Empower Recruit

Submitted for partial fulfillment of the requirements

for the award of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING -

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

by

RIMMALAPUDI RAJESH - 20BQ1A4247

RANKELA SAI SRI HARSHA - 20BQ1A4245

BUSI JOSEPH - 20BQ1A4208

VARA LAKSHMAIAH DUGGINENI - 20BQ1A4217

Under the guidance of

Mr. A. JANARDHANA RAO, ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING - ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Permanently Affiliated to JNTU Kakinada, Approved by AICTE

Accredited by NAAC with 'A' Grade, ISO 9001:2008 Certified

NAMBUR (V), PEDAKAKANI (M), GUNTUR – 522 508

Tel no: 0863-2118036, url: www.vvitguntur.com

April 2024

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY
Permanently Affiliated to JNTUK, Kakinada, Approved by AICTE
Accredited by NAAC with 'A' Grade, ISO 9001:20008 Certified
Nambur, Pedakakani (M), Guntur (Gt) -522508

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING - ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

CERTIFICATE

This is to certify that the project entitled **"Empower Recruit"** is the bonafide work of Mr. R. Rajesh, Mr. R. Sai Sri Harsha, Mr. B. Joseph, and Mr. D. Vara Lakshmaiah, bearing Reg. No. **20BQ1A4247, 20BQ1A4245, 20BQ1A4208 and 20BQ1A4217** respectively who had carried out the project entitled **"Empower Recruit"** under our supervision.

Project Guide
(Mr. A. Janardhana Rao**,** Assistant Professor)

Head of the Department
(Dr. K. Suresh Babu, Professor)

Submitted for Viva-voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, Mr. R. Rajesh, Mr. R. Sai Sri Harsha, Mr. B. Joseph, and Mr. D. Vara Lakshmaiah, hereby declare that the Project Report entitled **" Empower Recruit "** done by us under the guidance of Mr. A. Janardhana Rao, Assistant Professor, Computer Science Engineering - Artificial Intelligence & Machine Learning at Vasireddy Venkatadri Institute of Technology is submitted for partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science Engineering - Artificial Intelligence & Machine Learning. The results embodied in this report have not been submitted to any other University for the award of any degree.

DATE :

PLACE :

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We take this opportunity to express my deepest gratitude and appreciation to all those people who made this project work easier with words of encouragement, motivation, discipline, and faith by offering different places to look to expand my ideas and helped me towards the successful completion of this project work.

First and foremost, we express my deep gratitude to **Shri. Vasireddy Vidya Sagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the B.Tech programme.

We express my sincere thanks to **Dr. Y. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the B.Tech programme.

We express my sincere gratitude to **Dr. K. Suresh Babu**, Professor & HOD, Computer Science Engineering - Artificial Intelligence & Machine Learning Vasireddy Venkatadri Institute of Technology for his constant encouragement, motivation and faith by offering different places to look to expand my ideas.

We would like to express my sincere gratefulness to our Guide **Mr. A. Janardhana Rao,** Assistant Professor, Computer Science Engineering - Artificial Intelligence & Machine Learning for his insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project.

We would like to express our sincere heartfelt thanks to our Project Coordinator **Mr. N. Balayesu,** Assistant professor, Computer Science Engineering - Artificial Intelligence & Machine Learning for his valuable advices, motivating suggestions, moral support, help and coordination among us in successful completion of this project.

We would like to take this opportunity to express my thanks to the **Teaching and Non-Teaching** Staff in the Department of Computer Science Engineering - Artificial Intelligence & Machine Learning, VVIT for their invaluable help and support.

Name (s) of Students

R. Rajesh. [20BQ1A4247]
R. Sai Sri Harsha. [20BQ1A4245]
B. Joseph. [20BQ1A4208]
D. Vara Lakshmaiah [20BQ1A4217]

TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE

NLP	Natural Language Processing
BERT	Bert Model
GS	Cosine Similarity

ABSTRACT

The objective of "EmpowerRecruit: Revolutionizing Campus Placements" is to address the employability challenge faced by higher educational institutions in India by providing a comprehensive master database for tracking and analyzing campus placements nationwide. The project aims to empower government policymakers with valuable information to formulate targeted solutions for enhancing employability in diverse educational domains. By analyzing placement data from various regions, sectors, and institutions, the project seeks to identify areas needing immediate attention and implement well-informed policies that bridge the gap between industry demands and graduate skillsets. Additionally, the project aims to serve as a dynamic resource for corporate entities seeking skilled candidates, streamlining recruitment processes and identifying promising talent more efficiently. Overall, "EmpowerRecruit" aims to revolutionize campus placements in India, creating a harmonious and efficient employment ecosystem that empowers graduates with the right skills for success in the evolving job market.

CHAPTER 1

INTRODUCTION

1.1 WHAT IS CAMPUS PLACEMENTS?

Campus placements refer to the process through which companies recruit students from educational institutions, typically colleges or universities, for employment opportunities. It is a structured process where organizations visit campuses to conduct recruitment drives and hire students for various job roles within their companies.

Definition:

Campus placements involve a structured process where companies recruit students from educational institutions for employment opportunities.

Process

1. Preparation: Companies notify educational institutions about their intent to conduct campus placements. Students also prepare by updating resumes, practicing interview skills, and researching participating companies.

2.Recruitment Drives: Companies visit campuses and conduct recruitment drives, which may include written tests, group discussions, and personal interviews to assess candidates' skills and suitability for different roles.

3. Selection: Based on the performance in various rounds of the recruitment process, companies select candidates for job offers. Selected candidates are typically provided with offer letters outlining the terms of employment.

4. Placement: Once students accept job offers, they are placed in the respective companies, either immediately upon graduation or after completing their academic programs.

Objectives:

- To provide employment opportunities to students.
- To fulfill the hiring needs of companies by recruiting skilled and talented individuals.
- To bridge the gap between academia and industry by facilitating direct interactions between students and employers.
- To streamline the recruitment process for both companies and educational institutions.

1.2 WHAT IS RECRUITMENT TECHNOLOGY?

Recruitment technology is the technology employers use to improve the recruitment process, from candidate screening to onboarding. It often includes smart technology like software systems and AI. The goal is to ensure recruitment is time-efficient while ensuring the best candidates get the role.

1.3 RECRUITMENT TECHNOLOGY APPLICATIONS

1.3.1 Candidate screening

Candidate screening is a process of determining whether a candidate is qualified for the role based on their education, experience, and information based on their resume. The goal of screening candidates is to decide whether to process them to the next level of hiring or to reject the application

Steps of a candidate screening process

Step 1 – Ticking off the basic requirements

Step 2 – Scanning for preferred qualifications

Step 3 – Matching the picture of the candidate to the role

1.3.2 Applicant Tracking System (ATS)

Applicant tracking software keeps track of all applicants throughout each stage of the recruitment process, ensuring better organization. An Applicant Tracking System covers all recruitment stages, from sourcing to interviewing, and tracks the candidate's progress the entire way. record several videos of animals in their natural habitat and want the information to be accurate, object detection can give you the right data.

1.3.3 AI-Powered Tools

Artificial intelligence plays a significant role in recruitment technology. AI-powered tools utilize machine learning algorithms and natural language processing to automate tasks such as resume screening, candidate matching, and chatbot interactions, reducing manual efforts and improving efficiency.

1.3.4 Video Interviewing Platforms

Recruitment technology can also manage the Interview Process. Recruitment software can schedule interviews, while you can use video tools for interviewing candidates. Video interviews are particularly effective, as they allow the employer to interview talent from all over the world.

It saves time and widens the reach.

1.3.5 Candidate Relationship Management (CRM) Systems

Recruitment CRM systems help recruiters build and maintain relationships with candidates. They enable efficient candidate engagement, personalized communication, and centralized candidate data management

1.4 NATURAL LANGUAGE PROCESSING

Natural language processing (NLP) is a subfield of Artificial Intelligence (AI). This is a widely used technology for personal assistants that are used in various business fields/areas. This technology works on the speech provided by the user breaks it down for proper understanding and processes it accordingly.

This is a very recent and effective approach due to which it has a really high demand in today's market. Natural Language Processing is an upcoming field where already many transitions such as compatibility with smart devices, and interactive talks with a human have been made possible.

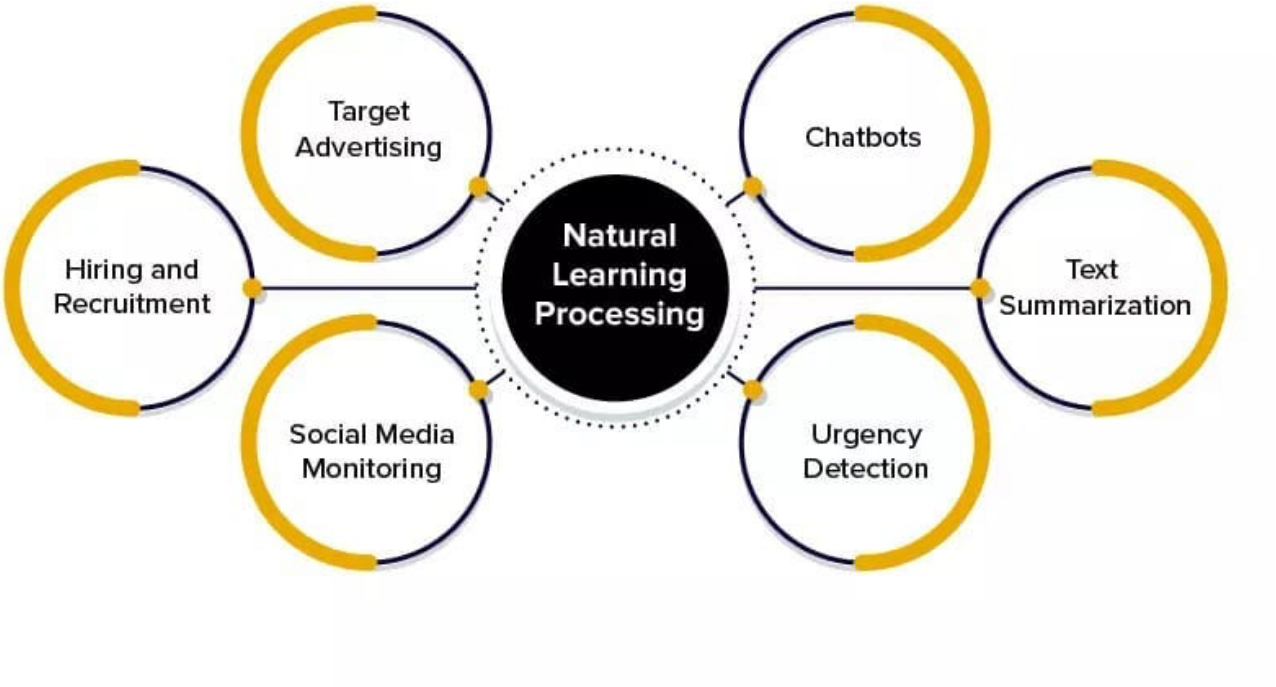


Fig: 1.1: Natural Language Processing

Knowledge representation, logical reasoning, and constraint satisfaction were the emphasis of AI applications in NLP. Here first it was applied to semantics and later to grammar. In the last decade, a significant change in NLP research has resulted in the widespread use of statistical approaches such as machine learning and data mining on a massive scale. The need for automation is never-ending courtesy of the amount of work required to be done these days. NLP is a very favorable, but aspect when it comes to automated applications. The applications of NLP have led it to be one of the most sought-after methods of implementing machine learning.

Natural Language Processing (NLP) is a field that combines computer science, linguistics, and machine learning to study how computers and humans communicate in natural language. The goal of NLP is for computers to be able to interpret and generate human language. This not only improves the efficiency of work done by humans but also helps in interacting with the machine. NLP bridges the gap of interaction between humans and electronic devices.

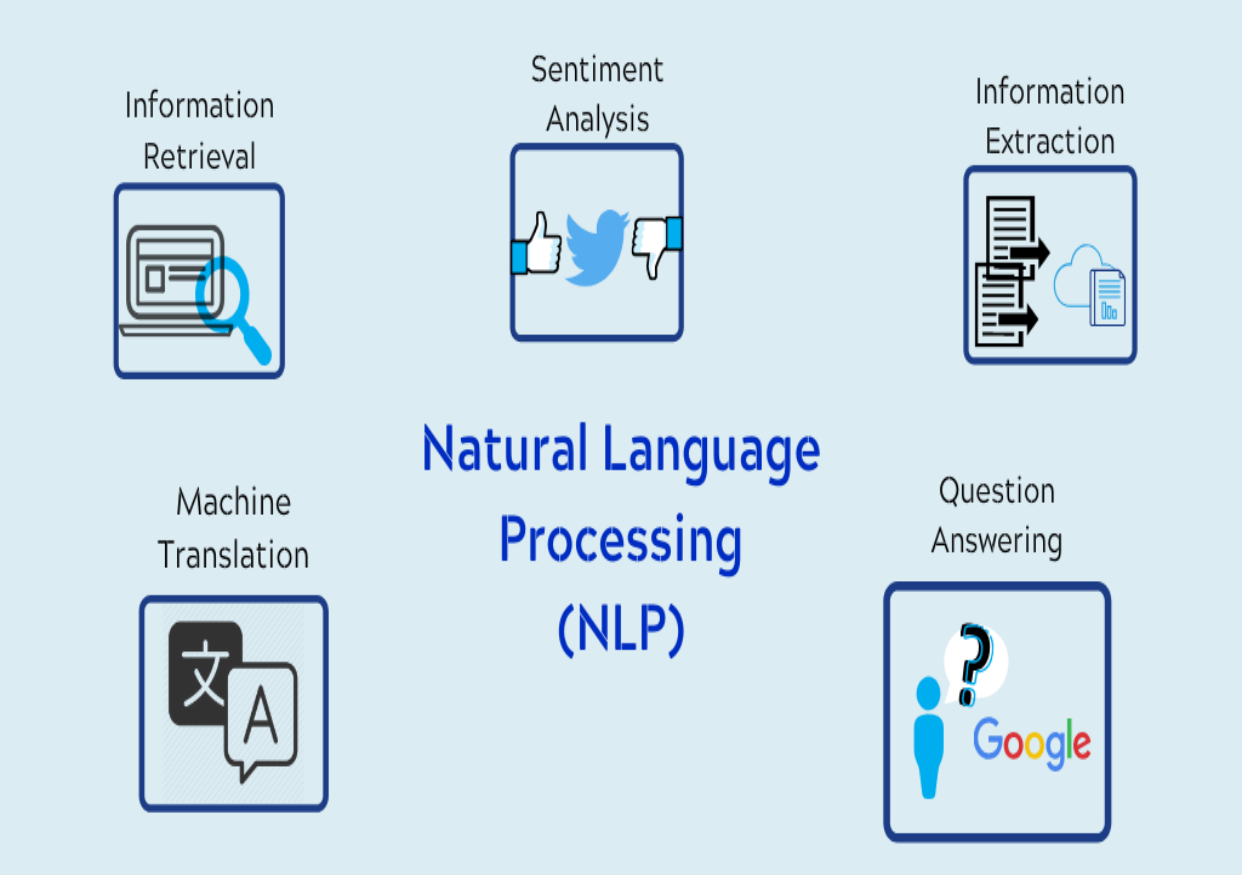


Fig: 1.2: NLP overview

Advantages

- Perform large-scale analysis
- Get a more objective and accurate analysis
- Streamline processes and reduce costs
- Improve customer satisfaction
- Better understand your market
- Empower your employees
- Gain real, actionable insights

Natural Language Processing Techniques

- **Tokenization**: the process of breaking text into individual words or phrases.
- **Part-of-speech tagging**: the process of labeling each word in a sentence with its grammatical part of speech.
- **Named entity recognition**: the process of identifying and categorizing named entities, such as people, places, and organizations, in text.
- **Sentiment analysis**: the process of determining the sentiment of a piece of text, such as whether it is positive, negative, or neutral.
- **Machine translation**: the process of automatically translating text from one language to another.
- **Text classification**: the process of categorizing text into predefined categories or topics.

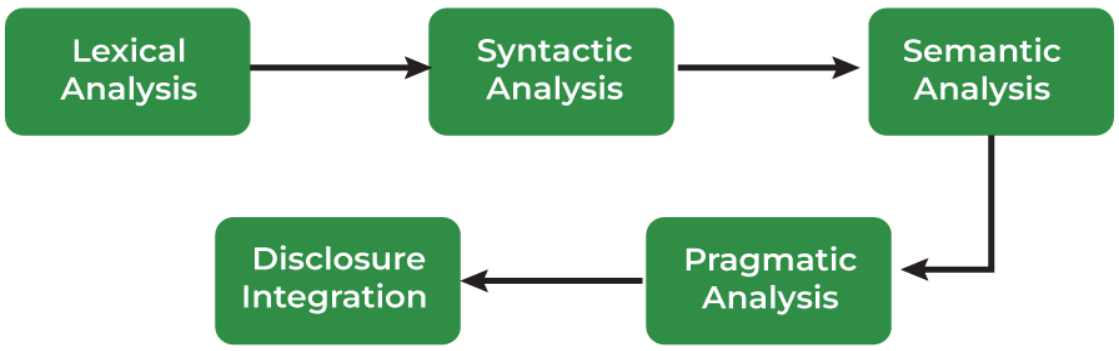


Fig: 1.3: NLP Phases

1.5 KEY COMPONENTS OF RECRUITMENT TECHNOLOGY

Campus placements refer to the process through which companies recruit students from educational institutions, typically colleges or universities, for employment opportunities. It is a structured process where organizations visit campuses to conduct recruitment drives and hire students for various job roles within their companies.

