being carried out by so many organizations in many fields. Right from the IT sector to the Medical fields, there are so many researches as well as progress made in day to day life. These progress in these fields are creating new employment opportunities all over the world. Now we know that the hiring process across the world is the same. That is the candidate has to make the resume first and as most of the recruitment process is based on the candidates' resume. Now most of the organizations tell the candidates to send the resume via e-mails. Now after they receive the email the next job is to sort them according to the requirement. Now usually they sort these manually but this process is time consuming. Also, there may be so many errors due to that. The efficiency of such sorting is extremely low. This may result in giving a chance to a candidate whose credentials are not up to the mark for the organization or this may also make the manual sorting miss the candidate who is extremely good for the organization. Hence a system which is intelligent enough to sort all these documents without any error and within time is needed which is exactly our aim. Machine learning for recruiting is the application of machine learning, such as the learning or problem-solving that a computer can do, to the recruitment function. This new technology is designed to streamline or automate some part of the recruiting workflow, especially repetitive, high-volume tasks. For example, software that applies machine learning to resumes to auto-screen candidates or software that conducts sentiment analysis on job descriptions to identify potentially biased language.

1.2 Description

Recruitment is a 200-billion-dollar business. It deals with hiring the best-fit candidates having the relevant skills for a given job profile from an immensely large pool of candidates. If a company has any job opening for a position, scores of candidates mail their resumes to the company to apply for that opening. In the hiring

process, the first task for any recruiter is to screen the resumes of all the job applicants. Any company having a job opening for a particular position will have their mail inboxes bombarded with thousands of emails from aspiring job applicants every single day. Selecting the prospective candidates for that job position from a large pool of candidates for any recruiter is very tedious. It is an extremely daunting task for the recruiters of a company to manually go through thousands of resumes and select the most appropriate candidates for the job. Out of those thousands of resumes submitted to the company for the given job posting, about 75% of them do not showcase the relevant skills that are required for the job profile.

Due to this, the recruiters quite often find it really arduous to narrow down the most appropriate candidates from a large applicant pool. In recent years, there have been more than 50,000 e-recruitment sites have been developed. The developers of these online recruitment sites have used various approaches to identify the prospective candidates for a given job profile of a company. Some of these, have managed to employ classification techniques that will classify the candidate resumes into various categories for every job posting given by every company. In these approaches, every candidate's resume is tried to match with every given job posting on the recruitment site. The aim of these recruitment sites is to throw up the results to the candidate to which they are the best fit. The techniques used by these sites have resulted in high accuracy and precision, but one of the major disadvantages is the factor of time complexity. If every candidate's resume is matched with every other job posting given on the online recruitment site, the time complexity for acquiring the results is very high.

The world of Artificial Intelligence [AI] and Machine Learning [ML] has grown significantly in recent times. The availability of large amounts of data brought about by advancements in technology which has made the internet cheap and accessible to previously inaccessible regions of the world has contributed to a great increase in the performance of the ML models in recent times. Software companies around the world exploit the advances in ML to drive automation and increase their productivity in areas that relied mostly on manual human labor. The approach discussed in this project is by using machine learning to train the dataset for a particular type of job position. It is also proposed to use section-based segmentation for data extraction using Natural

language Processing (NLP). In order to improve the time efficiency of the web application, the candidate's resume will only be matched to those job openings where they are interested in and have applied to which will, in turn, reduce the time complexity. Besides, the results of the resume matching of all the candidates who have applied for the job opening will be visible only to the recruiter of that particular company. This is done with the aim to aid the recruiters of any company from the long and tedious task of viewing and analyzing thousands of candidates' resumes. In this intelligent-based approach, they will be given the option to view the candidate's resume as well as they will get the results of the best candidates suitable for the required job position. same disease, we use it as the score to rank the similarity among patients.

1.3 Objective

- The major objective of our system is to take the current resume ranking system to another level and makes it more flexible for both the entity.
 1. Candidates, who have been hired.
 2. Client company, who is hiring the candidates
- To propose an algorithm that provides a list of applicants with appropriate experience and then presents the high points of each selected resume, unlike the conventional way of applying filters and manually scanning resumes.
- To identify the most qualified candidates for a certain vacancy.
- To automate tasks that the human resources can do and recognize and identify human faces
- It proposes a resume ranking software that uses natural language processing (NLP) and machine learning.
- It could be used in MNCs where multiple resumes must be screened every single day for multiple jobs, government, and administrative offices.
- If an organization gets the right candidates in place in the company, human resources can focus their efforts on building a driven company culture, and onboarding the right human capital for the positions that come up.

CHAPTER 2

LITERATURE REVIEW

Resume Classification And Ranking Using KNN And Cosine Similarity
Rajath V; Riza Tanaz Fareed; Sharadadevi Kaganurmamath[8].

The authors in created an automated machine learning-based algorithm that recommends acceptable applicant resumes to HR based on the job description provided. The suggested methodology had two stages: first, it classified resumes into various groups. Second, it suggests resumes based on their resemblance to the job description. If an industry produces a high number of resumes, the proposed approach can be used to create an industry-specific model.

Natural Language Processing based Jaro-The Interviewing Chatbot
proposed by **Jitendra Purohit; Aditya Bagwe; Rishabh Mehta; Ojaswini Mangaonkar; Elizabeth George[5].**

The proposed system, JARO accelerates the interview process towards an unbiased decision-making process by proposing a chatbot that would conduct interviews by analyzing the candidate's Curriculum Vitae (CV), based on which, it then prepares a set of questions to be asked to the candidate. The system will consist of features like resume analysis and automatic interview processes.

Design and Development of Machine Learning based Resume Ranking System
proposed by **Tejaswini K; Umadevi V; Shashank M Kadiwal; Sanjay Revanna[9].**

The top applicants might be rated using content-based suggestion, which uses cosine similarity to find the curriculum vitae that are the most comparable to the job description supplied and KNN algorithm is used to pick and rank Curriculum Vitae (CV) based on job descriptions in huge quantities.

A Machine Learning approach for automation of Resume Recommendation system **Pradeep Kumar Roy; Sarabjeet Singh Chowdhary; Rocky Bhatia** suggested that an automated way of “**Resume Classification and Matching**” could really ease the tedious process of fair screening and shortlisting [7], it would certainly

expedite the candidate selection and decision making process. This system could work with a large number of resumes for first classifying the right categories using different classifier, once classification has been done then as per the job description, top candidates could be ranked using Content-based Recommendation, using cosine similarity and by using k-NN to identify the CVs that are nearest to the provided job description.

A Hybrid Approach to Conceptual Classification and Ranking of Resumes and Their Corresponding Job Post proposed by **Abeer Zaroor; Mohammed Maree; Muath Sabha**[1].

The research work of the authors focused on a hybrid approach that employs conceptual-based classification of resumes and job postings and automatically ranks candidate resumes (that fall under each category) to their corresponding job offers. In this context, the exploit an integrated knowledge base for carrying out the classification task and experimentally demonstrate - using a real-world recruitment dataset- achieving promising precision results compared to conventional machine learning-based resume classification approaches

An Empirical Study of Artificial Intelligence and its Impact on Human Resource Functions proposed by **Garima Bhardwaj; S. Vikram Singh; Vinay Kumar** [4].

The proposed work was conducted among 115 HR professionals at various IT sectors in Delhi/NCR region. A multiple regression method was used to test the hypothesis and confirmed a positive relationship between these two factors establishing about the increased use of AI at work results in better HR functional performance. However, AI has a significant relationship with innovativeness and also with the ease of use which reflects AI affects HR with innovations and ease of use.

Intelligent Recruitment System Using NLP proposed by **Anushka Sharma; Smiti Singhal; Dhara Ajudia** [2].