Recruitment technology is the technology employers use to improve the recruitment process, from candidate screening to onboarding.

It often includes smart technology like software systems and AI. The goal is to ensure recruitment is time-efficient while ensuring the best candidates get the role.

Advantages:

- Increased Efficiency
- Wider Reach
- Improved Candidate Experience
- Better Candidate Matching
- Data-Driven Decision Making
- Enhanced Collaboration
- Compliance and Consistency
- Scalability
- Cost Savings

1.5.1 Resume Parsing

Resume parsing is the process of extracting relevant information from a resume or CV (curriculum vitae) and converting it into a structured format that can be easily stored, searched, and analyzed by software applications

By extracting structured data from a resume, Hugging Face's Transformer model increases pattern recognition and flexibility across formats and languages. Now more and more people release their resumes through the Internet, and PDF is a wide adopted format of resume documents which contain lots of valuable information for recruitment, personal profile mining, etc.

Extracting information from academic PDF documents is crucial for numerous indexing, retrieval, and analysis tasks.Document search, recommendation, summarization, classification, knowledge base construction, question answering, and bibliometric analysis are just a examples

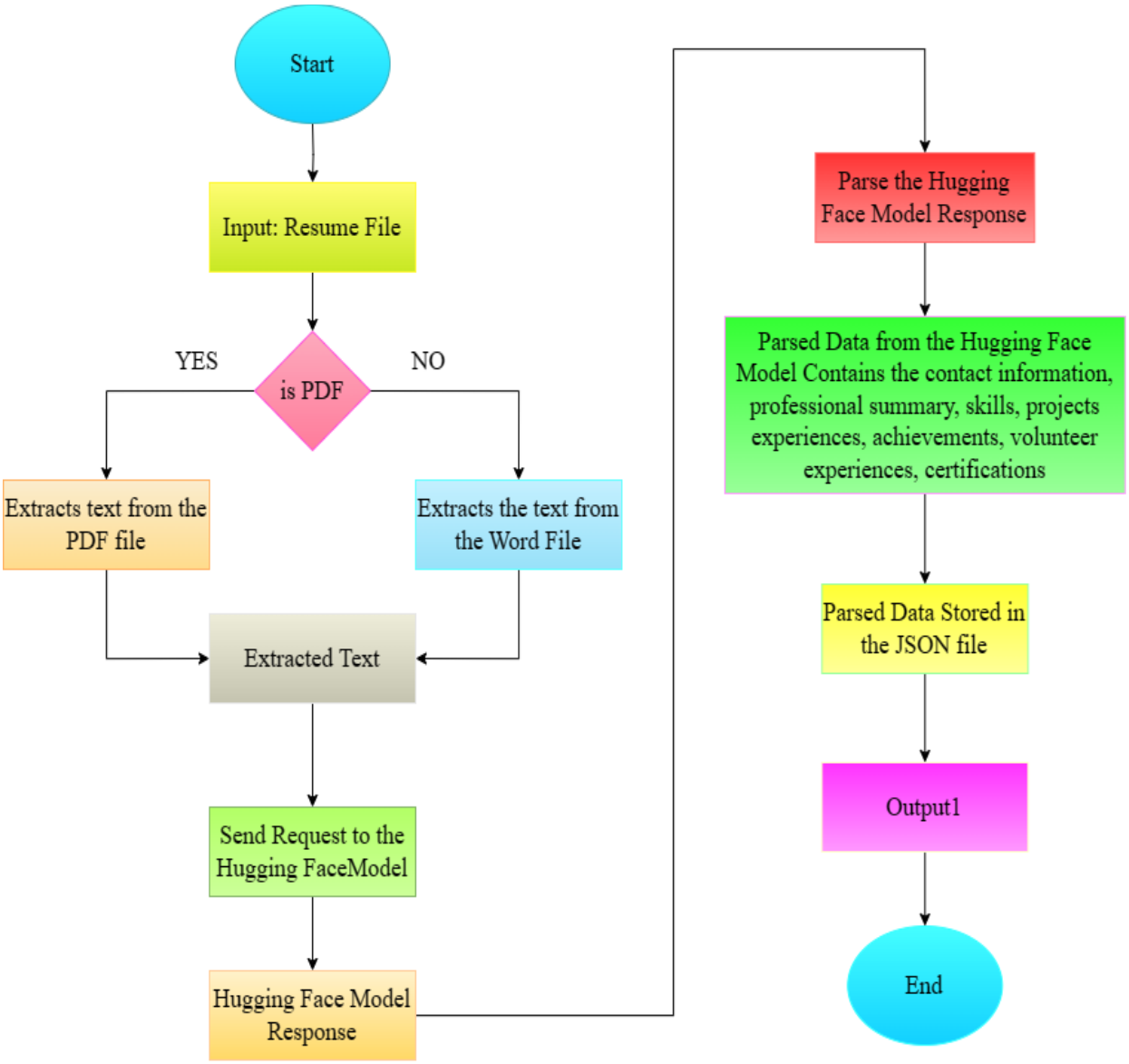


Fig: 1.4: Resume Parsing Mode

1.5.2 Job Description Analysis

Job Description Analysis is the process of examining and evaluating the content and components of a job description to gain insights into the requirements, responsibilities, and qualifications associated with a particular job role. This analysis is essential for various aspects of the recruitment process, including candidate sourcing, screening, and selection.

Tokenization and dependency parsing are two NLP techniques for extracting important requirements from job descriptions. Advanced models such as BERT improve understanding of context.

1.5.2.1BERT Language Model

BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.

BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. In the paper, the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible**.**

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. The detailed workings of Transformer are described in a paper by Google.

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

How BERT works

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. The detailed workings of Transformer are described in a paper by Google.

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once.

Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

The chart below is a high-level description of the Transformer encoder. The input is a sequence of tokens, which are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index.

When training language models, there is a challenge of defining a prediction goal. Many models predict the next word in a sequence (e.g. "The child came home from ___"), a directional approach which inherently limits context learning. To overcome this challenge, BERT uses two training strategies:

Masked LM (MLM)

Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. In technical terms, the prediction of the output words requires:

- Adding a classification layer on top of the encoder output.
- Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension.
- Calculating the probability of each word in the vocabulary with softmax.
- BERT's encoder output is fed into a classification layer, which helps in predicting the masked words. This layer fine-tunes the model's parameters to effectively capture the relationships between words and their contexts.
- The output vectors from BERT's encoder are multiplied by the embedding matrix, transforming them into the vocabulary dimension. This step enables BERT to project the contextualized representations into a space where they can be compared with the vocabulary.
- BERT leverages softmax to estimate the probability of each word being the correct prediction for the masked token, given contextual cues.
- Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension.

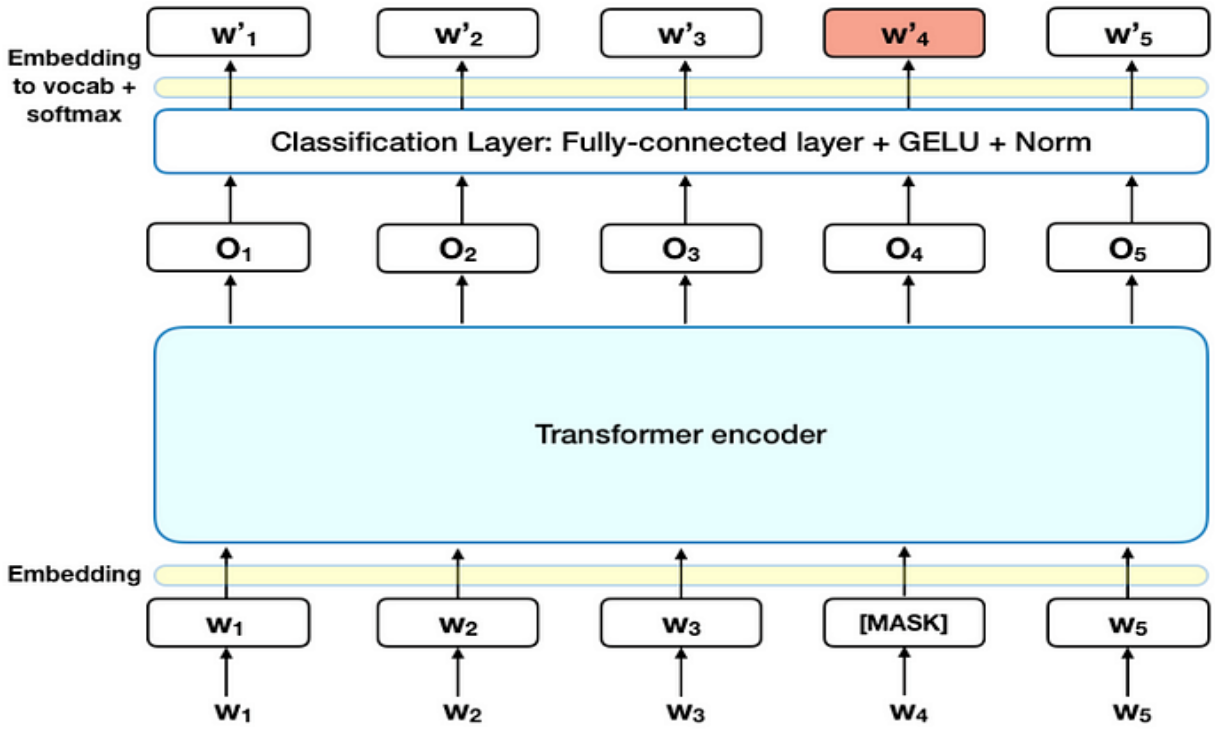


Fig: 1.5: BERT Model

1.6 AIM

The aim of the project is to modernize and optimize the campus placement process by leveraging technology, data, and user-centered design to meet the evolving needs of universities, students, administrators, and recruiters.

1.7 PROCESS

Efficient recruitment processes are essential in human resources to find the right candidates for specific job roles. This study explores using advanced natural language processing techniques to streamline recruitment. Using the Hugging Face transformer library, we focus on parsing resumes to extract crucial details. We also use NLP algorithms to extract key information from job descriptions, aiding in understanding job requirements comprehensively. One key aspect of this research is using cosine similarity to measure the similarity between candidate resumes and job descriptions. This method helps match candidates with job postings based on their skills and experiences. This approach improves the accuracy and speed of candidate screening, helping recruiters find potential matches efficiently. The study also looks at the technical implementation of these methods, emphasizing the

seamless integration of Hugging Face for resume parsing and NLP algorithms for job description extraction. It provides practical insights on steps like preprocessing, feature engineering, and parameter tuning to enhance system performance. Results show that this approach effectively automates and improves the recruitment process. By reducing manual work and biases in traditional screening, it leads to a more objective and efficient recruitment process. The study also explores potential applications of this methodology in other fields beyond recruitment that require text matching and analysis, showcasing its versatility and impact.

1.8 FEATURES

- Multi-Role Authentication and Registration System
- Administrator Oversight
- Job Listings for Recruiters
- Candidate Capability Score for Recruiters
- Job Parser for Recruiters
- Placement Management for Colleges
- Job Application Management for Students
- Resume Parsing for Students
- Application Tracking for Students
- Job Capability Score for Students
- Notifications and Alerts

1.9 EXISTING SYSTEM

The existing campus placement process, crucial for linking students with potential employers, faces challenges of being time-consuming and inefficient in today's competitive job market. To address these issues, this abstract proposes an interlinked platform for campus placement, leveraging modern technology to streamline recruitment processes and enhance student employability. This platform integrates various stakeholders, including students, employers, placement officers, and alumni, on a unified digital platform to create a seamless recruitment ecosystem.

A unique aspect of the proposed platform is the integration of alumni networks, enabling alumni to serve as mentors, provide guidance to students, and participate in recruitment activities. Leveraging alumni expertise and connections enhances student employability and adds value to the campus placement process.

1.10 STUDENT WORKFLOW

The student workflow refers to the series of steps and actions that students take when participating in various activities within an educational or professional context. In the context of campus placements or job recruitment platforms, the student workflow typically involves processes such as creating a profile, uploading resumes, searching and applying for job opportunities, managing applications, communicating with recruiters, attending interviews, and tracking the status of their job applications.

The student workflow aims to guide students through the process of exploring job opportunities, showcasing their qualifications and experiences to potential employers, and ultimately securing employment or internships that align with their career goals and aspirations.

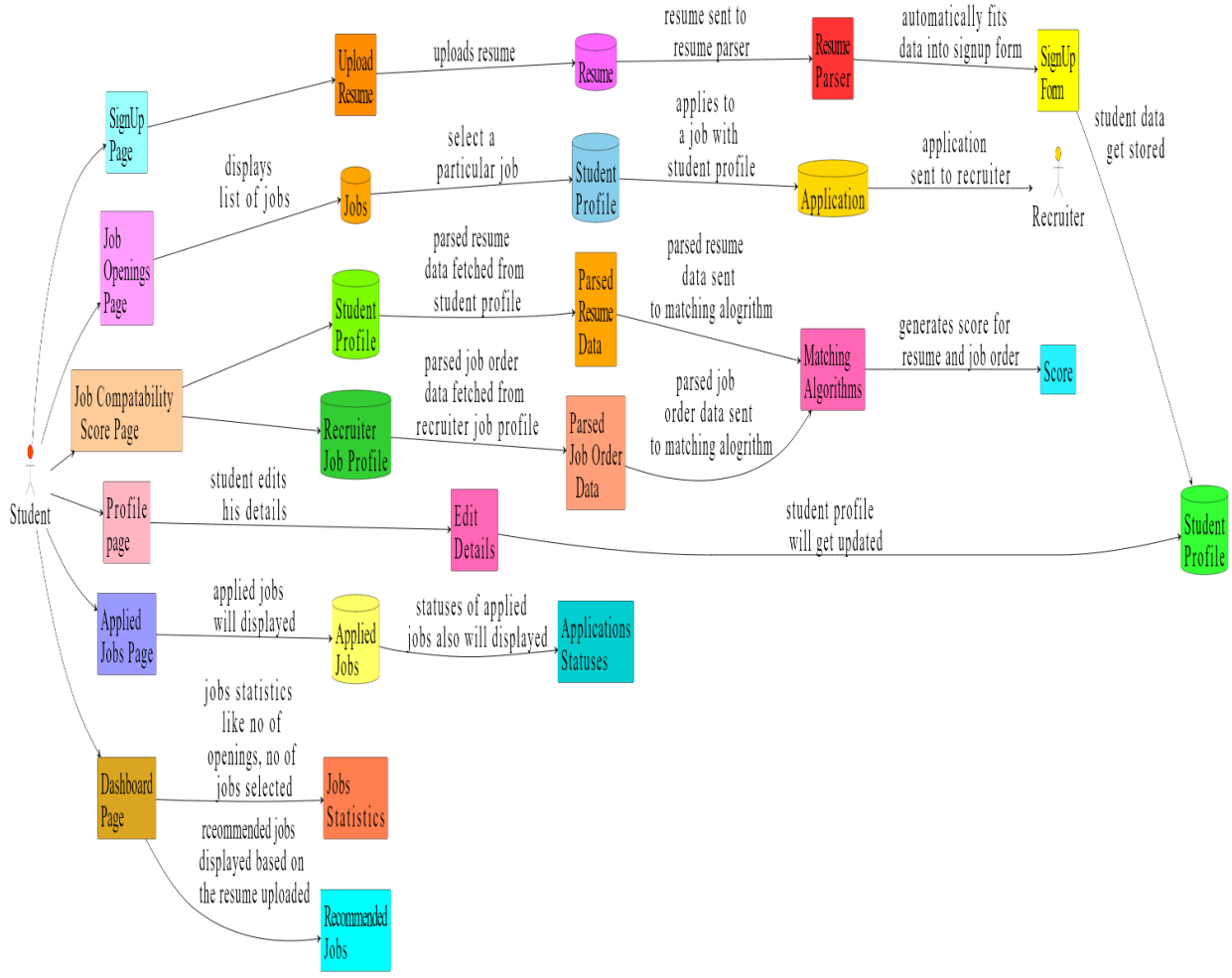


Fig: 1.6: Student Login Flow

1.11 RECRUITER WORKFLOW

The recruiter workflow refers to the sequence of activities and processes involved in the recruitment and hiring of candidates for job openings within an organization. It encompasses all the steps a recruiter takes, from identifying job requirements to sourcing candidates, screening applications, conducting interviews, and ultimately making hiring decisions. The recruiter workflow may vary depending on the organization's size, industry, and specific recruitment practices, but it typically involves tasks such as job posting, candidate sourcing, application review, candidate assessment, interview scheduling, offer negotiation, and onboarding coordination.The goal of the recruiter workflow is to efficiently and effectively fill open positions with qualified candidates who align with the organization's needs and culture. The recruiter workflow aims to streamline the hiring process and ensure a seamless transition for new hires into the organization, ultimately contributing to the company's success and growth. By optimizing each stage of the recruitment process, recruiters can identify top talent efficiently while providing a positive candidate experience, fostering long-term relationships, and enhancing employer brand reputation. This systematic approach to recruitment helps organizations attract, retain, and develop the best talent, driving innovation and competitiveness in the marketplace.

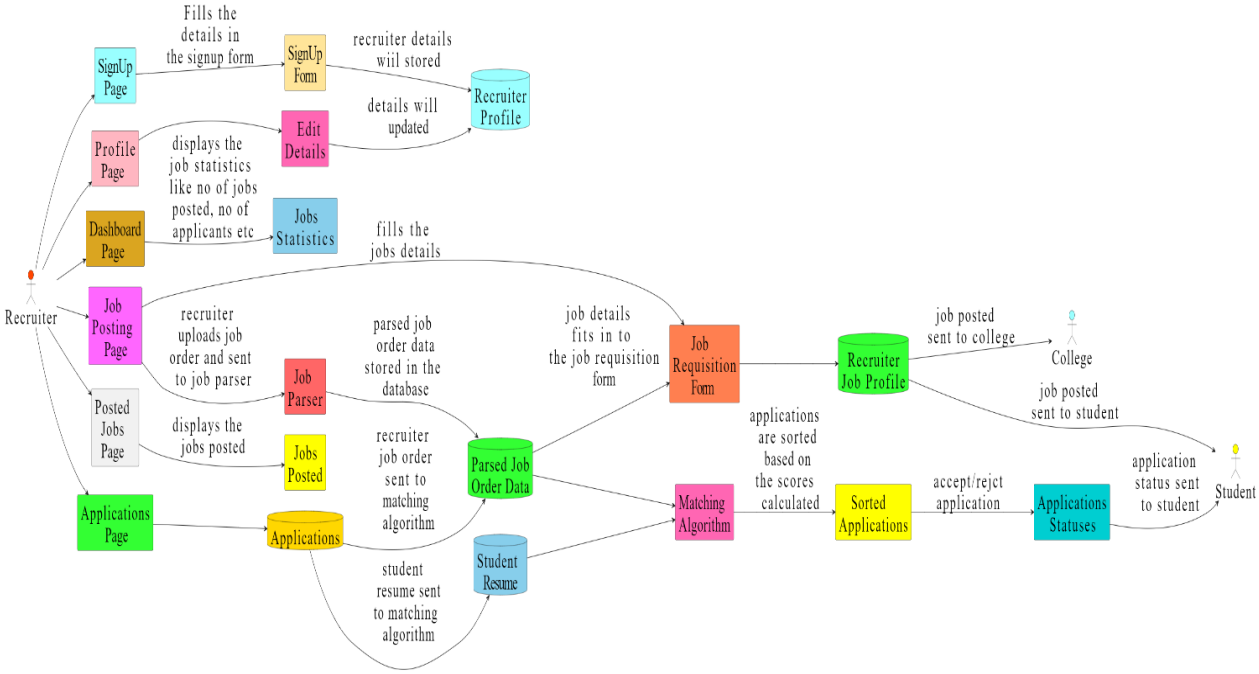


Fig: 1.7: Recruiter Login

1.12 COLLEGE FLOWCHART

The college workflow refers to the sequence of processes and activities involved in the management and operation of various functions within an educational institution, particularly in the context of placement activities and engagement with recruiters and students. It encompasses all the steps and tasks undertaken by college representatives, such as placement officers, administrators, and faculty members, to facilitate placement drives, manage student data, coordinate with recruiters, and ensure the smooth functioning of placement-related activities. The college workflow typically includes processes such as drive approval, company tie-ups, student section management, profile management (institution, TPO, principal, courses), placement statistics tracking, drive details viewing, and student details management. The goal of the college workflow is to efficiently organize and execute placement activities, enhance student employability, foster industry connections, and optimize placement outcomes for students.

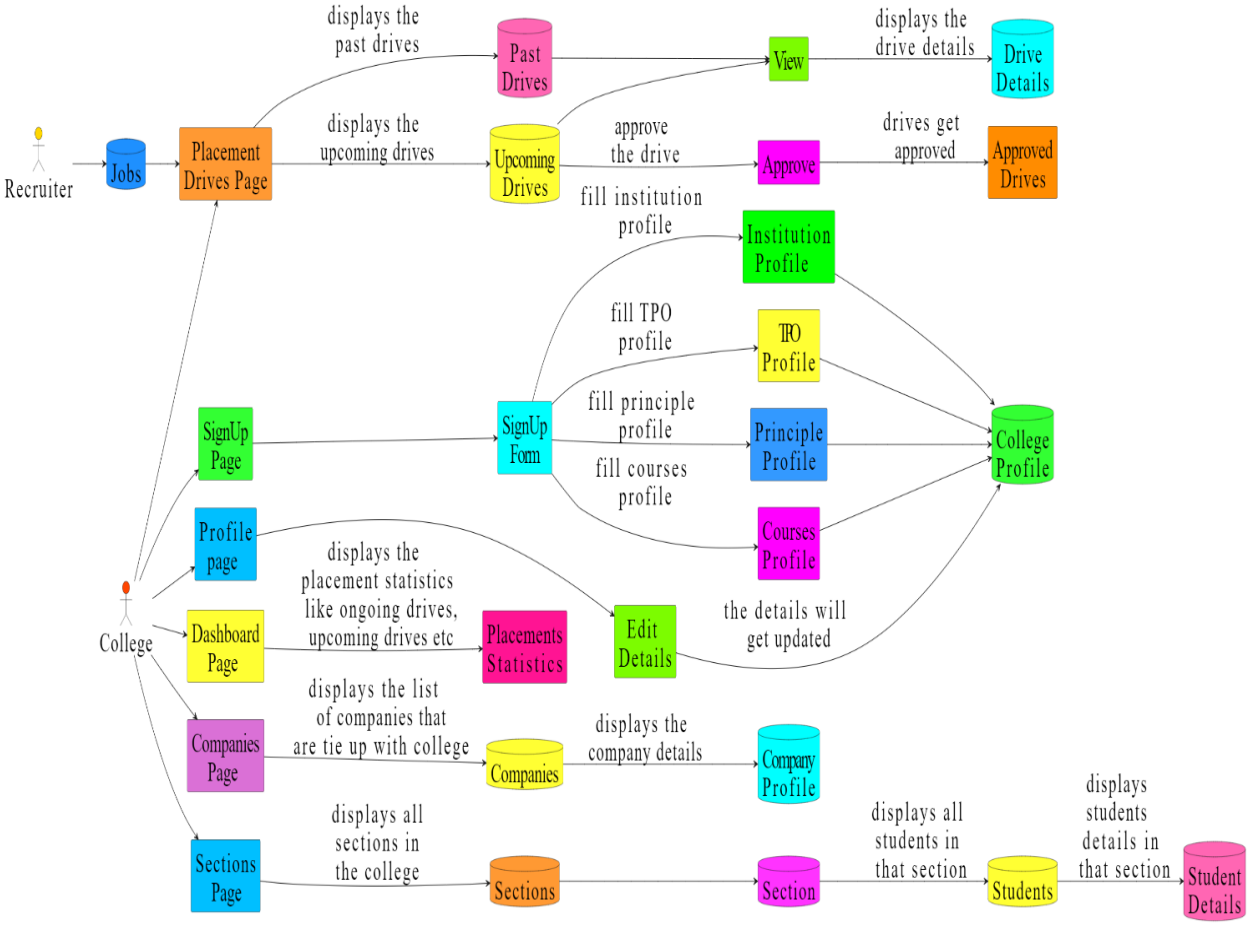


Fig: 1.8: College Login Flow

1.13 PROPOSED SYSTEM

The proposed system aims to revolutionize traditional recruitment methods by leveraging advanced NLP technologies. It addresses the shortcomings of conventional approaches and emphasizes the need for innovative solutions to adapt to changing recruitment trends.
The system utilizes the Hugging Face library for seamless extraction of relevant information from diverse resume styles. Advanced NLP techniques are employed to analyze job descriptions, identifying essential qualifications and requirements. The system employs cosine similarity to ensure candidate profiles align closely with job specifications, enhancing the matching process. The framework undergoes extensive testing with various algorithms, including Named Entity Recognition (NER), and models like BERT. Through rigorous testing, the system proves to be effective and scalable, demonstrating its potential to handle diverse recruitment scenarios.

The system successfully extracts relevant information from different resume styles, ensuring a comprehensive understanding of candidate profiles. NLP techniques accurately analyze job descriptions, identifying crucial qualifications and requirements. Utilizing cosine similarity, the system ensures that candidate profiles closely align with job specifications, improving the matching process. Future research opportunities include exploring advanced NLP algorithms to enhance the system's capabilities further.
The system can benefit from research focused on improving matching strategies to ensure better alignment between candidates and job roles. Integrating with emerging technologies can enhance the system's performance in dynamic recruitment environments, ensuring adaptability to evolving trends.

The proposed system contributes to discussions on modern recruitment practices, emphasizing the importance of technology-driven solutions for improving efficiency, accuracy, and inclusivity in talent acquisition. It underscores the need for innovative approaches to meet the challenges of dynamic recruitment environments and adapt to evolving industry trends.

- High feature extraction
- Better Performance
- High Accuracy

CHAPTER 2

REVIEW OF LITERATURE

2.1 Introduction to the Survey

2.1.1 Abundance of Applications:

Abundance of Applications Significant advancements in the field of object detection have been made over the last ten years. Drone support for Search and Rescue (SAR) operations has become increasingly common. Across the past two decades, computer vision research has focused a lot on object detection. Many publications on the detection of objects have been published as part of this vast research.

2.1.2 Challenges of Conventional Methods:

Traditional candidate selection methods, often reliant on human evaluation, are prone to biases and inefficiencies. Human evaluators may unintentionally introduce biases based on factors such as gender, race, or personal preferences, leading to unfair treatment of candidates and suboptimal hiring decisions

2.1.3 Promise of AI and NLP:

Artificial Intelligence (AI) and Natural Language Processing (NLP) technologies offer a paradigm shift in candidate selection. These technologies enable the automation of various aspects of the recruitment process, from resume parsing to candidate screening and assessment.

2.1.4 Increased Objectivity and Efficiency:

AI and NLP solutions provide opportunities for increased objectivity and efficiency in candidate selection. By relying on algorithms rather than subjective human judgment, these technologies can help reduce biases and ensure fair treatment of all candidates. Additionally, automation streamlines the recruitment process, saving time and resources for organizations.

2.1.5 Potential Benefits:

The introduction of AI and NLP in candidate selection holds the potential to revolutionize recruitment practices. With improved objectivity and efficiency, organizations can make better-informed hiring decisions, leading to higher-quality hires and increased productivity in the workforce.

2.1.6 Continued Evolution:

As AI and NLP technologies continue to evolve, so too will their impact on candidate selection techniques. Ongoing research and development in this field will further enhance the capabilities of these technologies, making them indispensable tools for modern recruitment processes.

2.2 Automated Resume Parsing

2.2.1 Process Overview:

AI-powered NLP techniques, such as the Hugging Face transformer library, utilize machine learning algorithms to analyze and understand the content of resumes. The system identifies key sections of the resume, such as contact information, education history, work experience, skills, and achievements. NLP algorithms process the textual data, extracting relevant information in a structured format that can be easily analyzed by recruiters and hiring managers.

2.2.2 Key Features:

Extraction of Contact Information: The system can accurately extract contact details like name, address, phone number, and email address from resumes.

Parsing Education and Work Experience: NLP algorithms parse through the resume to identify educational qualifications and work experience, including job titles, company names, dates of employment, and responsibilities.

Skills and Keywords Identification: The system identifies relevant skills, technologies, and keywords mentioned in the resume, helping recruiters match candidates with job requirements.

Achievements and Accomplishments Recognition: Automated parsing can also identify notable achievements, awards, certifications, and publications listed in the resume.

2.2.3 Benefits:

Time Efficiency: Automated parsing significantly reduces the time and effort required for manual resume screening. Recruiters can process a large volume of resumes in a shorter timeframe.

Accuracy and Consistency: AI-powered parsing ensures consistency in data extraction and reduces errors compared to manual methods.

Enhanced Candidate Experience: By streamlining the screening process, candidates receive quicker responses, leading to a more positive experience.

Strategic Evaluation: Recruiters can allocate more time to strategic evaluation tasks, such as assessing candidate fit, conducting interviews, and engaging with potential hires, rather than manual data entry and sorting.

2.2.4 Customization and Integration:

Automated parsing systems can be customized to match specific job requirements and organization preferences.

Integration with Applicant Tracking Systems (ATS): The parsed data can seamlessly integrate with ATS platforms, enabling recruiters to manage candidate profiles efficiently.

2.2.5 Challenges and Considerations:

Language and Format Variability: Resumes come in various formats and languages, posing challenges for accurate parsing.

Data Privacy and Security: Handling sensitive personal information requires robust data protection measures to ensure compliance with privacy regulations.

Continuous Improvement: Regular updates and refinements to the parsing algorithms are necessary to adapt to evolving resume formats and language patterns

2.3 Improved Understanding of Job Descriptions

2.3.1 Semantic Analysis:

NLP algorithms analyze job descriptions semantically, going beyond keyword matching. They understand the context, nuances, and requirements embedded in the text. This enables a deeper comprehension of the skills, qualifications, and experience sought by employers.

2.3.2 Skill Extraction:

NLP models can extract specific skills and qualifications mentioned in job descriptions. They identify key phrases related to technical skills, soft skills, certifications, and educational requirements. This skill extraction process helps recruiters better understand the job requirements.

2.3.3 Contextual Understanding:

NLP algorithms consider the context of job descriptions, including industry-specific terminology and jargon. They interpret the meaning of phrases and sentences in relation to the job role, ensuring a comprehensive understanding of the position's demands.

2.3.4 Matching Candidates:

By aligning candidate profiles with job specifications, NLP algorithms facilitate accurate matching. They compare the skills, qualifications, and experience listed in resumes with the requirements outlined in job descriptions. This ensures that candidates closely fit the desired profile.

2.3.5 Personalization:

Advanced NLP techniques personalize the understanding of job descriptions based on historical data and previous hiring patterns. They adapt to the unique preferences and priorities of each organization, enhancing the accuracy of candidate selection

2.3.6 Optimization:

NLP algorithms continually learn and optimize their understanding of job descriptions over time. They incorporate feedback from recruiters and hiring managers to improve accuracy and relevance in candidate matching.

2.3.7 Efficiency and Scalability:

Automating the understanding of job descriptions through NLP algorithms streamlines the recruitment process. It enables recruiters to handle a large volume of job postings efficiently while ensuring that candidates are accurately assessed against job specifications.

2.3.8 Reduction of Bias:

By providing an objective analysis of job descriptions, NLP algorithms help mitigate unconscious biases in the recruitment process. They focus on the essential requirements of the job, reducing the influence of subjective factors in candidate evaluation.

2.3.9 Integration with Applicant Tracking Systems (ATS):

NLP-powered job description understanding can seamlessly integrate with ATS platforms. This integration ensures consistency in candidate evaluation and simplifies the process of matching candidates with suitable job openings.

2.4 Assessing Candidate-Job Compatibility

2.4.1 Cosine Similarity-based Matching Algorithms:

Explain how cosine similarity works in the context of comparing candidate profiles and job descriptions. Provide examples of how cosine similarity is implemented in real-world recruitment systems. Discuss the advantages and limitations of cosine similarity in assessing candidate-job compatibility.

2.4.2 AI-driven Recruitment Systems and Bias Reduction:

Explore how AI-driven recruitment systems leverage objective criteria to counteract unconscious biases. Discuss specific techniques used by AI systems to mitigate bias in candidate selection processes. Provide case studies or examples of organizations that have successfully implemented AI-driven recruitment systems to reduce biases.

2.4.3 Automation and Decision-making Enhancement:

Detail the tasks automated by AI-driven recruitment systems and how this automation enhances recruiters' decision-making capabilities. Discuss the role of AI in engaging with candidates and improving the overall recruitment experience. Provide insights into how AI automation streamlines recruitment processes, leading to time and cost savings for organizations.

2.4.5 Potential Benefits of Integrating AI and NLP:

Highlight the potential benefits organizations can expect from integrating AI and NLP into recruitment practices, including improved efficiency, impartiality, and diversity.

Discuss case studies or research findings that demonstrate the tangible benefits of using AI and NLP in recruitment. Explore potential challenges or barriers organizations may face when implementing AI and NLP in recruitment and strategies to overcome them.

2.4.6 Technical Implementations and Experimental Findings:

Provide an overview of the latest technical implementations and experimental findings in AI-driven recruitment, including advancements in Hugging Face transformer models. Discuss emerging trends and future directions in AI and NLP research for recruitment purposes.

Highlight areas where further exploration and experimentation are needed to fully realize the potential of AI in recruitment. By expanding on these aspects, we can provide a comprehensive understanding of assessing candidate-job compatibility and the broader implications of AI and NLP in recruitment processes

2.5 Comparative Studies

2.5.1Lund's Study:

Lund's study introduces a unique collaboration matching approach using cosine similarity. This method not only improves resume parsing and job matching algorithms but also enhances the overall understanding of candidate-job compatibility. By quantifying similarities between candidate profiles and job descriptions, Lund's approach offers a more accurate assessment of candidate suitability, thus improving recruitment outcomes.