The research work in focuses on extracting data from resumes and performing the required analysis on the data to convert it into useful information for the recruiters. Thus, the Resume Parser would help the recruiters to select the best relevant

candidates in a minimal amount of time, consequently saving their time and effort.

ResumeNet: A Learning-Based Framework for Automatic Resume Quality proposed by **AssessmentYong Luo; Huaizheng Zhang; Yongjie Wang; Yonggang Wen; Xinwen Zhang** [10].

The authors developed a method for automatic RQA. Since there is also no public dataset for model training and evaluation, we build a dataset for RQA by collecting around 10K resumes, which are provided by a private resume management company. By investigating the dataset, we identify some factors or features that could be useful to discriminate good resumes from bad ones, e.g., the consistency between different parts of a resume.

Ethical Considerations in AI-Based Recruitment proposed by **Mujtaba, Dena F., and Nihar R. Mahapatra** [6].

This presents an overview of fairness definitions, methods, and tools as they relate to recruitment and establishes ethical considerations in the use of machine learning in the hiring space. Considering Deep Learning (DL) method recognize artificial Neural Network (NN) to nonlinear process, NLP tools become increasingly accurate and efficient that begin a debacle. Multi-Layer Neural Network obtaining the importance of the NLP for its capability including standard speed and resolute output. Hierarchical designs of data operate recurring processing layers to learn and with this arrangement of DL methods manage several practices

Inflectional review of deep learning on natural language processing poposed by **Fahad, SK Ahmed, and Abdul Samad Ebrahim Yahy** [3].

This resumed striving to reach a review of the tools and the necessary methodology to present a clear understanding of the association of NLP and DL for truly understanding in the training. Efficiency and execution both are improved in NLP by Part of speech tagging (POST), Morphological Analysis, Named Entity Recognition (NER), Semantic Role Labeling (SRL), Syntactic Parsing, and Coreference resolution. Artificial Neural Networks (ANN), Recurrent Neural Network (RNN), Convolution

Neural Networks (CNN), dealings among Dense Vector (DV), Windows Approach (WA), and Multitask learning (MTL) as a characteristic of Deep Learning.

CHAPTER 3 SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing system is a traditional Machine Learning based system. It gives lower rates of accuracy. It has lower efficiency. It gives lower inaccurate results.

This system may lead to loss of human potential.

3.2 EXISTING SYSTEM DRAWBACKS

- Costs can spiral
- It can be difficult to measure their effectiveness
- It's informal
- It attracts bad candidates
- There's a lot of competition
- It could lead to lost labour hours
- It attracts fraudulent applicants
- It can affect communication

3.3 PROPOSED SYSTEM

The proposed system is a resume ranking software that uses natural language processing (NLP) and machine learning. This AI-powered resume screening program goes beyond keywords to contextually screen resumes. Following resume screening, the software rates prospects in real-time depending on the recruiter's job needs.

The web application aims to order the resumes, by intelligently reading job descriptions as input and comparing the resumes which fall into the category of given Job Descriptions. It provides a ranking after filtering and recommends the better resume for a given textual job description.

In order to match and rate candidates in real-time, the software employs natural language processing. Unlike generic processes, this app utilizes Mong for string matching, Cosine Similarity, Overlapping coefficient Natural Language. Our work takes a different approach as it focuses mainly on the content of the resumes where we perform the extraction of skills and related parameters to match candidates with the job descriptions

The interactive web application will allow the job applicants to submit their resumes and apply for job postings they may still be interested in.

The resumes submitted by the candidates are then compared with the job profile requirement posted by the company recruiter by using techniques like machine learning and Natural Language Processing (NLP).

Scores can then be given to the resumes and they can be ranked from highest match to lowest match. This ranking is made visible only to the company recruiter who is interested to select the best candidates from a large pool of candidates. This is done with the aim to aid the recruiters of any company from the long and tedious task of viewing and analyzing thousands of candidate resumes. The calculated ranking scores can then be utilized to determine best-fitting candidates for that particular job opening. Since the dynamic model leverages NLP, it gives the output instantly. While going through all these pipelines, it will score each resume and give out accurate output with higher efficiency, precision, and accuracy. It works the following way -

Resumes from the Data set are parsed to remove white spaces, numbers, stop words like and, or, etc. TF-IDF vectorization is then applied to convert the words in the resumes to vectors. The text in the job description is also converted to vectors using the TF-IDF vectorizer. Cosine distance is computed to measure the similarity between the resume and the job description provided and Then Mong String algorithm is applied to identify the resumes which are closely matching with the JD provided by the recruiters. The main contributions of our work can be summarized as follows: Use of our framework is not limited to a single field of application and is useful for many more real-world applications. Providing a ranking-based approach after filtering Framework to highlight skills of a resume

3.4 REQUIREMENT SPECIFICATION

3.4.1 HARDWARE REQUIREMENTS

Processor	: Pentium Dual Core 2.00GHZ
Hard disk	: 120 GB
RAM	: 2GB (minimum)
Keyboard	: 110 keys enhanced

3.4.2 SOFTWARE REQUIREMENTS

Operating system	: Windows7 (with service pack 1), 8, 8.1 and 10
IDE	:Anaconda
Language	: Python

3.5 PLATFORM SPECIFICATION– ANACONDA

Anaconda is an open-source package manager for Python and R. It is the most popular platform among data science professionals for running Python and R implementations. There are over 300 libraries in data science, so having a robust distribution system for them is a must for any professional in this field. Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help with data collection through artificial intelligence and machine learning algorithms. With Anaconda, the process can easily set up, manage, and share Conda environments. Moreover, it can deploy any required project with a few clicks while using Anaconda. There are many advantages to using Anaconda and the following are the most prominent ones among them: Anaconda is free and open-source. In the data science sector, Anaconda is an industry staple. It is open-source too, which has made it widely popular. In order to become a data science professional, one must know how to use Anaconda for Python because every recruiter expects to have this skill. It is a must-have for data science.

It has more than 1500 Python and R data science packages, so it don't face any compatibility issues while collaborating with others. For example, suppose when a company sends a project which requires packages called A and B but the company only have package A. Without having package B, company wouldn't be able to run the project. Anaconda mitigates the chances of such errors. It can easily collaborate on projects without worrying about any compatibility issues. It gives a seamless environment that simplifies deploying projects. one can deploy any project with just a few clicks and commands while managing the rest. Anaconda has a thriving community of data scientists and machine learning professionals who use it regularly. If anyone encounter an issue, the community has already answered the same. On the other hand, one can also ask people in the community about the issues face there, it's

a very helpful community ready to help new learners. With Anaconda, one can easily create and train machine learning and deep learning models as it works well with popular tools including TensorFlow, Scikit-Learn, and Theano. One can create visualizations by using Bokeh, Holoviews, Matplotlib, and Datashader while using Anaconda.

How to Use Anaconda for Python

Now discussion has been done in all the basics in our Python Anaconda tutorial, let's discuss some fundamental commands one can use to start using this package manager.

Listing All Environments

To begin using Anaconda, one need to see how many Conda environments are present in the machine.

```
conda env list
```

It will list all the available Conda environments in the machine.

Creating a New Environment

One can create a new Conda environment by going to the required directory and use this command:

```
conda create -n <your_environment_name>
```

One can replace <your_environment_name> with the name of environment. After entering this command, conda will ask if it should want to proceed to which One should reply with y:

```
proceed ([y])/n)?
```

On the other hand, if One want to create an environment with a particular version of Python, One should use the following command:

```
conda create -n <your_environment_name> python=3.6
```

Similarly, if one want to create an environment with a particular package, one can use the following command:

```
conda create -n <your_environment_name> pack_name
```

Here, one can replace pack_name with the name of the package one want to use.

If one have a .yaml file, one can use the following command to create a new Conda environment based on that file:

```
conda env create -n <your_environment_name> -f <file_name>.yaml
```

Discussion has been done how one can export an existing Conda environment to a .yml file later in this article.