2.5.2 Research on Resume Ranking using BERT Embeddings:

BERT embeddings, an advanced NLP technique, demonstrate the potential for enhancing recruitment automation. By utilizing BERT embeddings, organizations can achieve a deeper understanding of textual data, leading to improved accuracy in candidate-job matching. This research underscores the importance of leveraging advanced NLP models like BERT to streamline hiring processes and enhance candidate evaluation .

2.5.3 Kadri's Focus on Job Description Mapping:

Kadri's research emphasizes the significance of accurately defining job roles and requirements for effective candidate-job matching. Proper job description mapping ensures that candidate skills align closely with job specifications, leading to successful hiring outcomes. Kadri's work highlights the importance of aligning candidate attributes with job requirements to ensure a good fit .

2.5.4 Mittal et al. and Sinha et al.'s Research on Machine Learning Techniques:

Mittal et al. and Sinha et al.'s research contributes to the development of machine learning techniques in recruitment. While their approaches may not be as advanced as transformer-based models, they offer valuable insights into using ML algorithms for tasks such as resume parsing

and job domain prediction. Understanding the strengths and limitations of these methodologies is crucial for optimizing recruitment processes and leveraging machine learning effectively .

These studies collectively enhance understanding of challenges such as bias detection and mitigation in AI-driven recruitment systems. By focusing on innovative approaches, advanced NLP techniques, and machine learning methods, they provide valuable insights for organizations aiming to optimize their recruitment processes and ensure ethical and effective candidate selection.

They underscore the importance of transparency, fairness, and accountability in algorithmic decision-making, particularly in sensitive areas like recruitment. By addressing issues of bias detection and mitigation head-on, organizations can foster more inclusive and equitable hiring practices while leveraging the potential benefits of AI-driven recruitment systems. Overall, the research in this area serves as a foundation for ongoing efforts to harness AI's capabilities responsibly and ethically in the realm of talent acquisition.

CHAPTER 3

PROPOSED SOLUTION

3.1 OVERVIEW

Efficient recruitment processes are essential in human resources to find the right candidates for specific job roles. This study explores using advanced natural language processing techniques to streamline recruitment. Using the Hugging Face transformer library, we focus on parsing resumes to extract crucial details. We also use NLP algorithms to extract key information from job descriptions, aiding in understanding job requirements comprehensively. One key aspect of this research is using cosine similarity to measure the similarity between candidate resumes and job descriptions. This method helps match candidates with job postings based on their skills and experiences. This approach improves the accuracy and speed of candidate screening, helping recruiters find potential matches efficiently. The study also looks at the technical implementation of these methods, emphasizing the seamless integration of Hugging Face for resume parsing and NLP algorithms for job description extraction. It provides practical insights on steps like preprocessing, feature engineering, and parameter tuning to enhance system performance. Results show that this approach effectively automates and improves the recruitment process. By reducing manual work and biases in traditional screening, it leads to a more objective and efficient recruitment process. The study also explores potential applications of this methodology in other fields beyond recruitment that require text matching and analysis, showcasing its versatility and impact.

3.2 DESCRIPTION

In the modern recruitment landscape, organizations struggle to efficiently choose suitable candidates among a large number of applicants. The traditional approach of manually reviewing resumes and job descriptions is time-consuming, error-prone, and can be influenced by human biases. However, the advancement of artificial intelligence (AI) and natural language processing (NLP) offers a chance to revolutionize this process. This study delves into how AI, particularly NLP techniques, can improve different stages of recruitment. Key to this effort is the use of advanced NLP models, like the Hugging Face transformer library, to automate tasks like resume

parsing and job description understanding. Resume parsing involves extracting pertinent details from candidate resumes, while job description understanding involves grasping the key requirements and qualifications listed in job postings. By utilizing AI-powered NLP algorithms, we seek to improve the efficiency and accuracy of candidate screening and matching. By utilizing techniques like cosine similarity to similarities, measure text document we can effectively evaluate candidate-job compatibility. This streamline hiring processes and ensures a fair assessment based on qualifications and skills. AI recruitment systems can reduce biases in traditional methods, promoting workforce diversity and inclusivity. Automation allows recruiters to focus on decision-making and candidate engagement, improving overall recruitment experiences. Our research highlights the importance of using AI in recruitment for increased efficiency, fairness, and diversity. Future sections will detail technical implementation, share experimental results, and explore broader implications.

3.3 PROCESS

The methodology outlines the use of technology to automate recruitment through resume parsing, job description analysis, and candidate-job matching. The integration of Hugging Face for resume parsing and NLP for job description analysis is central to this research. The first step involves extracting information from resumes using Hugging Face, a transformer library. This involves fine-tuning a pre-trained model on a dataset with various resume formats to convert unstructured data into structured information, such as skills and work experience. Hugging Face's transformer models categorize resume sections for a comprehensive understanding of candidates' qualifications. Transformer-based models enable accurate extraction of skills and qualifications from resumes. The refined model improves the parsing process by recognizing patterns and linguistic nuances in resumes, enhancing adaptability across different formats and languages. Concurrently, NLP techniques are used to analyse job descriptions and extract key requirements and qualifications. The NLP pipeline includes tokenization, part-of-speech tagging, named entity recognition, and dependency parsing to identify essential elements in job descriptions. Advanced NLP models like BERT are also utilized in this process. The NLP pipeline involves multiple stages like tokenization, POS tagging, named entity recognition, and dependency parsing. These steps help extract crucial details from job descriptions such as skills, qualifications, experience, and responsibilities. Advanced models like BERT are used to capture

context and relationships in job postings. Fine-tuning these models on job datasets helps adapt to domain-specific language, extraction accuracy. To improving match candidate skills with job requirements, cosine similarity is used to measure the similarity between profiles and postings. This metric calculates the cosine of the angle between vectors representing skills and requirements in a semantic space. Each dimension in this space corresponds to a specific skill or requirement in the context of job matching. The system uses cosine similarity between vectors to determine how similar candidate qualifications are to job requirements. High scores indicate strong alignment, while lower scores suggest mismatches. This method allows recruiters to prioritize candidates based on job relevance. Transformers and NLP algorithms ensure accurate extraction of qualifications and for efficient candidate-job requirements matching. Using advanced models like BERT helps capture context and relationships in job postings by considering both preceding and succeeding words. Fine-tuning on job description data improves extraction accuracy by adapting to domain-specific language, leveraging BERT embeddings for a deeper understanding of job requirements. Cosine similarity is used to measure similarity between candidate profiles and job postings by calculating the cosine of the angle between vectors in representing a semantic space. Vectors candidate skills and job requirements allow for quantitative assessment of alignment in the context of job matching. Utilizing cosine similarity, the system assesses the level of alignment between candidate skills and job criteria. Higher scores indicate a strong match, while lower scores point to discrepancies. Prioritizing candidates with higher scores aids recruiters in focusing on the most suitable candidates for the position. The technology combines Hugging Face, NLP, and cosine similarity for automating recruitment processes, ensuring accurate extraction of qualifications and enhancing the objectivity of candidate screening.

3.3.1. Resume parsing using HuggingFace

In this section, we explain how resumes are parsed using the Hugging Face library. Resumes in PDF or DOCX format are first converted to raw data. A class called ResumeParser is created to help with parsing. The methodology includes importing modules and classes like trace malloc, contour, and parcv package components (Models). The ResumeParser class is initialized with pre trained models for NER, NER dates, zero-shot classification, a tagger, and a QA model, along with a dictionary to store resume information. Various methods are defined for parsing different resume segments using techniques like regex, named entity recognition, and zero shot classification.

Utility methods aid in tasks such as finding names and email addresses, while helper functions assist in parsing tasks. The parse method acts as the primary entry point for parsing a resume by iterating over segments and delegating tasks based on segment type. Overall, the Resume Parser class uses ML models and NLP techniques to automate the extraction of key information from resumes, including contact details, education, skills, and job history

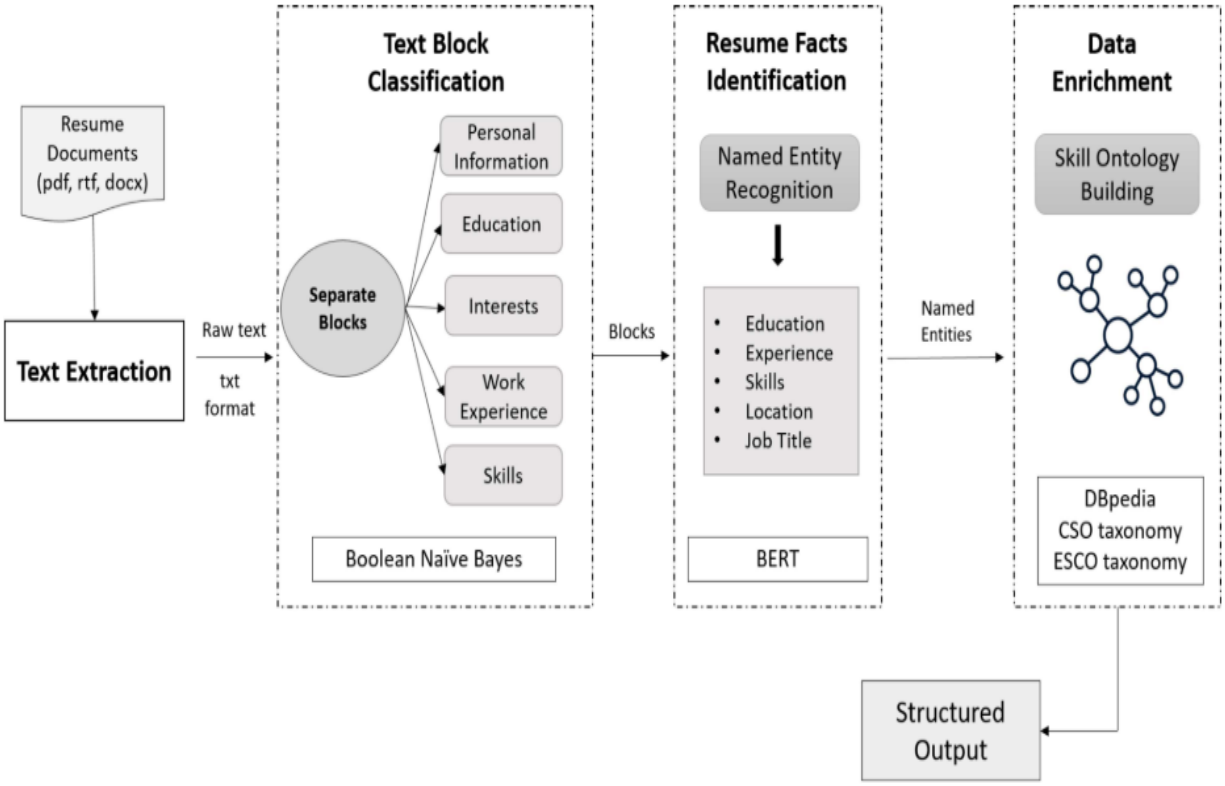


Fig: 3.1: Resume Parsing Algorithm

3.3.2 Resume Segmentation

This section elaborates on the methodology for resume segmentation through zero-shot classification. The process involves using the ResumeSegmenter class to divide resumes into distinct sections. The methodology begins with initializing the class using a zero-shot classifier for text categorization. Common section headers, such as objectives, work history, education, skills, accomplishments, and miscellaneous, are defined as tuples. The segmentation process includes methods like find_segment_indices to identify section starting points and slice_segments to create subsections within find_true_segment sections. method attempts The to determine the accurate segment within each section based on content and a list of possible segment classes.

The main segmentation method, segment, coordinates the identification of section starting points, section slicing, and determining the true segment in each section. Overall, the goal of the Resume Segmenter class is to automatically partition resumes into meaningful sections using defined section headers and a zero-shot classifier to simplify the analysis and extraction of pertinent information.

3.3.3 Model Management with Hugging Face

Model Management using Hugging Face and Flair Libraries Serialization: The Models class provides serialization methods for efficient handling of trained models. Serialization, with the pickle module, converts Python objects into byte streams for easy storage and retrieval. This ensures model states are preserved for smooth transitions during training, evaluation, and deployment phases. Loading Trained Models: The Models class helps in loading pre-trained NLP models for various tasks like NER, Zero Shot Classification, POS Tagging, and QA. Specialized models for recognizing dates and NER tasks enhance information extraction accuracy. Usage of transformer models like BERT reflects the commitment to cutting-edge technology for optimal performance. Pickling Models: The Models class implements pickling methods to serialize and deserialize loaded models, streamlining model management. The Resume Dataset includes resumes from different categories, each in a subdirectory with varying file numbers. Preprocessing is done to ensure dataset consistency and compatibility with the training process, involving text normalization, tokenization, and data augmentation if needed. Training the Named Entity Recognition (NER) model involves using resumes to recognize entities like names, organizations, and dates. The model is trained iteratively on labelled data batches, with performance assessed on a validation set Adjusting hyperparameters like learning rate, batch size, and optimizer settings to improve model performance. Zero-Shot Classification Training: The model is trained on various text inputs and their categories to classify unseen text. Training involves NLU techniques and ML algorithms. Serialization turns complex objects into byte streams for storage and transmission. Pickling models ensures persistent storage, enhancing efficiency by reducing the need for repeated model loading and optimizing resource usage. Loading Pickled Models: The Models class offers a comprehensive approach to model management, including loading pre-pickled models for enhanced efficiency. By bypassing redundant loading processes, initialization is faster with reduced runtime overhead.

In cases where pre-pickled models are not available, the method seamlessly loads models from scratch, ensuring consistent access to essential NLP capabilities. The Models class plays a crucial role in managing and utilizing pre-trained NLP models from Hugging Face Transformers and Flair libraries. It provides serialization methods, supports model loading, pickling, and enables integration of Flair models for efficient utilization of advanced NLP capabilities in resume parsing and segmentation. This feature allows users to generate Flair Sentence instances from text inputs, enabling them to harness Flair's advanced functionalities for tailored input processing. The smooth incorporation of Flair models amplifies the system's flexibility and adaptability. This holistic approach optimizes the use of cutting edge technologies, improving the efficiency and effectiveness of NLP-driven solutions in recruitment automation.

3.3.4 Job Description Skills Extraction using NLP

This part describes a methodical approach using NLP techniques to automatically extract important qualification sections from job descriptions. The main aim is to create a strong methodology that can effectively identify and extract essential qualifications mentioned in job postings. The first step involves standardizing and unifying the raw job description data through text normalization techniques and fixing any HTML tag inconsistencies. This sets the groundwork for the following processing stages. After extraction, the qualification sections go through thorough data cleaning procedures to improve data quality by removing HTML tags, special characters, and unnecessary whitespace, along with filtering out duplicates and noise to ensure accuracy and consistency. Identification of key phrases: Through careful analysis of common patterns and keywords in job postings, key phrases indicating qualification sections are identified. These phrases play a crucial role in guiding the extraction process, making it easier to find relevant qualification sections. Parsing HTML content: The BeautifulSoup library is used to parse the HTML content of job descriptions and extract important information. By navigating the parsed HTML structure, specific sections containing qualifications are targeted for extraction, efficiency of the process. Extracting improving the qualifications: The extraction process involves locating sections with qualification details within the parsed HTML elements. Various parsing techniques are utilized to handle different HTML structures and identify key phrases within specific HTML tags, leading to the extraction of qualification sections based on these phrases and the HTML context.

Creating output: The methodology results in a comprehensive list of extracted qualification sections from job descriptions. This output is valuable for further analysis and processing, aiding in candidate-job recruitment automation. matching and Tools and libraries: Powerful tools like BeautifulSoup for HTML parsing and regular expressions for text manipulation are utilized in the methodology. These tools enhance the extraction and processing of qualification information from job descriptions, improving the overall effectiveness of the methodology. The methodology relies heavily on the qualification extraction process, which involves carefully navigating through parsed HTML elements and effectively handling various HTML structures. By implementing parsing strategies and identifying key phrases within HTML tags, the extraction algorithm successfully extracts qualification sections from job descriptions with precision. By utilizing contextual cues within the HTML structure, the algorithm ensures accurate extraction results that are relevant to the context, enabling the efficient utilization of essential qualifications in recruitment.

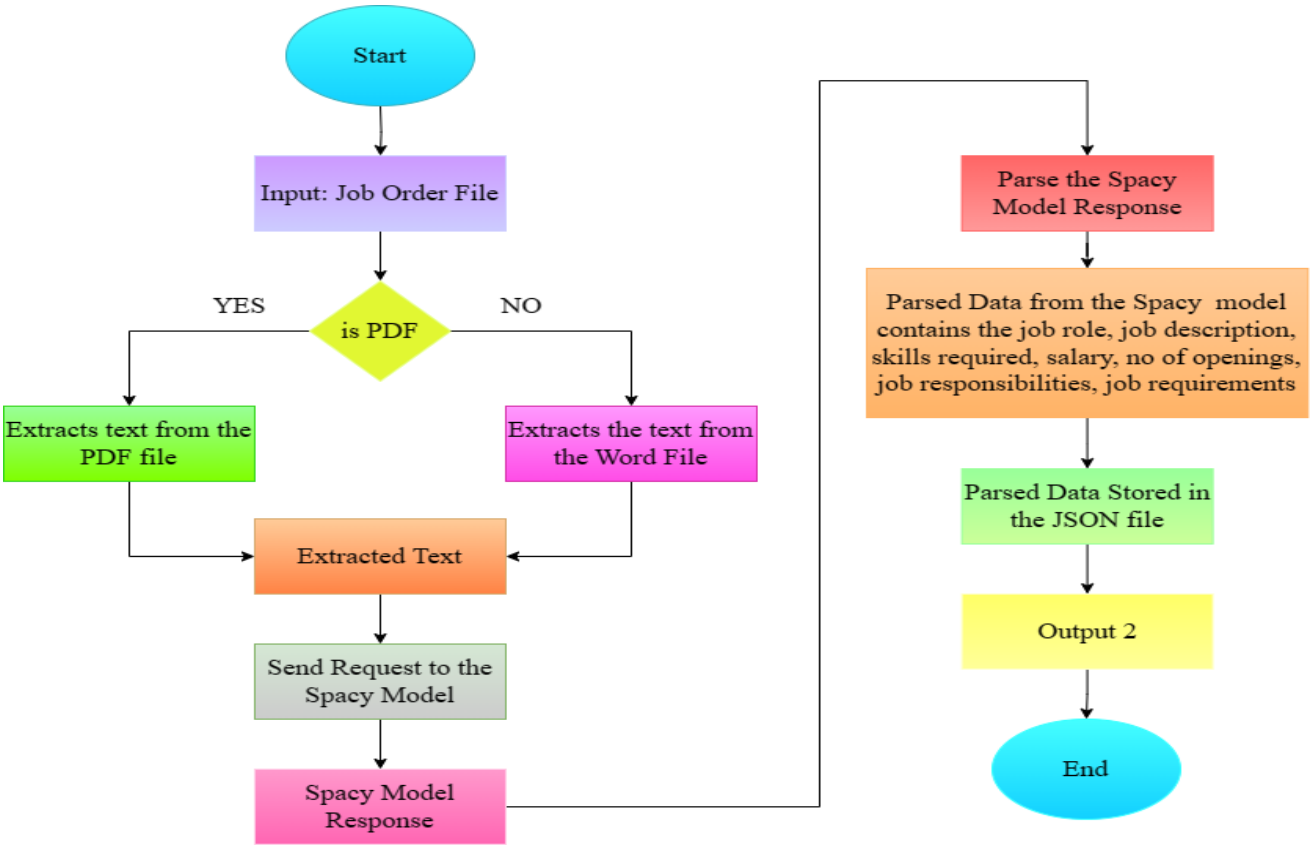


Fig 3.2: Job Parsing Model

3.3.5 Job Matching

Functions Functions readtext(txt) readtext(txt) Input: Input: txt - Text to be processed. Output: Output: List of preprocessed sentences. This function takes a raw text input, removes unnecessary characters, converts it to lowercase, removes stop words, lemmatizes words, and returns a list of preprocessed sentences. mean_pooling(model_output, attention_mask) mean_pooling(model_output, attention_mask) Input: Input: model_output : Output from the pre-trained language model. attention_mask : Attention mask for tokenized input. Output: Output: Mean-pooled sentence embeddings. Calculates mean-pooled sentence embeddings by considering attention masks to handle variable-length input sequences. model_1(sentences, sentences1) model_1(sentences, sentences1) Input: Input: sentences : List of preprocessed sentences from the job description. sentences1 : List of preprocessed sentences from the target text (product, service, or quality). Output: Output: Percentage match between the two sets of sentences. This function utilizes the Sentence Transformers library to obtain sentence embeddings for the input sentences. Cosine similarity is then computed between each pair of sentences, and the average similarity is calculated to represent the overall percentage match. textMatcher(doctext, producttext, servicetext, qualitytext) textMatcher(doctext, producttext, servicetext, qualitytext) Input: Input: doctext : Job description text. producttext : Product description text. servicetext : Service details text. qualitytext : Quality information text. Output: Output: Tuple containing the percentage match for product, service, and quality texts, respectively. This is the main function that orchestrates the comparison process. It calls readtext to preprocess the input texts and then employs percentage match for each text category. The results are returned as a tuple. Notes Notes model_1 to obtain the The code uses pre-trained models from the Hugging Face Transformers library, specifically the "all-MiniLM-L6-v2" model for sentence embeddings. The percentage match is calculated based on cosine similarity, with higher values indicating greater similarity. Mean Pooling: Mean Pooling: In natural language processing (NLP) tasks, word embeddings are often used to represent words in a continuous vector space. These embeddings capture semantic information about words. However, when dealing with entire sentences or documents, it's common to use sentence embeddings, which represent the overall meaning of a sentence. Mean pooling Mean pooling is a technique used to obtain a fixed-size representation (embedding) for a sequence of embeddings.

In the provided code, the is responsible for performing mean pooling on the output of a pre-

trained language model (MiniLM-L6-v2). mean_pooling function Breakdown: 1. Token Embeddings: Token Embeddings: The 2. Attention Mask: model_output[0] contains token embeddings for each word in the input sentence. Attention Mask: The attention_mask is used to mask out padding tokens. It is a binary mask that indicates which tokens are actual words and which ones are padding. 3. Summation: Summation: The token embeddings are multiplied element-wise by the attention mask to eliminate the contribution of padding tokens. Then, these modified embeddings are summed along the sequence axis. 4. Normalization: Normalization: The sum of the attention mask is used to normalize the sum of embeddings, ensuring that the contribution of each word is appropriately weighted. 5. Result: Result: The final result is the mean-pooled representation of the input sequence. Attention Mask: Attention Mask: The attention mask is a crucial component in many transformer-based models, such as BERT (Bidirectional Encoder Representations from Transformers) and MiniLM (Mini Language Model).

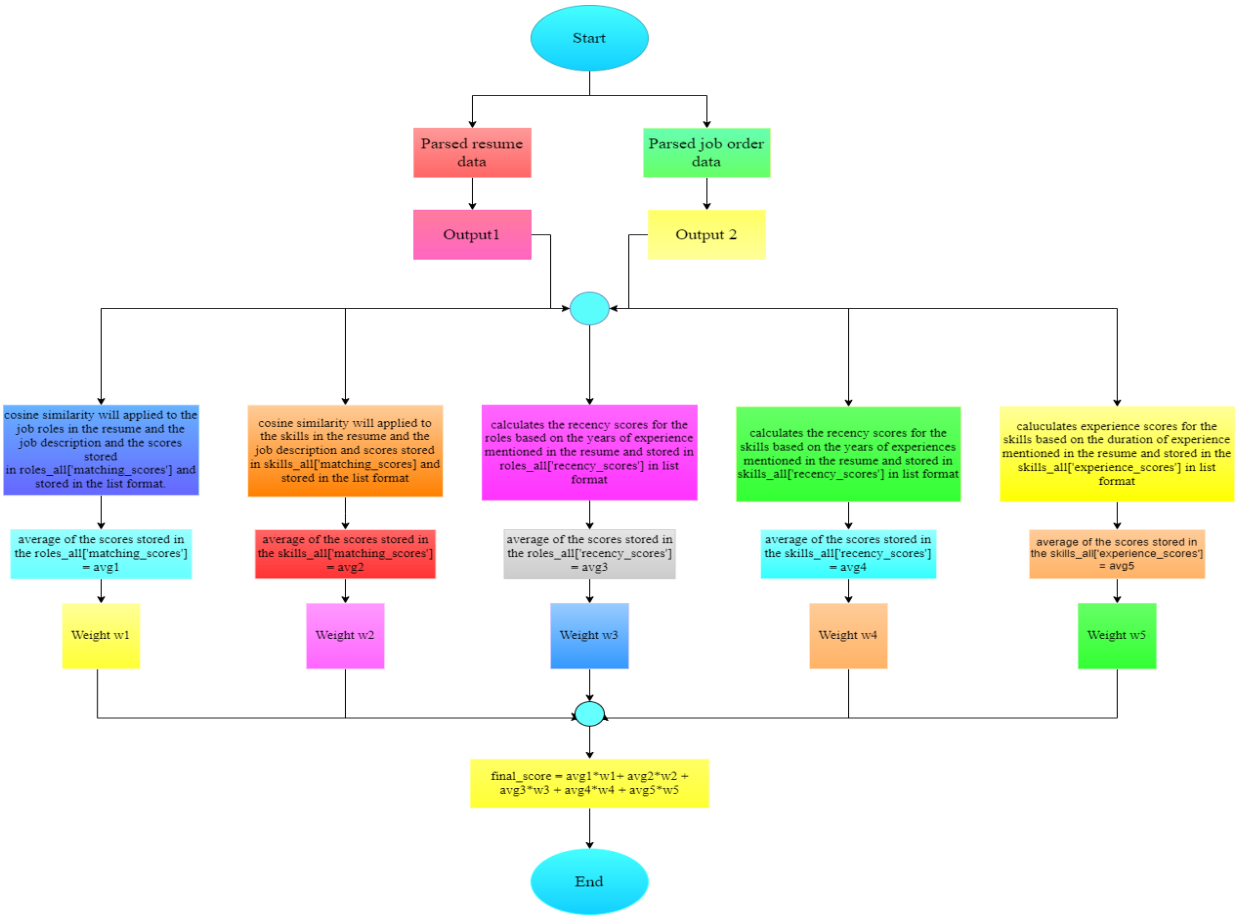


Fig 3.3: Job Compatibility Score Model

It helps the model focus on relevant parts of the input sequence and ignore padding tokens. In the provided code: The tokenizer function is used to tokenize the input sentences. The encoded_input includes the tokenized input and the corresponding attention mask. The attention mask is a binary vector with the same length as the input sequence. It has a value of 1 for actual tokens and 0 for padding tokens. During mean pooling, this attention mask is applied to the token embeddings to exclude the impact of padding tokens on the final sentence representation. By using attention masks and mean pooling, the code efficiently generates sentence embeddings that capture the contextual information of the input sentences, which is crucial for comparing the similarity between different sets of text data.

3.3.5.1 Cosine Similarity

The function textMatcher is fundamental for comparing job description (doctext) with product descriptions (producttext), service details (servicetext), and quality information (qualitytext) using sentence embeddings and cosine similarity.

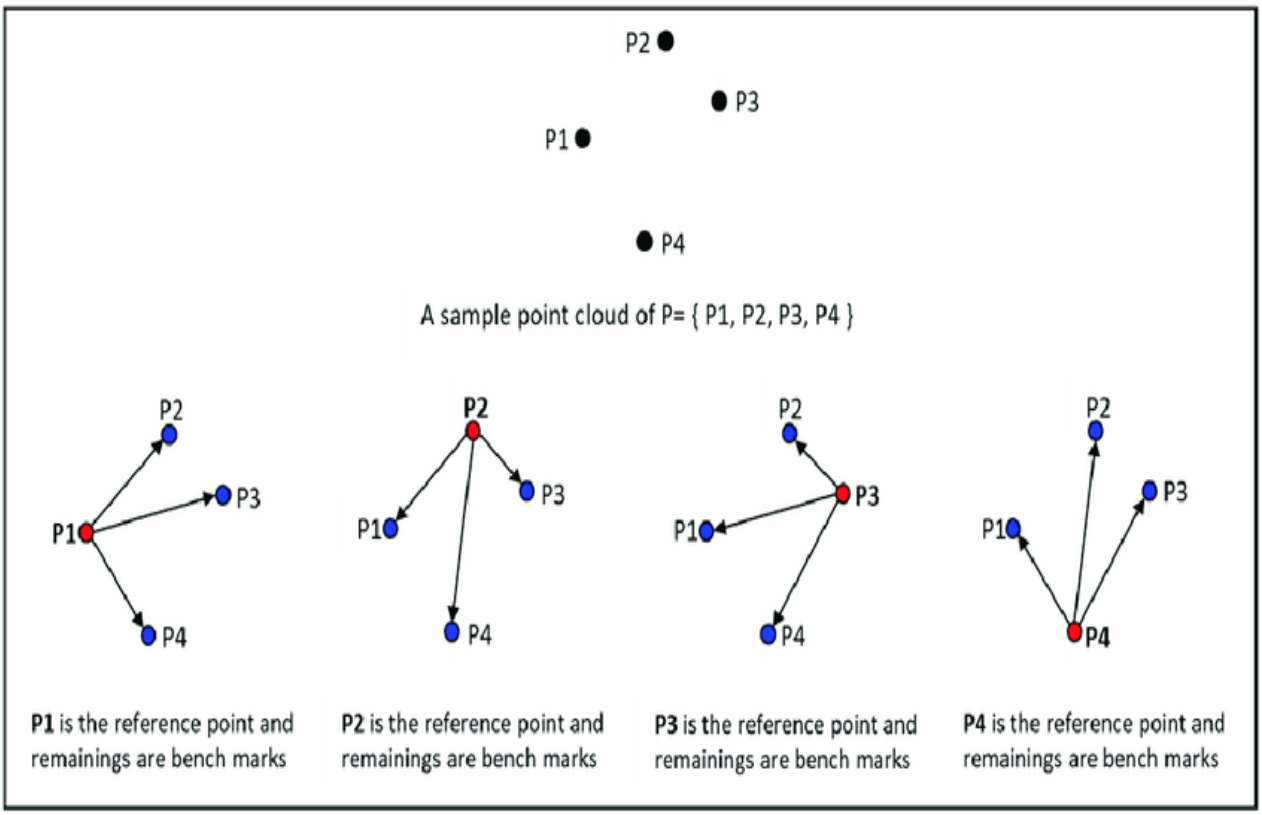


Fig 3.4: Cosine Similarity

It preprocesses input texts, removes unnecessary characters, converts to lowercase, eliminates stop words, and lemmatizes words to ensure standardized data. It then utilizes the model_1 function to calculate the match percentage between preprocessed sentences, generated by the Sentence Transformers library, from the job description and the target text . Afterwards, cosine similarity is computed for each sentence pair, and the average similarity is used to display the overall match percentage. In the background, the code utilizes pre-trained models from the Hugging Face Transformers library, specifically the "all-MiniLM-L6-v2" model for creating sentence embeddings. Cosine similarity is the chosen metric for similarity measurement, with higher values indicating closer resemblance between texts. Moreover, mean pooling is key in obtaining fixed-size embeddings for sequences of embeddings. This method, implemented through the mean_pooling function, helps maintain the contextual information captured in the sentence embeddings for accurate text comparisons. Additionally, attention masks are employed during the mean pooling to disregard the influence of padding tokens on the final sentence representation. This step is crucial for precise text comparisons.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Fig 3.5: Cosine Similarity Formula

The cosine similarity measure, crucial in natural language processing and information retrieval, allows evaluating similarity between two vectors in a multidimensional space, irrespective of their magnitudes. Vectors represent numerical values linked to specific features, with normalization ensuring magnitude doesn't affect similarity score, providing a consistent measure. Word embeddings capture semantic relationships using the MiniLM-L6-v2 pre-trained language model from the Sentence Transformers library, facilitating efficient comparison and analysis through tokenization, encoding, and pooling. The "all-MiniLM-L6-v2" model maps sentences to a dense vector space, aiding tasks like clustering and semantic search. Implementing this concept involves encoding sentences, calculating similarity matrices, and displaying results like the cosine similarity between first embeddings.

The use of word embeddings in diverse downstream tasks, such as comparing text data sets, is highlighted. The introduction of the BGEM3FlagModel showcases enhanced text retrieval capabilities across languages and input data types. Strategies like hybrid retrieval and re-ranking in RAG systems improve accuracy and efficiency. The program goes beyond simple text processing by techniques sophisticated like incorporating advanced word embeddings and models, demonstrating a comprehensive approach to natural language processing text analysis and retrieval.

CHAPTER 4
IMPLEMENTATION

4.1 VS CODE

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE

In normal terms, it facilitates the user's ability to write the code easily. Many say it is half an IDE and half an editor, but the decision is up to the coders. Any program/software we see or use works on the code running in the background. Traditionally, coding was done in traditional or basic editors like Notepad! These editors used to provide basic support to the coders.

Some were so basic that writing basic English-level programs was very difficult. As time went by, some programming languages needed a specific framework and support for further coding and development, which was impossible using these editors. VI Editor, or Sublime Text Editor, is one of the many kinds of editors that came into existence. VS Code is the most prominent, which supports almost every coding language. Its features let the user modify the editor as per the usage, which means the user can download the libraries from the internet and integrate them with the code as per his requirements.

Features:

Some of the remarkable features of VS Code are:

- **Language Support**: VS Code supports a wide range of programming languages, including but not limited to JavaScript, TypeScript, Python, C#, Java, Go, Ruby, and others.
- **Intelli-Sense:** It can detect if any snippet of code is left incomplete. Also, common variable syntax and variable declarations are made automatically. Ex: If a certain variable is being used

=

in the program and the user has forgotten to declare it, intelli-sense will declare it for the user.

- **Extensions and Support:** Usually supports all the programming languages, but if the user/programmer wants to use a programming language that is not supported, he can download and use the extension. And performance-wise, the extension doesn't slow down the editor as it runs as a separate process.