Activating an Environment

One can activate a Conda environment by using the following command:

```
conda activate <environment_name>
```

One should activate the environment before it start working on the same. Also, replace the term <environment_name> with the environment name it want to activate. On the other hand, if one want to deactivate an environment use the following command:

```
conda deactivate
```

Installing Packages in an Environment

Now that one have an activated environment, it can install packages into it by using the following command:

```
conda install <pack_name>
```

Replace the term <pack_name> with the name of the package if one want to install in the Conda environment while using this command.

Updating Packages in an Environment

If one want to update the packages present in a particular Conda environment, one should use the following command:

```
conda update
```

The above command will update all the packages present in the environment. However, if one want to update a package to a certain version, one will need to use the following command:

```
conda install <package_name>=<version>
```

Exporting an Environment Configuration

Suppose anyone want to share the project with someone else (colleague, friend, etc.). The project can be directly shared on Github, it would have many Python packages, making the transfer process very challenging. Instead of that, anyone can create an environment configuration .yml file and share it with that person. Now, it can create an environment like by using the .yml file.

For exporting the environment to the .yml file, one will first have to activate the same and run the following command:

```
conda env export ><file_name>.yml
```

if a person want to share the environment with only has to use the exported file by using the 'Creating a New Environment' command we shared before.

Removing a Package from an Environment

If anyone want to uninstall a package from a specific Conda environment, use the following command:

```
conda remove -n <env_name><package_name>
```

On the other hand, if anyone want to uninstall a package from an activated environment, one have to use the following command:

```
conda remove <package_name>
```

Deleting an Environment

Sometimes, one don't need to add a new environment but remove one. In such cases, onemust know how to delete a Conda environment, which it can do by using the following command:

```
conda env remove --name <env_name>
```

The above command would delete the Conda environment right away.

CHAPTER 4 SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

This diagram is nothing but a simple description of all the entities that have been incorporated into the system. The diagram represents the relations between each of them and involves a sequence of decision-making processes and steps which can be referred as a visual or the whole process and its implementation. All functional correspondences are explained in the fig.4.1

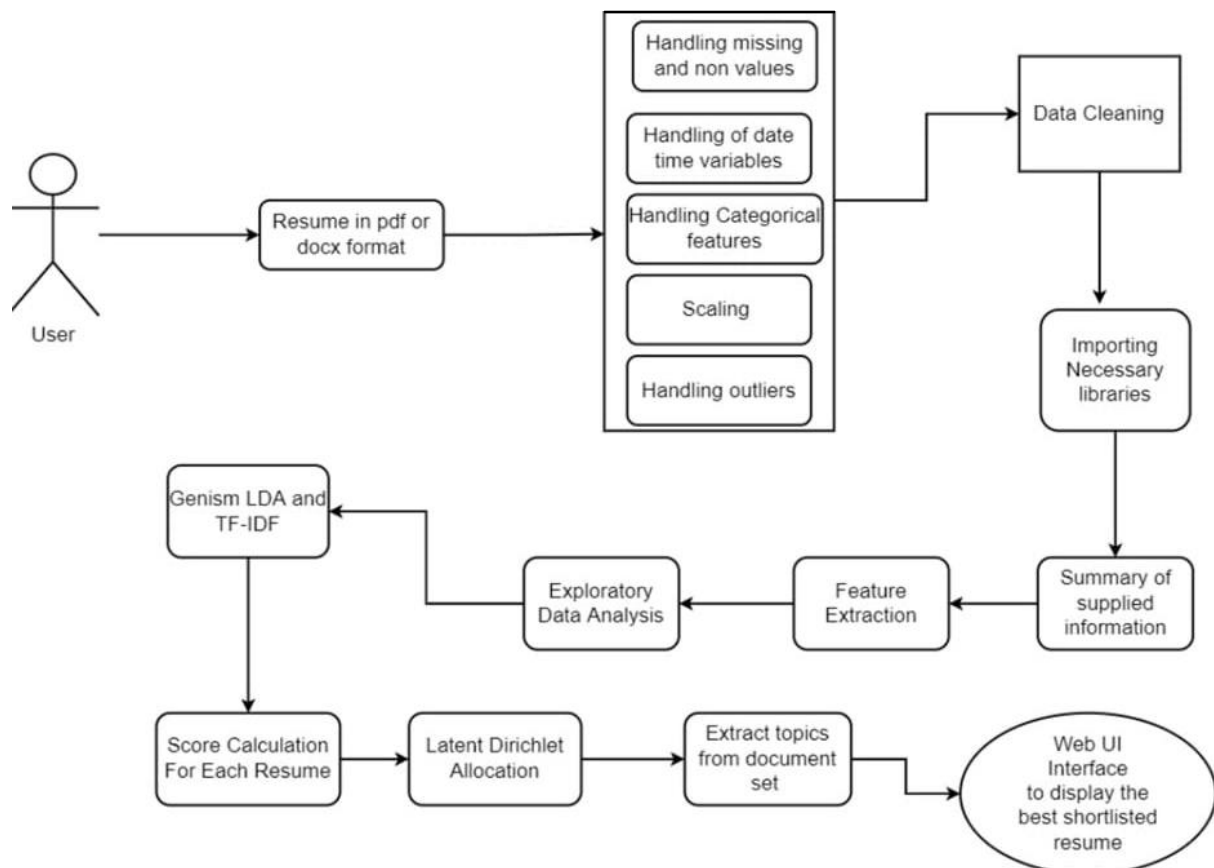


Fig:4.1: Architecture Diagram

4.2 USE-CASE DIAGRAM

A use case diagram is a standard diagram that shows all interactions between the user, dataset, and algorithm used. It is developed in the early stages of the process.