- **Repository:** With the ever-increasing demand for the code, secure and timely storage is equally important. It is connected with Git or can be connected with any other repository for pulling or saving the instances.

- **Web-Support:** Comes with built-in support for Web applications. So web applications can be built and supported in VSC.

- **Hierarchy Structure:** The code files are located in files and folders. The required code files also have some files that may be required for other complex projects. These files can be deleted as per convenience.

- **Improving Code:** Some code snippets can be declared a bit differently, which might help the user in the code. This function prompts the user, wherever necessary, to change it to the suggested option.

- **Terminal Support:** Often, the user must start from the root of the directory to start with a particular action; an in-built terminal or console provides user support to not switch between two screens for the same.

- **Multi-Projects:** Multiple projects containing multiple files/folders can be opened simultaneously. These projects/folders might or might not be related to each other.

- **Git Support:** Resources can be pulled from Git Hub Repo online and vice-versa; saving can be done too. Resource pulling also means cloning the code made available on the internet. This code can later be changed and saved.

- **Git Integration:** With VS Code's built-in Git integration, you can perform version control tasks directly within the editor. You can stage, commit, and push changes, view diffs, and manage branches without switching to a separate Git client.

- **Command Palette:** The Command Palette is a useful tool that allows you to quickly execute commands and access various functionalities with just a few keystrokes. It offers a fast

=

and efficient way to navigate the editor and perform actions without relying on menus and toolbars.

- **Debugging Support:** VS Code offers a robust debugging experience. It lets you set breakpoints, inspect variables, step through code, and handle exceptions. This makes finding and fixing bugs in your applications a more manageable task.

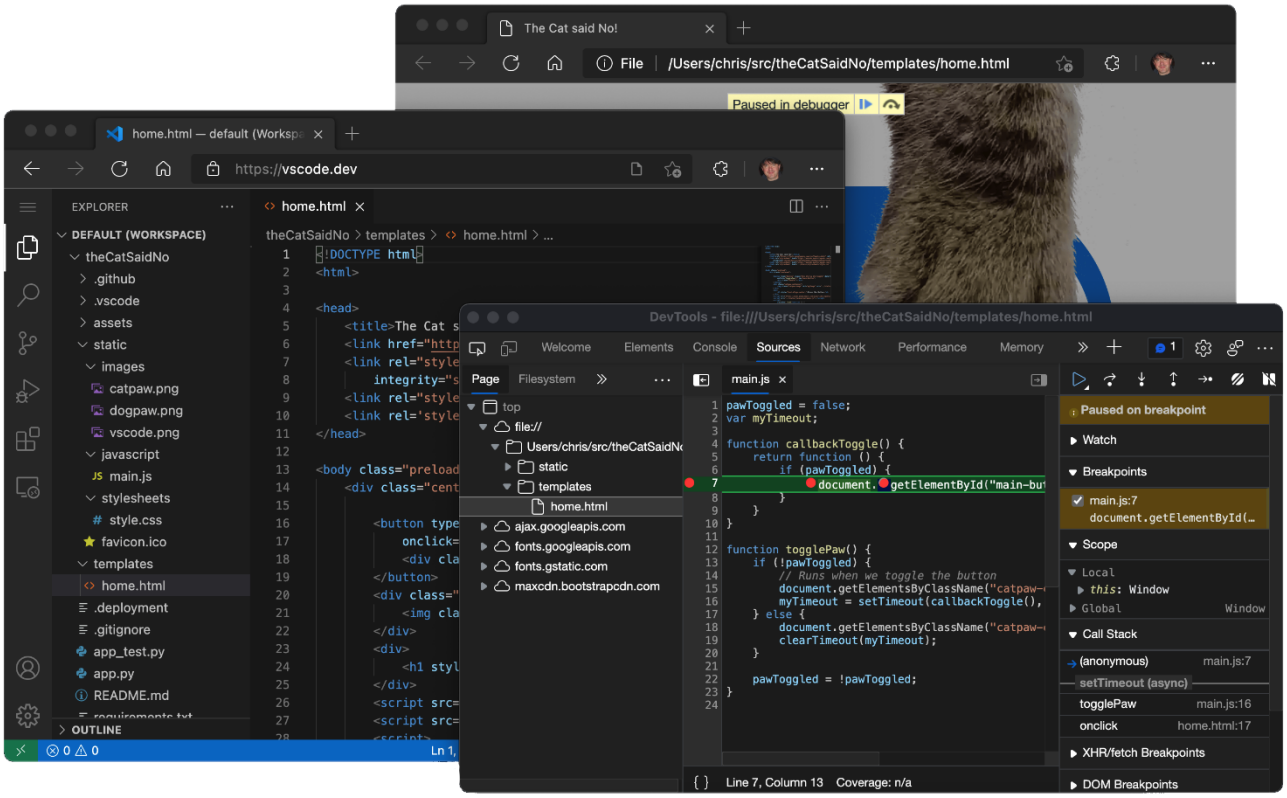


Fig 4.1: Visual Studio Code Interface

Components:

- Editor
- File Management
- Window Management
- Preference Settings
- Debugger
- Help

Plugins:

- ES7 React/Redux/React-Native/JS snippets
- Prittier
- ESLint

4.2 PYTHON

Python is a high-level, general-purpose, and very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber, etc.

Its simplicity, readability, and extensive library support make Python an ideal choice for a wide range of applications. As the demand for Python developers continues to rise, proficiency in this language opens up diverse career opportunities in various industries. With its versatility and widespread adoption, Python remains at the forefront of innovation, driving advancements in technology and shaping the future of software development. Python's versatility and extensive library support make it a top choice for web development, machine learning, and other cutting-edge applications in the software industry.

Python Libraries:

- Natural Language Processing
- Multimedia
- Scientific computing
- Text processing
- Parsing (like OpenCV, Pillow

Applications:

- Web and Internet Development
- Scientific and Numeric
- Education
- Desktop GUI’s
- Software Development
- Business Application

4.3 OPENCV

OpenCV (OpenSource Computer Vision) is an open-source library of computer vision and machine learning algorithms designed to help developers build applications that can perform a variety of image and video processing tasks. It was first developed by Intel in 1999 and is now maintained by the OpenCV Foundation. OpenCV provides a wide range of tools and functions for tasks such as image processing, object detection, feature detection and matching, optical flow, and camera calibration. It supports multiple programming languages including C++, Python, Java, and MATLAB, and can be used on a range of operating systems including Windows, Linux, macOS, Android, and iOS.

One of the key features of OpenCV is its ability to work with real-time video streams, making it particularly well-suited for applications such as video surveillance, facial recognition, and augmented reality. It also includes a range of pre-trained models for tasks such as face detection and recognition, pedestrian detection, and text detection. OpenCV has a large and active community of users and developers who contribute to its development and improvement. This community provides resources, tutorials, and support to help developers learn and use the library effectively.

OpenCV provides a wide range of tools and functions for computer vision and machine learning tasks.

Tools and Modules:

- **Core:** This module provides basic functionality such as data structures for multi-dimensional arrays and matrices, image I/O functions, and basic image processing functions.
- **Image Processing:** This module provides functions for image filtering, thresholding, edge detection, morphological operations, and other common image processing tasks.
- **Feature Detection and Description:** This module provides functions for detecting and describing features in images, such as corners, blobs, and edges. It also includes functions for feature matching, which can be used for tasks such as object recognition and tracking.
- **Video Analysis:** This module provides functions for working with video streams, including reading and writing video files, video stabilization, optical flow, and motion detection.

- **Object Detection:** This module provides functions for detecting objects in images and videos using pre-trained models, such as Haar cascades or deep neural networks.
- **Machine Learning:** This module provides functions for machine learning tasks suchas classification, regression, clustering, and dimensionality reduction. It also includes tools for training and evaluating machine learning models
- **Deep Neural Networks:** This module provides functions for building and training deep neural networks using popular frameworks such as TensorFlow and Caffe.
- **High-level GUI and Media:** This module provides high-level functions for creating graphical user interfaces and working with media files, such as displaying images and videos, capturing video streams, and playing back audio files.

These are just some of the tools and modules available in OpenCV. The library is constantly evolving and improving, with new tools and modules being added regularly.

4.4 JAVASCRIPT

JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. It is one of the three core technologies of World Wide Web content production, along with HTML and CSS.

JavaScript is used to make web pages interactive. For example, you can use JavaScript to create animated graphics, menus, and forms. You can also use JavaScript to validate user input and send data to a server.

JavaScript is a lightweight, interpreted programming language. This means that it does not need to be compiled before it can be run. JavaScript code is executed by a JavaScript engine, which is a program that interprets JavaScript code and executes it.JavaScript is a very popular programming language. It is used by millions of web developers around the world. JavaScript is also used in many other applications, such as games, mobile apps, and desktop apps.

Here are some of the things you can do with JavaScript:

- Add interactivity to web pages
- Create animated graphics
- Create menus and forms
- Validate user input
- Send data to a server

- Create games
- Create mobile apps
- Create desktop apps

4.4.1 ReactJs

ReactJS, also known as React, is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and corporations. React can be used as a base in the development of single-page or mobile applications.

React uses a declarative paradigm that makes it easier to create interactive UIs. React makes it possible to compose complex UIs from small and isolated pieces of code called "components".

React can also render on the server using Node.js, and it can power native mobile apps using React Native.

Here are some of the features of React:

- Declarative: React makes it easy to create interactive UIs by using a declarative programming style.
- Component-based: React makes it easy to break down complex UIs into smaller, reusable components.
- Virtual DOM: React uses a virtual DOM to efficiently update the UI.
- One-way data flow: React uses a one-way data flow to make it easier to reason about your code.
- Server-side rendering: React can render on the server using Node.js.
- Native mobile apps: React can power native mobile apps using React Native.
- React is a popular choice for building user interfaces because it is easy to use, efficient, and scalable. It is also supported by a large community of developers.

4.4.2 ExpressJs

Express.js (or simply Express) is a back-end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web and mobile applications.

It is one of the most popular Node.js frameworks, providing a robust set of features for building single-page, multi-page, and hybrid web applications. It is also used to build APIs.

Express.js is known for its speed, simplicity, and flexibility. It provides a thin layer of fundamental web application features, without obscuring Node.js features. This makes it easy to get started with Express.js, and also allows developers to customize their applications to their specific needs. Express.js is used by a wide range of companies, including PayPal, Uber, and IBM. It is also a popular choice for startups and small businesses.

Here are some of the key features of Express.js:

- Robust routing:
- Express.js provides a powerful routing system that makes it easy to create and manage routes for your application.
- High performance:
- Express.js is designed to be high-performance, making it ideal for building scalable applications.
- HTTP helpers:
- Express.js provides a number of HTTP helper methods that make it easy to work with HTTP requests and responses.
- Asynchronous programming:
- Express.js supports asynchronous programming, making it easy to build efficient and responsive applications.

4.5 NUMPY

NumPy is a Python library used for numerical computing, specifically for scientific computing and data analysis. It provides powerful data structures for efficient computation with large arrays and matrices, as well as a collection of functions for operating on these arrays.

Features:

- **Ndarray:** A fast and flexible array object for storing and manipulating large arrays of homogeneous data.
- **Mathematical functions:** NumPy provides a wide range of mathematical functions such as trigonometric, logarithmic, and exponential functions.

- **Linear algebra operations:** NumPy provides a suite of linear algebra functions,

including matrix multiplication, eigenvalues, and singular value decomposition.

- **Random number generation:** NumPy includes a random number generation module for creating arrays of random data.
- **Broadcasting:** NumPy allows for arithmetic operations between arrays of different shapes and sizes, which can greatly simplify code and improve performance.

Applications:

- **Scientific computing:** NumPy is extensively used in scientific computing and data analysis, particularly in fields such as physics, chemistry, engineering, and statistics. It provides a powerful and efficient framework for numerical computations and simulations.
- **Data analysis and visualization:** NumPy provides tools for performing complex mathematical operations on large datasets, such as statistical analysis, linear algebra, and Fourier analysis. Additionally, NumPy can be used with other Python libraries such as Matplotlib and Pandas to create powerful data visualization tools.
- **Machine learning:** NumPy is widely used in machine learning, particularly in tasks such as deep learning, natural language processing, and computer vision. It provides efficient support for linear algebra operations such as matrix multiplication and inversion, which are fundamental to many machine learning algorithms.
- **Image and signal processing**: NumPy provides tools for processing and manipulating images and signals, including Fourier analysis, image filtering, and signal processing. These tools are widely used in applications such as medical imaging, audio processing, and computer vision.
- **Gaming and simulation:** NumPy is used in game development and simulation environments, where it can be used to simulate complex physics and mathematical models. It provides efficient support for matrix operations, which are fundamental to many physics-based simulations.

4.6 BootStrap

Bootstrap is a free, open-source framework for front-end development that helps create responsive websites and web apps. It's designed for mobile-first websites, and includes a collection of syntax for template designs. Bootstrap is made for people of all skill levels, devices, and projects, and

includes: CSS preprocessors Bootstrap uses vanilla CSS, but its source code uses the two most popular CSS preprocessors, Less and Sass Responsiveness. Bootstrap scales websites and applications with a single code base, from phones to tablets to desktops with CSS media queries.HTML and CSS based design templates includes typography, forms, buttons, tables, navigation, modals, image carousels, and more

Bootstrap is a free, opensource front-end development framework for the creation of websites and web apps. Designed to enable responsive development of mobile-first websites, Bootstrap provides a collection of syntax for template designs.

What is Bootstrap used for?

Bootstrap makes responsive web design a reality. It makes it possible for a web page or app to detect the visitor's screen size and orientation and automatically adapt the display accordingly. The mobile-first approach assumes smartphones, tablets and task-specific mobile apps are employees' primary tools for getting work done. Bootstrap addresses the requirements of those technologies in design and includes UI components, layouts, JavaScript tools and the implementation framework. The software is available precompiled or as source code.

Mark Otto and Jacob Thornton developed Bootstrap at Twitter to improve the consistency of tools used on the site and to reduce maintenance. The software was formerly known as Twitter Blueprint and is sometimes referred to as Twitter Bootstrap.

4.6.1 What is bootstrap in a computer?

In computing, the term bootstrap means to boot or to load a program into a computer using a much smaller initial program to load in the desired program, which is usually an OS.

4.6.2What is Bootstrap CSS?

The most popular CSS framework for developing responsive and mobile-first websites is Bootstrap. The newest version is Bootstrap 5.

Bootstrap 5 offers numerous pre-designed components, utilities, and styles to streamline web development and ensure consistent design across various devices and screen sizes. Its modular approach and extensive documentation make it easy for developers to create modern, responsive websites quickly and efficiently. Bootstrap 5 also introduces several new features and enhancements, including updated utility classes, improved grid system, and revamped components, further enhancing its flexibility and usability for web development projects.

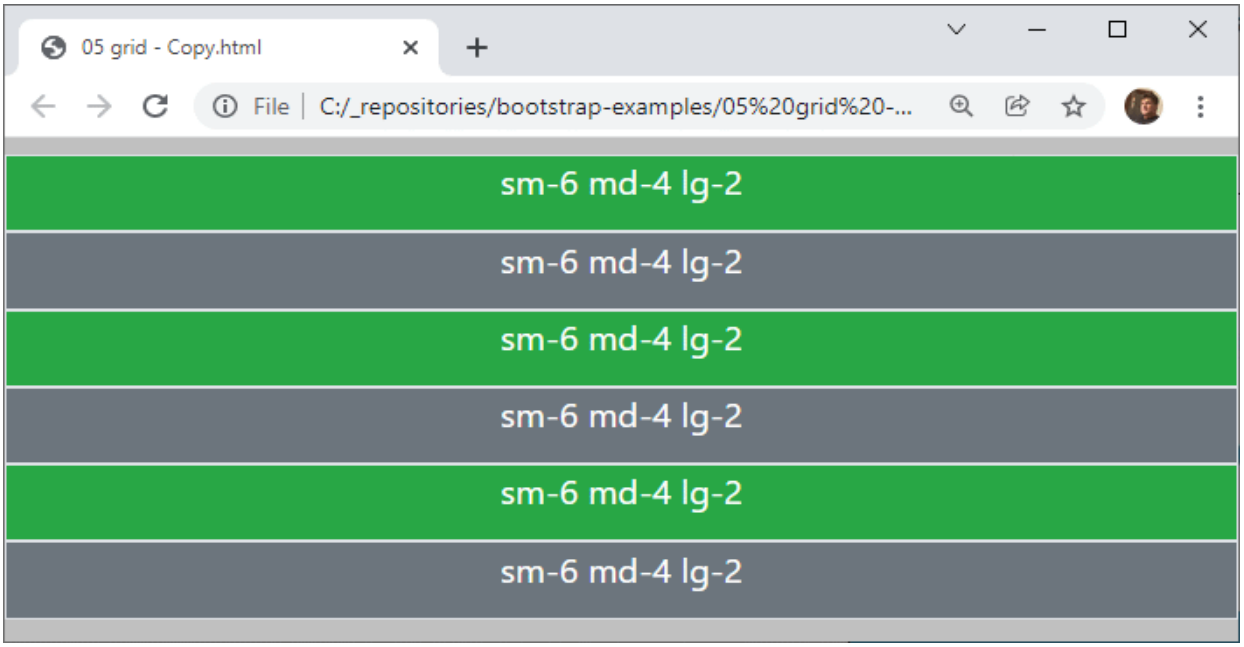


Fig 4.2: Bootstrap Grid System Layout

4.6.3 What is bootstrapping in statistics?

In statistics, bootstrapping describes the process of resampling a data set to create many simulated samples. This approach enables users to calculate standard errors, perform hypothesis testing and construct confidence intervals for different types of sample statistics.