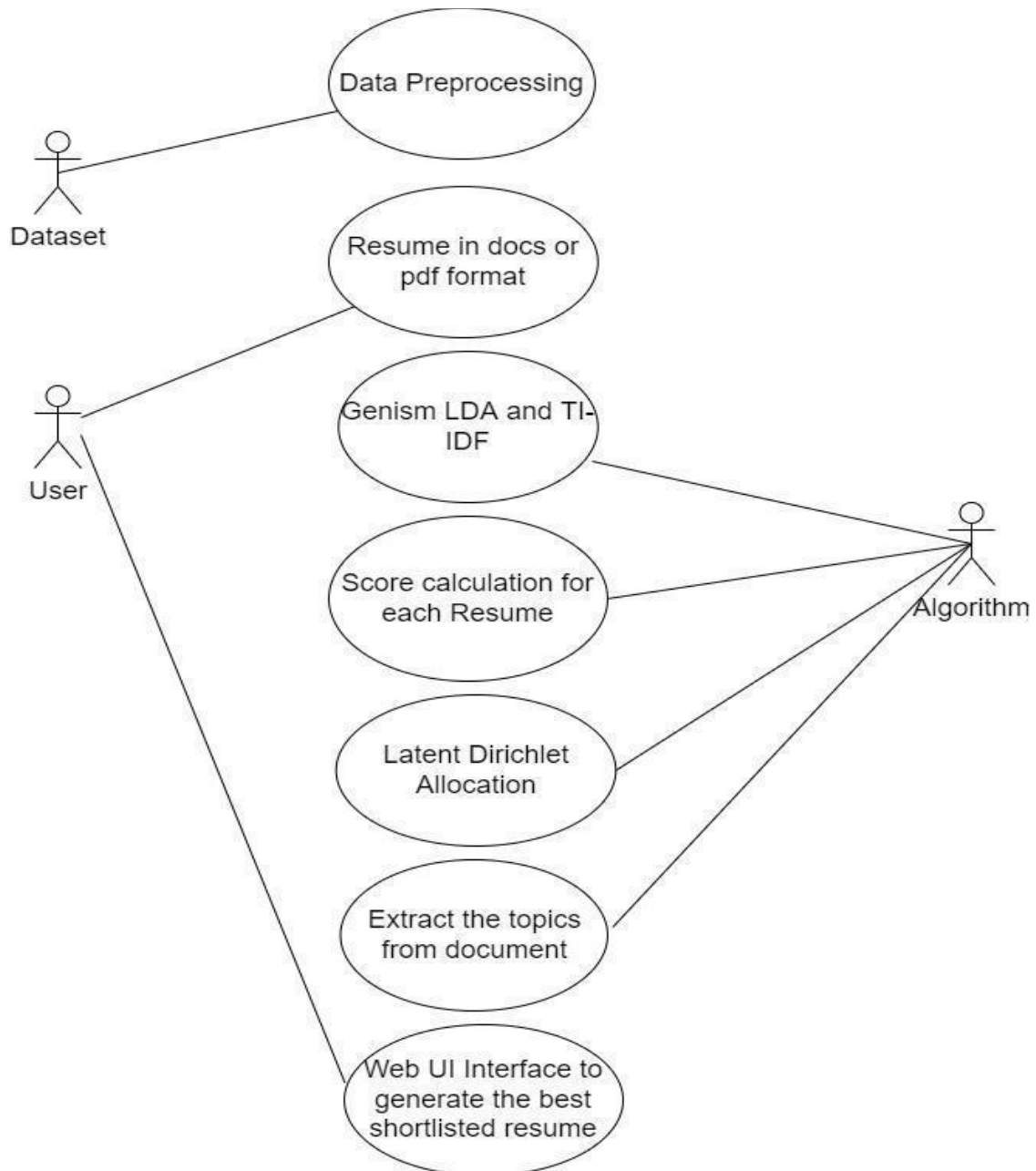


Fig: 4.2 : Use-Case Diagram

4.3 ACTIVITY DIAGRAM

In simple terms, a diagram that represents the order of all activities is called the activity diagram. It shows the workflow between different activities that take place in the whole process. However, these are not exactly flowcharts but are similar.

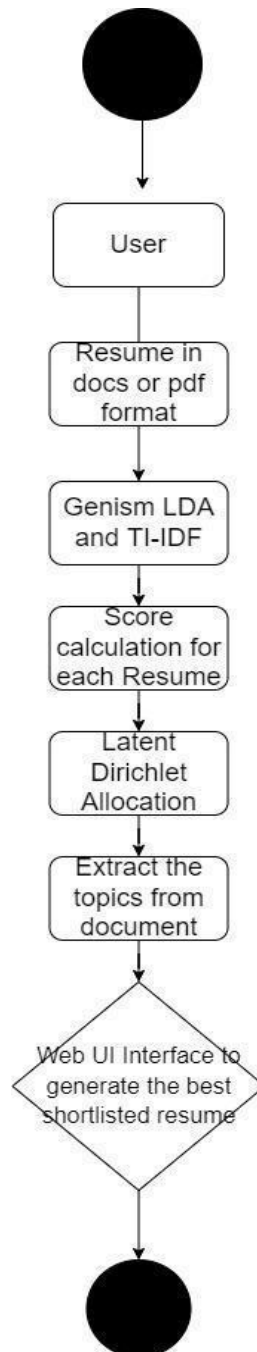


Fig: 4.3 : Activity Diagram

4.4 SEQUENCE DIAGRAM

These are other kinds of interaction-based diagrams that show how all the operations are carried out. They capture the context of collaborations between objects and processes.

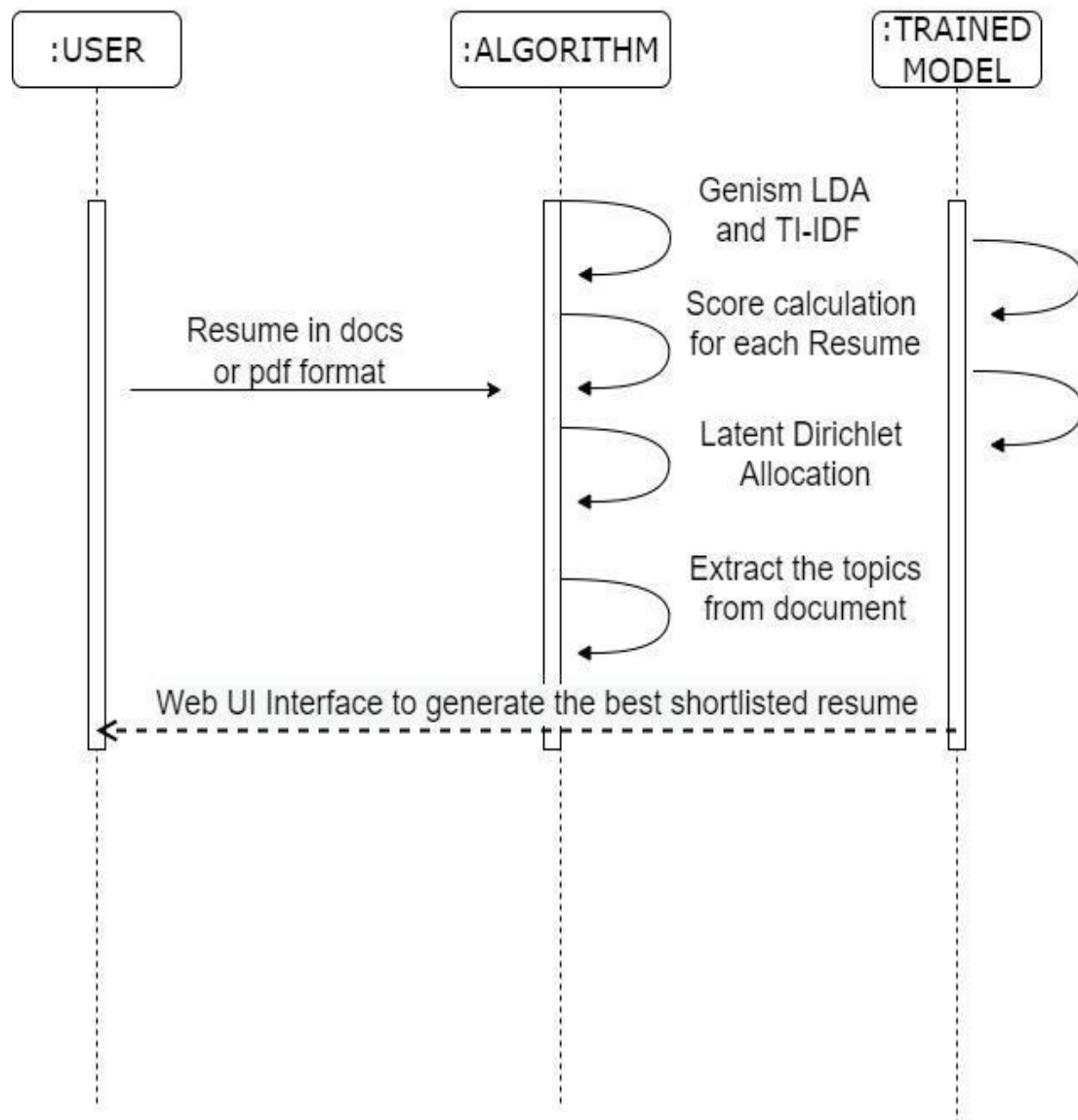


Fig: 4.4 : Sequence Diagram

4.5 DATA FLOW DIAGRAM LEVEL 0

This is basically a contextual diagram, also referred to as a “context diagram”. It only represents the top level or the 0 Level in the whole process. It gives an abstraction kind of view and shows the whole system as a single process and its relationship to externalities.



Fig: 4.5 : Data Flow Diagram 0

4.6 DATA FLOW DIAGRAM LEVEL 1

Level DFDs represent the complete system as a single process. It rotates every process and subprocess that comes together in a sequence to form the complete system. This along with) and 2-level data flow diagrams comprise the “fundamental system model”.

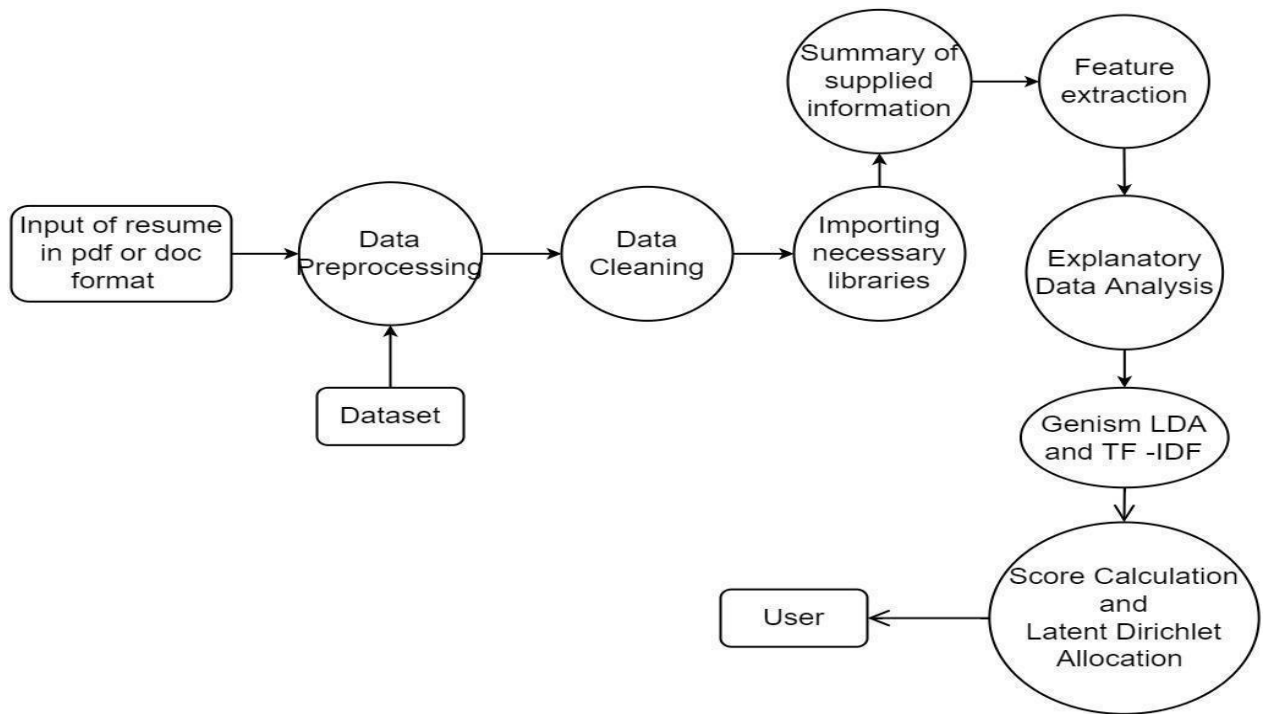


Fig: 4.6 : Data Flow Diagram 1

4.7 ER DIAGRAM

ER stands for Entity Relationship. These diagrams display the relationship of entities that are used and stored in the database. They explain the structure of the whole process. These diagrams can be made using three basic concepts, attributes, relationships, and entities.

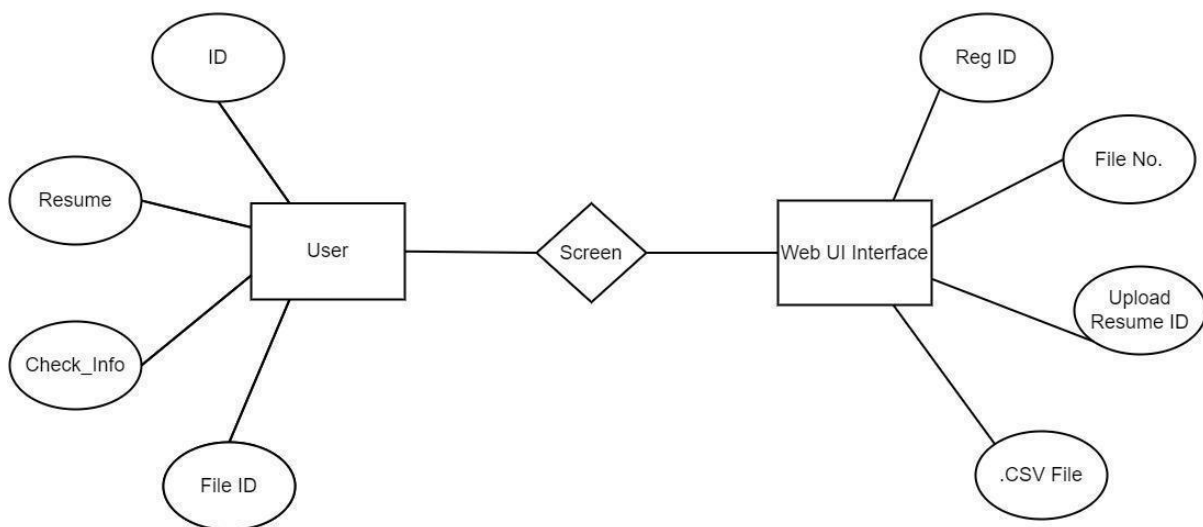


Fig: 4.7 : ER Diagram

CHAPTER 5

MODULE DESCRIPTION

5.1 NATURAL LANGUAGE PROCESSING (NLP)

Resume collection is being performed and folder Structure creation is being done. Data Cleaning such as removing clutter and unnecessary punctuation would be taken off, Feature Engineering would be performed for enhancement. This includes removing stop words, punctuation, and stemming. This process will construct a graph with sentences as the vertices. Importing necessary libraries is performed. This library creates a summary of the supplied information within the word limit. With the help of Natural language processing techniques, the application which extracts the names of people, places, and other entities from text, the main goal being to get a reduced version of it that retains the most important information. It is followed by Exploratory Data Analysis. Exploratory data analysis (EDA) is often a necessary task in uncovering hidden patterns, detecting outliers, and identifying important variables and any anomalies in data.

5.2 TF-IDF

At this stage, a dynamic Script for the TF-IDF approach is written. Term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection. TF-IDF is word frequency scores that aim to emphasize phrases that are more interesting, e.g., common in a text but not across texts, without delving into the arithmetic. The TF-IDF Vectorizer tokenizes texts, learns vocabulary, inverts frequency weightings, and allows encoding new ones. It provides information on a word frequency in the documents. Higher the TF-IDF score of a term which is computed using the above equations represents more relevance in a document.

5.3 LATENT DIRICHLET ALLOCATION (LDA)

A tool and technique for Topic Modeling, Latent Dirichlet Allocation (LDA) classifies or categorizes the text into a document and the words per topic, these are modeled based on the Dirichlet distributions and processes. Latent Dirichlet Allocation has been used in the application for the following functions such as discovering the hidden themes in the data. Classifying the data into the discovered themes. Using the classification to organize/summarize/search the documents. The application, then deals with the calculation of the score for a candidate's resume according to the job posting they have applied for. According to the score each candidate's resume receives, a rank list will be made with the candidate receiving a higher score placed higher as compared to the candidate receiving a lower score.

CHAPTER 6

TESTING

Testing Methodologies

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important testing methodologies are:

6.1 SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during this phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults in the design before coding commences, otherwise, the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as a walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

Unit Testing

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

Functional Tests

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

Performance Test

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

Stress Test

Stress Test is a test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

Structure Test

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been exercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file conditions, I/O errors, buffer problems, and textual errors in the output information

Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules that makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then test. This approach evolved from the unstructured testing of small programs. Another strategy is to construct the product in increments of tested

units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that interface dispenses can be easily found and corrected.