4.6.4 What is bootstrap distribution?

The resampling procedure, bootstrapping, uses data from a sample to generate a sampling distribution by repeatedly taking random samples from a known sample.

4.6.5 What is bootstrapping machine learning?

To improve the stability of machine learning (ML) algorithms, Bootstrap sampling is used in an ensemble algorithm called Bootstrap aggregating or bagging. In bootstrapping ML, a specific number of equally sized subsets of a data set are extracted with the replacement.

4.6.6 What is Bootstrap Protocol?

Bootstrap Protocol (BOOTP) is an internet protocol in which a network user can be configured automatically to receive an IP address and have an OS booted without user involvement. A network administrator manages the BOOTP server, which assigns the IP address automatically

from a pool of addresses for a specific duration.

4.6.7 What is bootstrap CDN?

Bootstrapped websites often need an increase in speed. A content delivery network helps resolve this issue and delivers static content to users faster. It's the best approach to simultaneously enhance user engagement and page loading speed.

4.6.8 What is bootstrapping in general terms?

In the physical world, a bootstrap is a small strap or loop at the back of a leather boot that enables the boot to be pulled on. In general use, bootstrapping is leveraging a small initial effort into something larger and more significant. The metaphor, "pulling yourself up by your bootstraps," means to achieve success from a small beginning.

As a framework, Bootstrap includes the basics for responsive web development, so developers only need to insert the code into a pre-defined grid system. The Bootstrap framework is built on Hypertext Markup Language (HTML), cascading style sheets (CSS) and JavaScript. Web developers using Bootstrap can build websites much faster without spending time worrying about basic commands and functions

4.7 Pseudo Code

```
from tracemalloc import start
from matplotlib.pyplot import contour
from parcv.Models import Models
from datetime import datetime
from dateutil import parser
import re
from string import punctuation
from collections import Counter
import math

class ResumeParser:

    def __init__(self, ner, ner_dates, zero_shot_classifier, tagger, qa_squad):
        self.models = Models()
        self.ner, self.ner_dates, self.zero_shot_classifier, self.tagger, self.qa_squad = ner, ner_dates, zero_shot_classifier, tagger,
        self.parsed_cv = {}

    def parse(self, resume_segments):
        for segment_name in resume_segments:
            resume_segment = resume_segments[segment_name]
            if segment_name == "contact_info":
                self.new_parse_contact_info(resume_segment)
            elif segment_name == "work_and_employment":
                self.new_parse_job_history(resume_segment)
            elif segment_name == "education_and_training":
                self.parse_education_history(resume_segment)
            elif segment_name == "skills":
                self.parse_skills(resume_segment)
        return self.parsed_cv
```

Fig 4.3: ResumeParser.py

```python
import re
import os
import logging
import docx
import pdfplumber

class ResumeReader:

    def convert_docx_to_txt(self, docx_file,docx_parser):
        """
            A utility function to convert a Microsoft docx files to raw text.

            This code is largely borrowed from existing solutions, and does not match the style of the rest of this repo.
            :param docx_file: docx file with gets uploaded by the user
            :type docx_file: InMemoryUploadedFile
            :return: The text contents of the docx file
            :rtype: str
        """

        doc = docx.Document(docx_file)
        allText = []
        for docpara in doc.paragraphs:
            allText.append(docpara.text)
        text = ' '.join(allText)
        try:
            clean_text = re.sub(r'\n+', '\n', text)
            clean_text = clean_text.replace("\r", "\n").replace("\t", " ")  # Normalize text blob
            resume_lines = clean_text.splitlines()  # Split text blob into individual lines
            resume_lines = [re.sub('\s+', ' ', line.strip()) for line in resume_lines if
                            line.strip()]  # Remove empty strings and whitespaces
            return resume_lines, text
```

Fig 4.4: ResumeReader.py

```python
from parcv.Models import Models

class ResumeSegmenter:

    def __init__(self, zero_shot_classifier):
        self.zero_shot_classifier = zero_shot_classifier

    objective = (  ...
    )

    work_and_employment = (  ...
    )

    education_and_training = (  ...
    )

    skills_header = (  ...
    )

    misc = (  ...
    )

    accomplishments = (  ...
    )


    def find_segment_indices(self, string_to_search, resume_segments, resume_indices):
        for i, line in enumerate(string_to_search):

            if line[0].islower():
                continue

            header = line.lower()
```

Fig 4.5: ResumeSegmenter.py

4.7.1 General Preprocessing and Translation

The following program is designed to filter out duplicate job postings, translate non-English job postings into English, and perform initial general preprocessing. The results will be outputted in an Excel file, which can be used for further analysis.

Load raw data

- Import of job advertisements

```
import pandas as pd
# Fetching raw data
workbook =
'https://github.com/schmcklr/skill_extractor/blob/main/job_data/job_ad
vertisements.xlsx?raw=true'

# Import of tabs
job_data = pd.read_excel(workbook, sheet_name="data")
```

General Preprocessing

- Convert to lower case
- Convert dates to datetime
- Elimination of duplicates

```
# Initial preprocessing of job advertisements
import pandas as pd
from nltk.corpus import stopwords

# Convert text to lower case
job_data = job_data.apply(lambda x: x.astype(str).str.lower())

# Convert 'created_at' to datetime
job_data['created_at'] = pd.to_datetime(job_data['created_at'])

# Extract the year from the 'created_at'
job_data['year'] = job_data['created_at'].dt.year

# Elimination of duplicates
job_data = job_data[~job_data.duplicated(subset=['title', 'year'],
keep='first')]
job_data = job_data[~job_data.duplicated(subset=['description',
'year'], keep='first')]

print(len(job_data['created_at']))
```

```
3062
```

Translation

- Function for data translation

```
#!pip install --upgrade googletrans --quiet
!pip install --upgrade translatepy --quiet
!pip install langdetect --quiet
from bs4 import BeautifulSoup
from translatepy import Translator
from langdetect import detect
import langdetect

# Function for language detection
def detect_language(text):
    try:
        return detect(text)
    except langdetect.lang_detect_exception.LangDetectException:
        return 'unknown'

# Initialization of global variables
translated_job_ads = 0
all_job_adds = 0

# Initialization of translator
translator = Translator()

# Function for translation of job description
def translate_job_description(text, count, html):
    # Global keyword to access variables global
    global all_job_adds
    global translated_job_des

    if count == 'y':
        all_job_adds += 1

    if html:
      # Erstellen eines BeautifulSoup-Objekts
      soup = BeautifulSoup(text, 'html.parser')
      # Entfernen von HTML-Tags und Ersetzen durch Leerzeichen
      cleaned_text = soup.get_text(separator=' ')
    else:
      cleaned_text = text

    # Translate if text not in English
    if detect_language(cleaned_text) != 'en':
        # Global keyword to access global variables
        global translated_job_ads
        # Variable to count number of translated job ads
```

```
        if count == 'y':
            translated_job_ads += 1
        try:
          translation = translator.translate(text, "English")
          translated = translation.result
        except Exception as e:
          translated = text

        # Store translated job description (for development purposes
only)
        if count == 'n':
          translated_job_des.append([text, translated])
    else:
        translated = text
    return translated
```

- Function to translate the content within HTML tags when there are no spaces between the tags and the enclosed text (disabled by default, because of higher runtime)

```
from bs4 import BeautifulSoup
import pandas as pd

def process_html(html_text):
    soup = BeautifulSoup(html_text, 'html.parser')

    # Iterate over the relevant tags and apply the function to the
element string
    for element in soup.find_all():
        if element.string is not None:
            modified_text = translate_job_description(element.string,
'n')
            element.string.replace_with(modified_text)

    # Format the HTML txext to improve readability and structure
    formatted_html_text = soup.prettify()
    return formatted_html_text
```

- Translation of job advertisements

```
# Initialize list to store translated job descriptions
translated_job_des = []

# Translation of job ads (columns: title, description, describtion
with html tags)
job_data['title'] = job_data['title'].apply(lambda x:
translate_job_description(x, 'n', False))
job_data['description_translated'] =
job_data['description'].apply(lambda x: translate_job_description(x,
'y', True))
```

```
# Needed to also translate the headlines and tags without spaces (disabled by default) #job_data['rawDescriptionTranslatedWithTags'] = job_data['rawDescriptionTranslated'].apply(lambda x: process_html(x))


# User info (number of ads that have been translated )
print('Translation successful! ' + str(translated_job_ads) + '/' + str(all_job_adds) + ' job advertisements were translated.')

# create dataframe that includes text before and after translation (for development purposes only)
#translated_descriptions = pd.DataFrame(translated_job_des, columns=['Original Text', x'Translated Text'])
#translated_descriptions.to_excel('translated_job_adx.xlsx', index=False)

# Export translated text (for development purposes only)
#translated_descriptions.to_excel('translated_job_adx.xlsx', index=False)
```

```
Translation successful! 382/3062 job advertisements were translated.
```

Export

- Export dataframe to excel

```
# Export dataframe to excel
job_data.to_excel('job_data_general_preprocessed_and_translated.xlsx', index=False)
```

- Download excel

```
from google.colab import files
import ipywidgets as widgets
from IPython.display import display

# Create the path for the saved model file
path = 'job_data_general_preprocessed_and_translated.xlsx'

# Create a button for downloading the model
download_button = widgets.Button(description="Download Excel")

# Link the download function to the button click event
def on_download_button_click(b):
    # Download the model file
    files.download(path)

download_button.on_click(on_download_button_click)
```

```
# Display the download button
display(download_button)
```


```
{"model_id":"4f5f67ab39db449ab1deb04420507982","version_major":2,"vers
ion_minor":0}
```

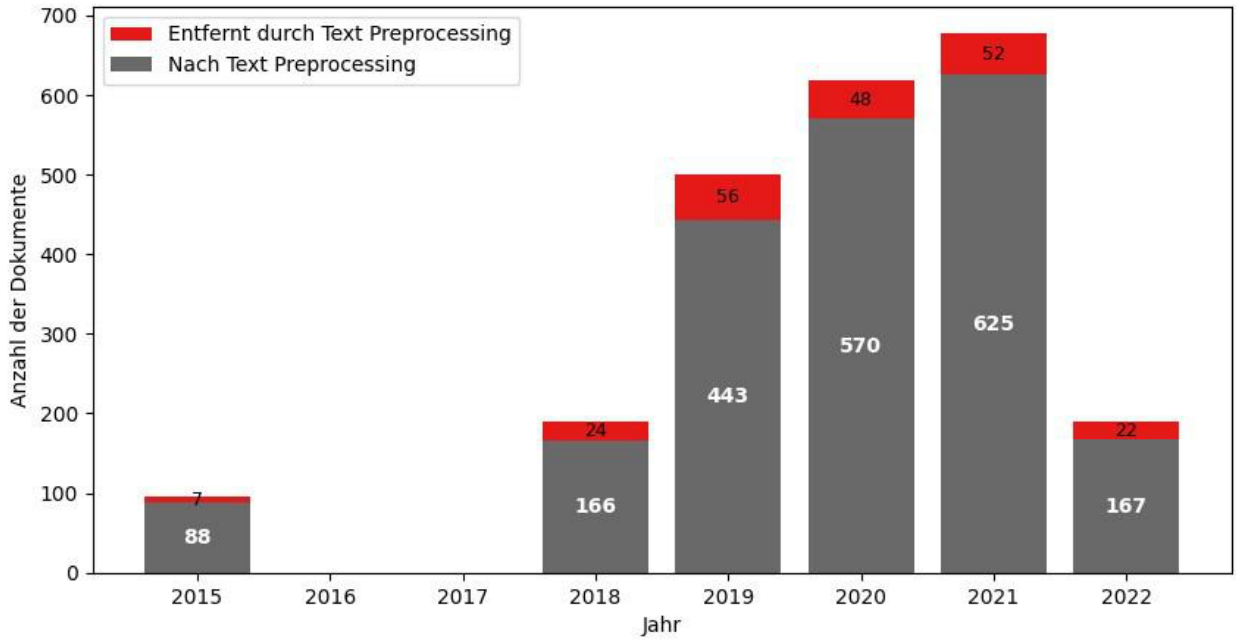

4.7.2 Document statistics

- number of documents
- tokens per document
- characters per document
- Number of job postings per year

```
import matplotlib.pyplot as plt

# Number of jobs ads per year after text preprocessing
number_of_ads_per_year_after_prepro = {}
for year in docs_by_year: number_of_ads_per_year_after_prepro[year] =
   len(docs_by_year[year])

# Set colors
color = '#696969'
color2 = '#e61919'#plt.cm.get_ cmap('Reds')(0.4) # Adjust the
saturation here (0.5 means 50% saturation)
#color2 = sns.color_palette("husl", 1)

# Sort the data by years
sorted_data = sorted(number_of_ads_per_year_after_prepro.items())
sorted_data_before_prepro = sorted(number_of_ads_per_year.items())


# Creating a figure object
fig = plt.figure(figsize=(10,5))
fig, ax = plt.subplots(figsize=(10,5))


# Creating bar plot with jobs ads per year before text preprocessing
years_before_prepro = [x[0] for x in sorted_data_before_prepro]
counts_before_prepro = [x[1] for x in sorted_data_before_prepro]
bars_before_prepro = ax.bar(years_before_prepro, counts_before_prepro,
color=color2, label='Entfernt durch Text Preprocessing')

# Creating bar plot with jobs ads per year after text preprocessing
years = [x[0] for x in sorted_data]
counts = [x[1] for x in sorted_data]
bars = ax.bar(years, counts, color=color,label='Nach Text
Preprocessing')
```

```
# Set lable for number of postings
for bar in bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2., height/2, int(height),
            ha='center', va='center', color='white',
fontweight='bold', fontsize=10)


# Set lable for number of postings before preprocessing
for i, bar in enumerate(bars_before_prepro):
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2., counts[i] + (height -
counts[i])/2, (str(int(height - counts[i]))),
            ha='center', va='center', color='black', fontsize=9)


# Set axis, labels, legend and title
ax.set_xlabel('Jahr', fontsize=10)
ax.set_ylabel('Anzahl der Dokumente', fontsize=10)
ax.legend()
#plt.title("Number of job postings by year", fontsize=16)

# Display chart
plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```

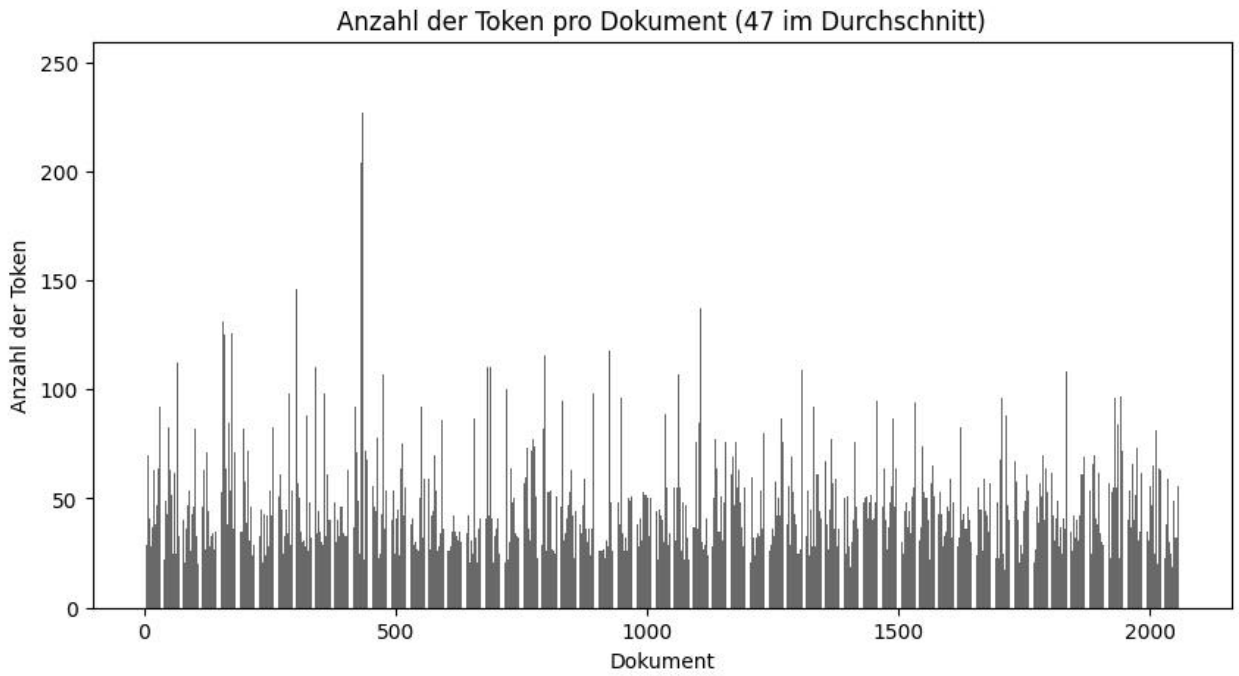


- Tokens per document