The major error that was faced during the project is a linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated as they complete unit testing. Testing is completed when the last module is integrated and tested

6.2 TESTING TECHNIQUES / TESTING STRATEGIES

Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet-undiscovered error. A successful test is one that uncovers an as-yet-undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hangs together. System testing requires a test consisting of several key activities and steps for running a program, string, and system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise, the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that has a probability of finding a yet undiscovered error. A successful test is one that uncovers a yet undiscovered error. Any engineering product can be tested in one of the two ways:

White-box testing

This testing is also called Glass box testing. In this testing, by knowing the specific functions that a product has been designed to perform tests can be

conducted that demonstrate each function is fully operational and at the same time search for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is white box testing.

Basis path testing:

- Flow graph notation
- Cyclomatic complexity
- Deriving test cases
- Graph matrices Control

Black box testing

In this testing by knowing the internal operation of a product, a test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black-box test case design are

- Graph-based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

Validation Testing

At the culmination of integration testing, the software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that is reasonably expected by the customer. Software validation is achieved through a series of black-box tests that demonstrate conformity with requirements. After the validation test has been conducted, one of two conditions exists.

- * The function or performance characteristics conform to specifications and are accepted.
- * A validation from the specification is uncovered and a deficiency is created.

Deviations or errors discovered at this step in this project are corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

User Acceptance Testing

User acceptance of the system is a key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective systems and users at the time of developing and making changes whenever required. This is done in regard to the following points.

- Input screen design.
- Output screen design.

6.3 OUTPUT AND RESULT

The web application returns a highly ranked candidate's resume that is most similar to the job description. The approach would aid the recruiter in expediting profile shortlisting while also ensuring the shortlisting process's authenticity, since they would be able to examine a large number of resumes in a short period of time, also with the proper fit, which a human would not be able to perform in near real-time



Fig:6.3.1 Job descriptions

PHP (Required)

JavaScript (Required)

	Rank
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10

	Name
0	Shambhai Mishra.docx
1	Bijjula Sahithi.docx

Fig:6.3.2 Rank alignment

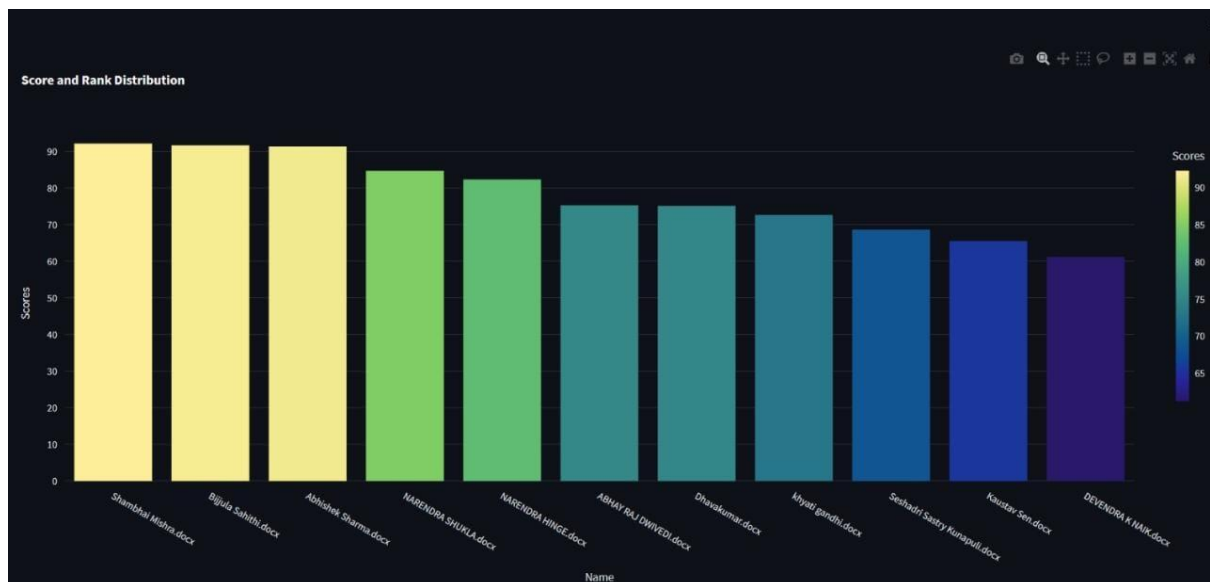


Fig:6.3.3 Scores and ranks distribution

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

Our algorithm was successfully able to screen and shortlist the best candidates with the help of NLP. Highly accurate results were obtained by using Latent Dirichlet Allocation to display the best-shortlisted resume on Web Ui. The web application was successfully able to order the resumes, by intelligently reading job descriptions as input and comparing the resumes which fall into the category of given Job Descriptions. The organization receives a large number of applications for each employment opening. Finding the right candidate's application from a sea of resumes is a time-consuming endeavor for any company these days. The classification of a candidate's resume is a laborious, time-consuming, and resource-intensive process. To address this problem, we created an automated machine learning-based algorithm that recommends acceptable applicant resumes to HR based on the job description provided. The suggested methodology had two stages: first, it classified resumes into various groups. Second, it suggests resumes based on their resemblance to the job description. If an industry produces a high number of resumes, the proposed approach can be used to create an Industry-specific model. By engaging domain experts such as HR professionals, a more accurate model may be built, and HR professionals' feedback can be used to iteratively enhance the model. The results from the model are encouraging. The resume classifier application is successful in automating the manual task of project allocation to the new recruits of the organization based on the interests, work experience, and expertise mentioned by the candidate in the profile. The Resume Screening System replaces ineffective manual screening, ensuring that no candidate is overlooked. The need for efficient and effective resume screening is at the heart of every excellent recruitment strategy. The system will be able to accept or reject a job applicant based on two factors: the company's requirements must match the skills listed in the applicant's resume, and the test evaluation will be based on the applicant's skills, ensuring that the resumes uploaded by the applicant are genuine and the applicant is truly knowledgeable about the skills. Using NLP(Natural Language

Processing) and ML(Machine Learning) to rank the resumes according to the given constraint, this intelligent system ranks the resume of any format according to the given constraints or the following requirement provided by the client company. We will basically take the bulk of input resume from the client company and that client company will also provide the requirement and the constraints according to which the resume should be ranked by our system. Besides the information provided by the resume, we are going to read the candidate's social profiles (like LinkedIn, GitHub, etc.) which will give us more genuine information about that candidate. The application automates the task of project allocation, thereby eliminating the tedious and redundant affair of opening and analyzing the resumes manually by the HR team of the organization.

7.2 FUTURE ENHANCEMENTS

The application can be extended further to other domains like Telecom, Healthcare, E-commerce, and public sector jobs. It can be put into effect and present a smart evaluation in the consistent database to survey with the present models. Efforts can be made to explore whether it is possible to identify in advance what lists might be ranked worse than the current baseline and to investigate whether there is another way to transform word embeddings into document embeddings, such as using doc2vec, that could have a greater impact on LTR results. Another area would want to focus on is ranking the resumes of people in different languages. Be it Dutch, German, French, or Hindi, we want to explore further ways to analyze and operate our model on major languages of the world.

REFERENCES

- [1] AbeerZaroor; Mohammed Maree; Muath Sabha," A Hybrid Approach to Conceptual Classification and Ranking of Resumes and Their Corresponding Job Post", Smart Innovation, Systems and Technologies book series (SIST, volume 72), 2017
- [2] Anushka Sharma; Smiti Singhal; DharaAjudia," Intelligent Recruitment System Using NLP", International Conference on Artificial Intelligence and Machine Vision (AIMV), 2021
- [3] Fahad, SK Ahmed, and Abdul Samad Ebrahim Yahya. "Inflectional review of deep learning on natural language processing." 2018 International Conference on Smart Computing and Electronic Enterprise (ICSEE). IEEE, 2018
- [4] Garima Bhardwaj; S.Vikram Singh; Vinay Kumar," An Empirical Study Artificial Intelligence and its Impact on Human Resource Functions", International Conference on Computation, Automation and Knowledge Management (ICCAKM), 2020
- [5] Jitendra Purohit; Aditya Bagwe; Rishabh Mehta; OjaswiniMangaonkar; Elizabeth George, "Natural Language Processing based Jaro-The Interviewing Chatbot", 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), August 2019
- [6] Mujtaba,Dena F., and Nihar R. Mahapatra. "Ethical Considerations in AI-Based Recruitment." 2019 IEEE International Symposium on Technology and Society (ISTAS). IEEE, 2019.
- [7] Pradeep Kumar Roy; Sarabjeet Singh Chowdhary; Rocky Bhatia," A Machine Learning approach for automation of Resume Recommendation system", Procedia Computer Science Volume 167, Pages 2318-2327, 2020

[8] Rajath V; Riza Tanaz Fareed; SharadadeviKaganurmamath," Resume Classification And Ranking Using KNN AndSimilarity",international JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY, Volume 10, Issue 08, AUGUST 2021

[9] Tejaswini K; Umadevi V; Shashank M Kadiwal; Sanjay Revanna," Design and Development of Machine Learning based Resume Ranking System", Global Transitions Proceedings, October 2021

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

APPENDIX

```
import matplotlib.colors as mcolors
import gensim
import gensim.corpora as corpora
from operator import index
from wordcloud import WordCloud
from pandas._config.config import options
import pandas as pd
import streamlit as st
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import Similar
from PIL import Image
import time
```

```
image = Image.open('Images//logo.png')
st.image(image, use_column_width=True)
```

```
st.title("Resume Screening")
```

```
# Reading the CSV files prepared by the fileReader.py
Resumes = pd.read_csv('Resume_Data.csv')
Jobs = pd.read_csv('Job_Data.csv')
```

```
##### JOB DESCRIPTION CODE
#####
# Checking for Multiple Job Descriptions
# If more than one Job Descriptions are available, it asks user to select one as well.
```

```

if len(Jobs['Name']) <= 1:
    st.write(
        "There is only 1 Job Description present. It will be used to create scores.")
else:
    st.write("There are ", len(Jobs['Name']),
        "Job Descriptions available. Please select one.")

# Asking to Print the Job Description Names
if len(Jobs['Name']) > 1:
    option_yn = st.selectbox(
        "Should we need to Show the Job Description Names?", options=['YES', 'NO'])
    if option_yn == 'YES':
        index = [a for a in range(len(Jobs['Name']))]
        fig = go.Figure(data=[go.Table(header=dict(values=["Job No.", "Job Desc.
Name"]), line_color='darkslategray',
                                fill_color='lightskyblue'),
                                cells=dict(values=[index, Jobs['Name']],
                                line_color='darkslategray',
                                fill_color='cyan'))
        ])
        fig.update_layout(width=700, height=400)
        st.write(fig)

# Asking to chose the Job Description
index = st.slider("Slide to select te JD : ", 0,
    len(Jobs['Name'])-1, 1)

option_yn = st.selectbox("Do you want Job Description to be displayed ?",
options=['YES', 'NO'])
if option_yn == 'YES':
    st.markdown("---")

```

```

st.markdown("### Job Description :")
fig = go.Figure(data=[go.Table(
    header=dict(values=["Job Description"],
        fill_color='#f0a500',
        align='center', font=dict(color='white', size=16)),
    cells=dict(values=[Jobs['Context'][index]],
        fill_color='#f4f4f4',
        align='left'))])

fig.update_layout(width=800, height=500)
st.write(fig)
st.markdown("---")

```

```

##### SCORE CALCULATION
#####

```

```

@st.cache()
def calculate_scores(resumes, job_description):
    scores = []
    for x in range(resumes.shape[0]):
        score = Similar.match(
            resumes['TF_Based'][x], job_description['TF_Based'][index])
        scores.append(score)
    return scores

```

```

Resumes['Scores'] = calculate_scores(Resumes, Jobs)

```

```

Ranked_resumes = Resumes.sort_values(
    by=['Scores'], ascending=False).reset_index(drop=True)

```

```

Ranked_resumes['Rank'] = pd.DataFrame(
    [i for i in range(1, len(Ranked_resumes['Scores'])+1)])

```

```
##### SCORE TABLE PLOT
#####
```

```
fig1 = go.Figure(data=[go.Table(
    header=dict(values=["Rank", "Name", "Scores"],
        fill_color='#00416d',
        align='center', font=dict(color='white', size=16)),
    cells=dict(values=[Ranked_resumes.Rank, Ranked_resumes.Name,
Ranked_resumes.Scores],
        fill_color='#d6e0f0',
        align='left')))]
```

```
fig1.update_layout(title="Top Ranked Resumes", width=700, height=1100)
st.write(fig1)
```

```
st.markdown("---")
```

```
fig2 = px.bar(Ranked_resumes,
    x=Ranked_resumes['Name'], y=Ranked_resumes['Scores'], color='Scores',
    color_continuous_scale='haline', title="Score and Rank Distribution")
# fig.update_layout(width=700, height=700)
st.write(fig2)
```

```
st.markdown("---")
```

```
##### TF-IDF Code
#####
```

```
@st.cache()
def get_list_of_words(document):
    Document = []
```

```

for a in document:
    raw = a.split(" ")
    Document.append(raw)

```

```

return Document

```

```

document = get_list_of_words(Resumes['Cleaned'])

```

```

id2word = corpora.Dictionary(document)

```

```

corpus = [id2word.doc2bow(text) for text in document]

```

```

lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=id2word,
num_topics=6, random_state=100,
update_every=3, chunksize=100, passes=50,
alpha='auto', per_word_topics=True)

```

```

##### LDA CODE
#####

```

```

@st.cache # Trying to improve performance by reducing the rerun computations

```

```

def format_topics_sentences(ldamodel, corpus):
    sent_topics_df = []
    for i, row_list in enumerate(ldamodel[corpus]):
        row = row_list[0] if ldamodel.per_word_topics else row_list
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df.append(
                    [i, int(topic_num), round(prop_topic, 4)*100, topic_keywords])

```

```

        else:
            break

    return sent_topics_df

#####      Topic      Word      Cloud      Code
#####
# st.sidebar.button('Hit Me')
st.markdown("## Topics and Topic Related Keywords ")
st.markdown(
    """This Wordcloud representation shows the Topic Number and the Top Keywords
that constitute a Topic.
    This further is used to cluster the resumes.    """)

cols = [color for name, color in mcolors.TABLEAU_COLORS.items()]

cloud = WordCloud(background_color='white',
                  width=2500,
                  height=1800,
                  max_words=10,
                  colormap='tab10',
                  collocations=False,
                  color_func=lambda *args, **kwargs: cols[i],
                  prefer_horizontal=1.0)

topics = lda_model.show_topics(formatted=False)

fig, axes = plt.subplots(2, 3, figsize=(10, 10), sharex=True, sharey=True)

for i, ax in enumerate(axes.flatten()):
    fig.add_subplot(ax)
    topic_words = dict(topics[i][1])
    cloud.generate_from_frequencies(topic_words, max_font_size=300)

```

```

plt.gca().imshow(cloud)
plt.gca().set_title('Topic ' + str(i), fontdict=dict(size=16))
plt.gca().axis('off')

plt.subplots_adjust(wspace=0, hspace=0)
plt.axis('off')
plt.margins(x=0, y=0)
plt.tight_layout()
st.pyplot(plt)

st.markdown("---")

##### SETTING UP THE DATAFRAME FOR SUNBURST-
GRAPH #####

df_topic_sents_keywords = format_topics_sentences(
    ldamodel=lda_model, corpus=corpus)
df_some = pd.DataFrame(df_topic_sents_keywords, columns=[
    'Document No', 'Dominant Topic', 'Topic % Contribution', 'Keywords'])
df_some['Names'] = Resumes['Name']

df = df_some

st.markdown("### Topic Modeling with Resume using LDA ")
st.markdown(
    "Using LDA to divide the topics into a number of usefull topics and creating a Cluster
    of matching topic resumes. ")
fig3 = px.sunburst(df, path=['Dominant Topic', 'Names'], values='Topic %
    Contribution',
    color='Dominant Topic', color_continuous_scale='viridis', width=800,
    height=800, title="Topic Distribution Graph")
st.write(fig3)

```

#####

RESUME

PRINTING

#####

```
option_2 = st.selectbox("Do you want to see the Best Resumes?", options=[
    'YES', 'NO'])
```

```
if option_2 == 'YES':
```

```
    indx = st.slider("Slide, Which resume to display ?:",
        1, Ranked_resumes.shape[0], 1)
```

```
    st.write("Displaying Resume with Rank: ", indx)
```

```
    st.markdown("---")
```

```
    st.markdown("## **Resume** ")
```

```
    value = Ranked_resumes.iloc[indx-1, 2]
```

```
    st.markdown("#### The Word Cloud For Resume")
```

```
    wordcloud = WordCloud(width=800, height=800,
        background_color='white',
        colormap='viridis', collocations=False,
        min_font_size=10).generate(value)
```

```
    plt.figure(figsize=(7, 7), facecolor=None)
```

```
    plt.imshow(wordcloud)
```

```
    plt.axis("off")
```

```
    plt.tight_layout(pad=0)
```

```
    st.pyplot(plt)
```

```
    st.write("With a Match Score of :", Ranked_resumes.iloc[indx-1, 6])
```

```
    fig = go.Figure(data=[go.Table(
        header=dict(values=["Resume"],
            fill_color='#f0a500',
            align='center', font=dict(color='white', size=16)),
        cells=dict(values=[str(value)],
            fill_color='#f4f4f4',
            align='left'))])
```

```

fig.update_layout(width=800, height=1200)
st.write(fig)
st.markdown("---")
import spacy
import Distill

try:
    nlp = spacy.load('en_core_web_sm')

except ImportError:
    print("Spacy's English Language Modules aren't present \n Install them by doing \n
python -m spacy download en_core_web_sm")

def _base_clean(text):
    """
    Takes in text read by the parser file and then does the text cleaning.
    """
    text = Distill.tokenize(text)
    text = Distill.remove_stopwords(text)
    text = Distill.remove_tags(text)
    text = Distill.lemmatize(text)
    return text

def _reduce_redundancy(text):
    """
    Takes in text that has been cleaned by the _base_clean and uses set to reduce the
    repeating words
    giving only a single word that is needed.
    """
    return list(set(text))

```

```
def _get_target_words(text):
```

```
    """
```

Takes in text and uses Spacy Tags on it, to extract the relevant Noun, Proper Noun words that contain words related to tech and JD.

```
    """
```

```
    target = []
```

```
    sent = " ".join(text)
```

```
    doc = nlp(sent)
```

```
    for token in doc:
```

```
        if token.tag_ in ['NN', 'NNP']:
```

```
            target.append(token.text)
```

```
    return target
```

```
def Cleaner(text):
```

```
    sentence = []
```

```
    sentence_cleaned = _base_clean(text)
```

```
    sentence.append(sentence_cleaned)
```

```
    sentence_reduced = _reduce_redundancy(sentence_cleaned)
```

```
    sentence.append(sentence_reduced)
```

```
    sentence_targetted = _get_target_words(sentence_reduced)
```

```
    sentence.append(sentence_targetted)
```

```
    return sentence
```

```
import nltk
```

```
import spacy
```

```
import re
```

```
from nltk.tokenize import word_tokenize, sent_tokenize
```

```
from nltk.corpus import stopwords
```

```
# Define english stopwords
```

```
stop_words = stopwords.words('english')
```

```

# load the spacy module and create a nlp object
# This need the spacy en module to be present on the system.
nlp = spacy.load('en_core_web_sm')
# proces to remove stopwords form a file, takes an optional_word list
# for the words that are not present in the stop words but the user wants them deleted.

```

```

def remove_stopwords(text, stopwords=stop_words, optional_params=False,
optional_words=[]):
    if optional_params:
        stopwords.append([a for a in optional_words])
    return [word for word in text if word not in stopwords]

```

```

def tokenize(text):
    # Removes any useless punctuations from the text
    text = re.sub(r'^\w\s', "", text)
    return word_tokenize(text)

```

```

def lemmatize(text):
    # the input to this function is a list
    str_text = nlp(" ".join(text))
    lemmatized_text = []
    for word in str_text:
        lemmatized_text.append(word.lemma_)
    return lemmatized_text

```

```

# internal fuction, useless right now.

```

```

def _to_string(List):
    # the input parameter must be a list

```

```

string = " "
return string.join(List)

```

```

def remove_tags(text, postags=['PROPN', 'NOUN', 'ADJ', 'VERB', 'ADV']):

```

```

    """

```

Takes in Tags which are allowed by the user and then eliminates the rest of the words

based on their Part of Speech (POS) Tags.

```

    """

```

```

    filtered = []

```

```

    str_text = nlp(" ".join(text))

```

```

    for token in str_text:

```

```

        if token.pos_ in postags:

```

```

            filtered.append(token.text)

```

```

    return filtered

```

```

from operator import index

```

```

from pandas._config.config import options

```

```

import Cleaner

```

```

import textract as tx

```

```

import pandas as pd

```

```

import os

```

```

import tf_idf

```

```

resume_dir = "Data/Resumes/"

```

```

job_desc_dir = "Data/JobDesc/"

```

```

resume_names = os.listdir(resume_dir)

```

```

job_description_names = os.listdir(job_desc_dir)

```

```

document = []

```

```

def read_resumes(list_of_resumes, resume_directory):

```

```

    placeholder = []

```

```

for res in list_of_resumes:
    temp = []
    temp.append(res)
    text = tx.process(resume_directory+res, encoding='ascii')
    text = str(text, 'utf-8')
    temp.append(text)
    placeholder.append(temp)
return placeholder

```

```

document = read_resumes(resume_names, resume_dir)

```

```

def get_cleaned_words(document):
    for i in range(len(document)):
        raw = Cleaner.Cleaner(document[i][1])
        document[i].append(" ".join(raw[0]))
        document[i].append(" ".join(raw[1]))
        document[i].append(" ".join(raw[2]))
        sentence = tf_idf.do_tfidf(document[i][3].split(" "))
        document[i].append(sentence)
    return document

```

```

Doc = get_cleaned_words(document)

```

```

Database = pd.DataFrame(document, columns=[
    "Name", "Context", "Cleaned", "Selective", "Selective_Reduced",
    "TF_Based"])

```

```

Database.to_csv("Resume_Data.csv", index=False)

```

```

def read_jobdescriptions(job_description_names, job_desc_dir):
    placeholder = []
    for tes in job_description_names:
        temp = []
        temp.append(tes)
        text = tx.process(job_desc_dir+tes, encoding='ascii')
        text = str(text, 'utf-8')
        temp.append(text)
        placeholder.append(temp)
    return placeholder

job_document = read_jobdescriptions(job_description_names, job_desc_dir)

Jd = get_cleaned_words(job_document)

jd_database = pd.DataFrame(Jd, columns=[
    "Name", "Context", "Cleaned", "Selective", "Selective_Reduced",
    "TF_Based"])

jd_database.to_csv("Job_Data.csv", index=False)

```