```
import matplotlib.pyplot as plt
```

```
# Create a figure with a specific size
fig = plt.figure(figsize=(10, 5))

# Create subplots within the figure with a specific size
fig, ax = plt.subplots(figsize=(10, 5))

# Determine the number of tokens per document
token_counts = [len(doc) for doc in filtered_docs_all]

# Create an index for the x-axis
x = range(1, len(filtered_docs_all) + 1)

# Calculate the token mean
mean = sum(token_counts) / len(token_counts)

# Create the chart and define labels
plt.bar(x, token_counts,color=color)
plt.xlabel('Dokument')
plt.ylabel('Anzahl der Token')
plt.title('Anzahl der Token pro Dokument (' + f"{mean:.0f} im
Durchschnitt)")
plt.show()
```

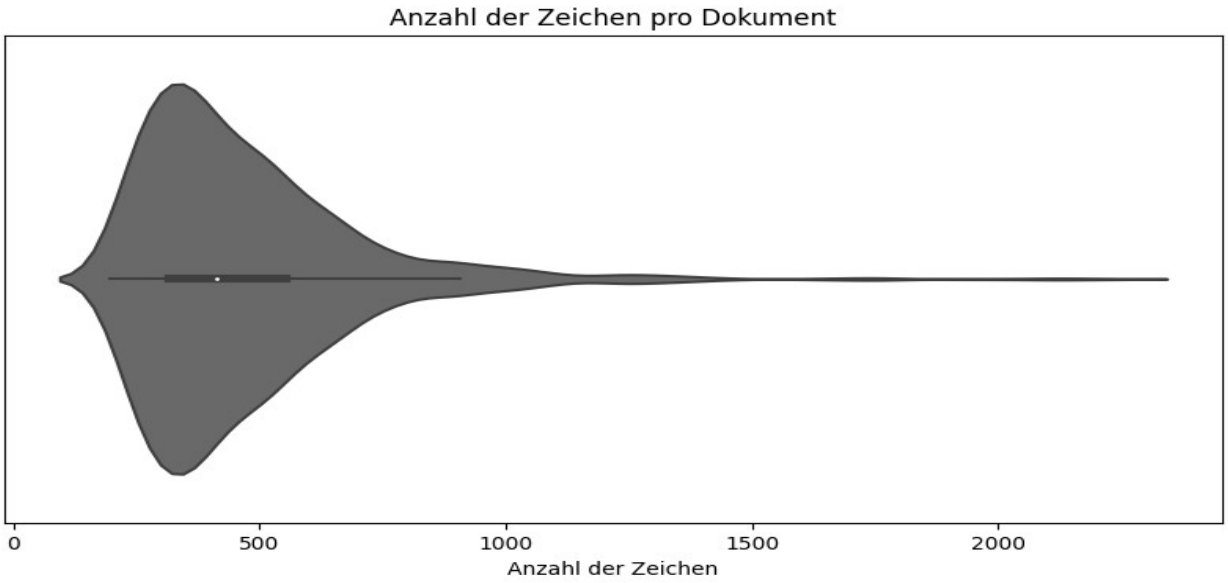
```
<Figure size 1000x500 with 0 Axes>
```



- Characters per document

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import statistics

# Zeichenanzahl pro Dokument (ohne Unterstrich "_")
char_counts = [len(" ".join(doc).replace("_", "")) for doc in
filtered_docs_all]

# Erstellen Sie den horizontalen Violinplot
plt.figure(figsize=(10, 5))
sns.violinplot(x=char_counts, color=color, orient="h")

#plt.ylabel('Dokument')
plt.yticks([])
plt.xlabel('Anzahl der Zeichen')
plt.title('Anzahl der Zeichen pro Dokument')

plt.show()

# Berechnen Sie Min, Max, Median und Durchschnitt
min_value = min(char_counts)
max_value = max(char_counts)
median_value = statistics.median(char_counts)
mean_value = statistics.mean(char_counts)

# Ausgabe der Werte als Text unter dem Plot
print(f'Min: {min_value}')
print(f'Max: {max_value}')
print(f'Median: {median_value}')
print(f'Durchschnitt: {mean_value:.0f}'
```



CHAPTER 5
SYSTEM DESIGN

5.1 INTRODUCTION OF INPUT DESIGN

The introduction of input design in system development is a critical phase that involves planning and creating interfaces through which users interact with the system. It aims to optimize user input processes, ensuring efficiency, accuracy, and user satisfaction.

Here's a more detailed description:

Understanding User Requirements: Input design begins with a thorough understanding of user needs and expectations. This involves conducting user research, gathering requirements, and defining the scope of the input design.

User Interface Design: Based on user requirements, designers create user interfaces that are intuitive and easy to use. This includes designing forms, screens, and dialogs that allow users to input data and interact with the system.

Input Controls: Designers choose appropriate input controls (e.g., text boxes, dropdown menus, radio buttons) based on the type of data users need to input. They ensure that these controls are easy to understand and use.

Layout and Organization: The layout of input elements is carefully considered to make it easy for users to navigate and complete forms. Elements are organized logically, with clear labels and instructions.

Feedback and Validation: The input design includes mechanisms for providing feedback to users, such as confirming that data has been successfully submitted. It also includes validation checks to ensure that the data entered by users is correct and meets specified criteria.

Accessibility: Designers ensure that the input interface is accessible to all users, including those with disabilities. This may involve providing alternative input methods or ensuring that the interface works with assistive technologies.

User Guidance: Input design includes providing users with guidance on how to use the interface effectively. This may include tooltips, help icons, or contextual help that explains the purpose of each input field.

Iterative Design Process: Input design is often an iterative process, where designers create prototypes, gather feedback from users, and make improvements based on that feedback. This helps to ensure that the final input interface meets the needs of users effectively.

In summary, input design is a crucial aspect of system development that focuses on creating interfaces that allow users to input data easily and efficiently. It requires a deep understanding of user needs and preferences, as well as a focus on usability and accessibility. A well-designed input interface can enhance user experience, improve productivity, and reduce errors.

5.2 UML DIAGRAMS

Unified Modeling Language (UML) is a versatile modeling language with the primary purpose of establishing a standardized method for visualizing system designs. Similar to blueprints in other engineering disciplines, UML diagrams are used to depict the structure and behavior of a system. It's important to note that UML is not a programming language but rather a visual language that aids in modeling, design, and analysis.

UML diagrams serve as graphical representations that help software engineers, business professionals, and system architects communicate and collaborate effectively. These diagrams provide a standardized approach to illustrate various aspects of a software system, including its architecture, design, and behavior. This standardization simplifies the understanding, communication, and documentation of software development projects.

5.2.1 USE CASE DIAGRAM

Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents(actors).A use case is basically a diagram representing different scenarios where the system can be used.A use case diagram gives us a high level view of what the system or a part of the system does without going into implementation details.

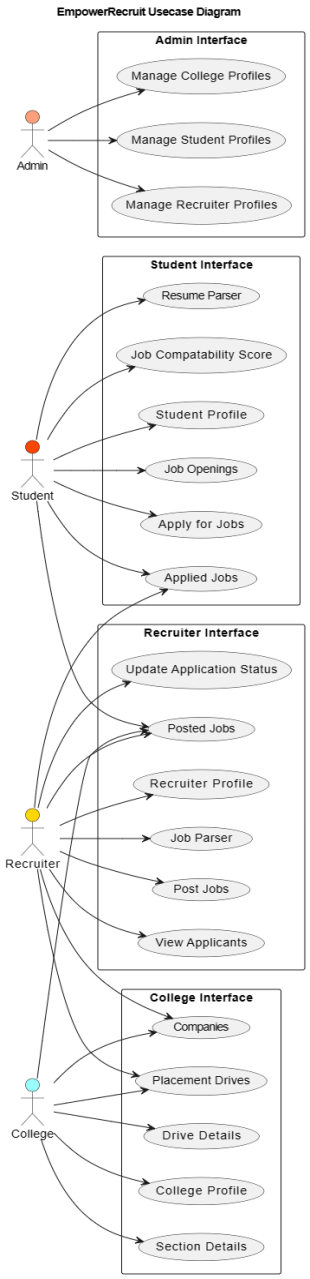


Fig 5.1: Use Case Diagram

5.2.2 CLASS DIAGRAM

The most widely use UML diagram is the class diagram. It is the building block of all object oriented software systems. We use class diagrams to depict the static structure of a system by showing system's classes, their methods and attributes. Class diagrams also help us identify relationship between different classes or objects.

Class notation is a graphical representation used to depict classes and their relationships in object-oriented modeling.

1. **Class Name:** The name of the class is typically written in the top compartment of the class box and is centred and bold.
2. **Attributes:** Attributes, also known as properties or fields, represent the data members of the class. They are listed in the second compartment of the class box and often include the visibility (e.g., public, private) and the data type of each attribute.
3. **Methods:** Methods, also known as functions or operations, represent the behavior or functionality of the class. They are listed in the third compartment of the class box and include the visibility (e.g., public, private), return type, and parameters of each method.

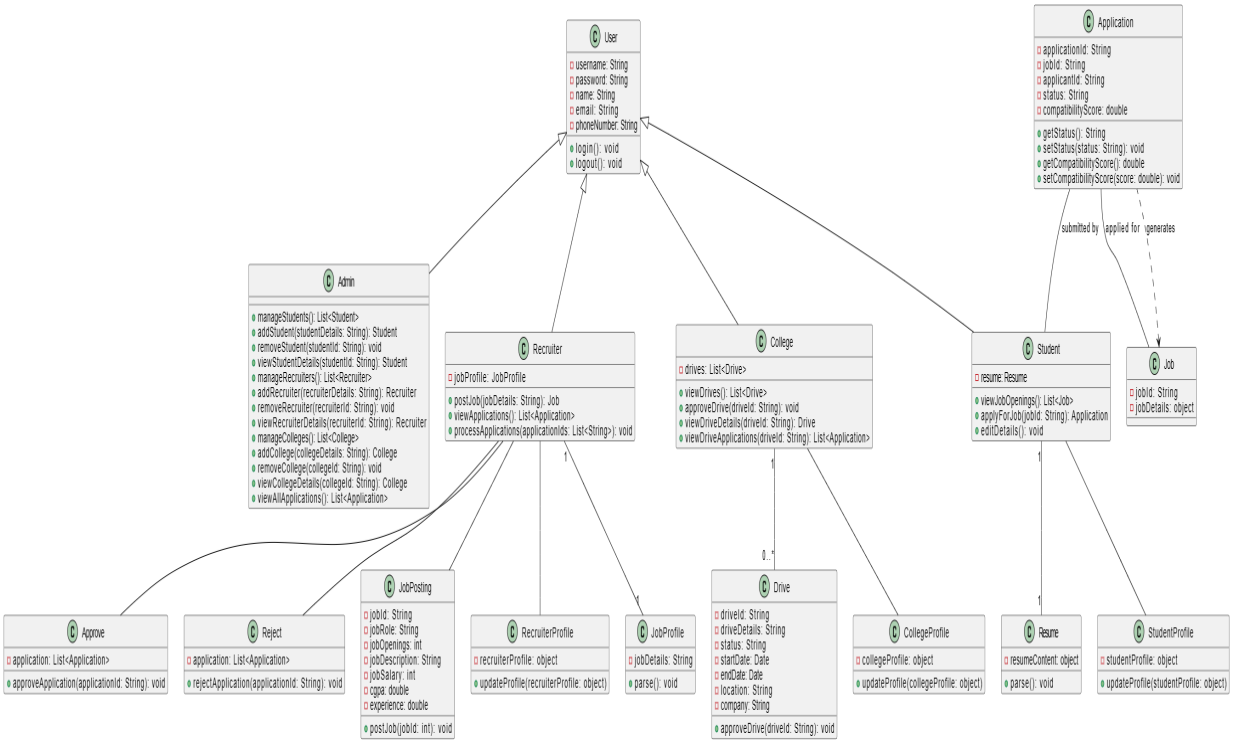


Fig 5.2: Class Diagram

5.2.3 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram in UML that shows how objects interact in a particular scenario of a use case. It represents the flow of messages between objects and the order in which these messages are sent and received. Sequence diagrams are used to visualize the dynamic behavior of a system, showing the interactions between objects over time. They are valuable for understanding the flow of control and collaboration between objects in a system and are commonly used during the design and analysis phases of software development.

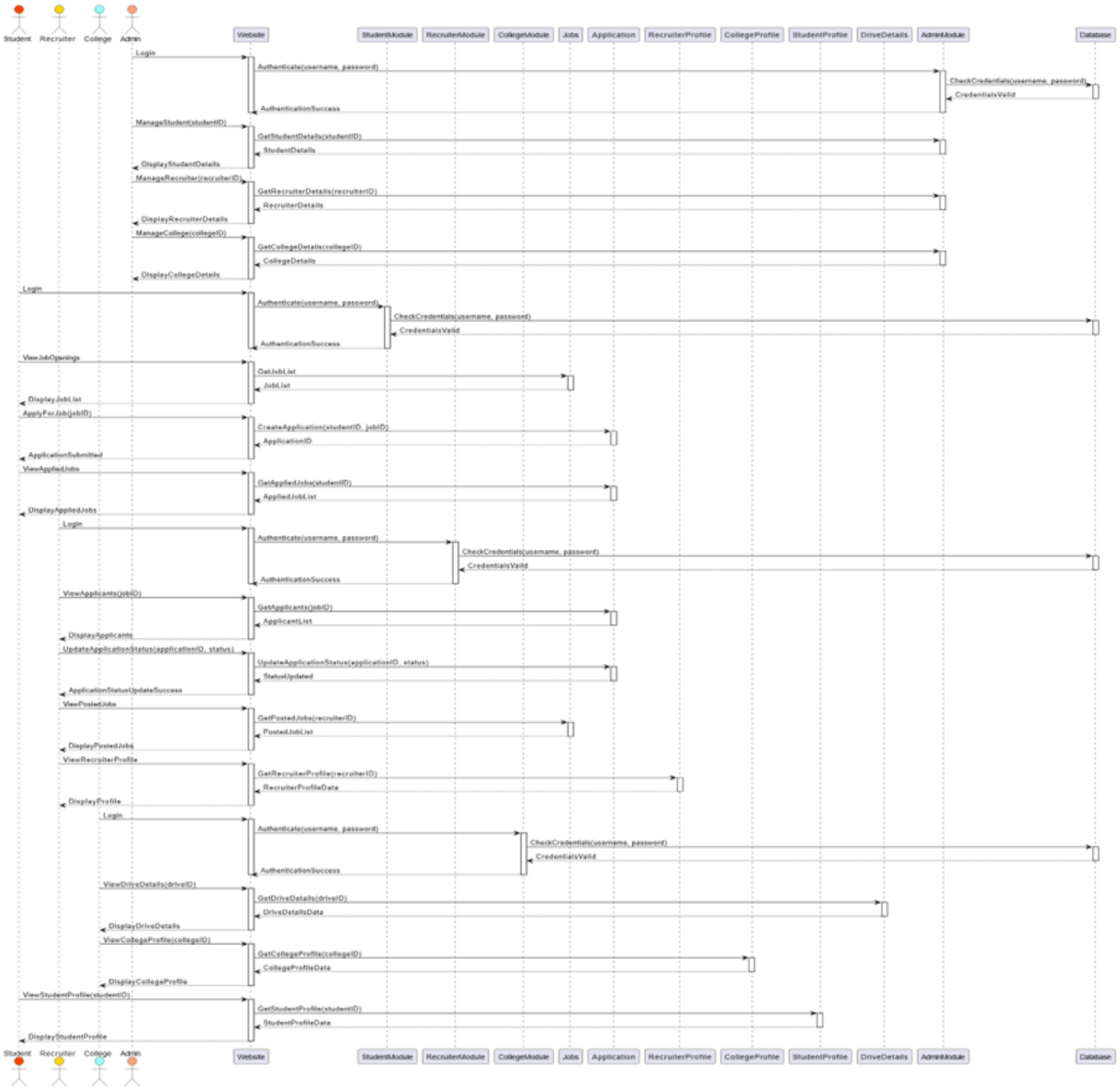


Fig 5.3: Sequence Diagram

5.2.4 STATECHAT DIAGRAM

A state chart diagram, a type of behavioral diagram in UML, visually represents the dynamic behavior of a system or a part of it. It illustrates the different states that an object or component can assume, along with the transitions between these states triggered by external events. This modeling tool is valuable for capturing the lifecycle of an entity within a system, offering a clear depiction of its behavior over time. State chart diagrams are extensively used in software engineering to model and understand complex system behaviors, aiding developers in designing and implementing software systems more effectively.

By mapping out the states and transitions of objects or components, state chart diagrams provide a structured approach to understanding the behavior of a system. They help developers visualize how entities within the system respond to various events and stimuli, leading to better-designed systems with more robust behavior. State chart diagrams are instrumental in improving the overall quality of software systems by enabling developers to identify potential issues early in the design phase and ensure that the system behaves as intended under different scenarios.

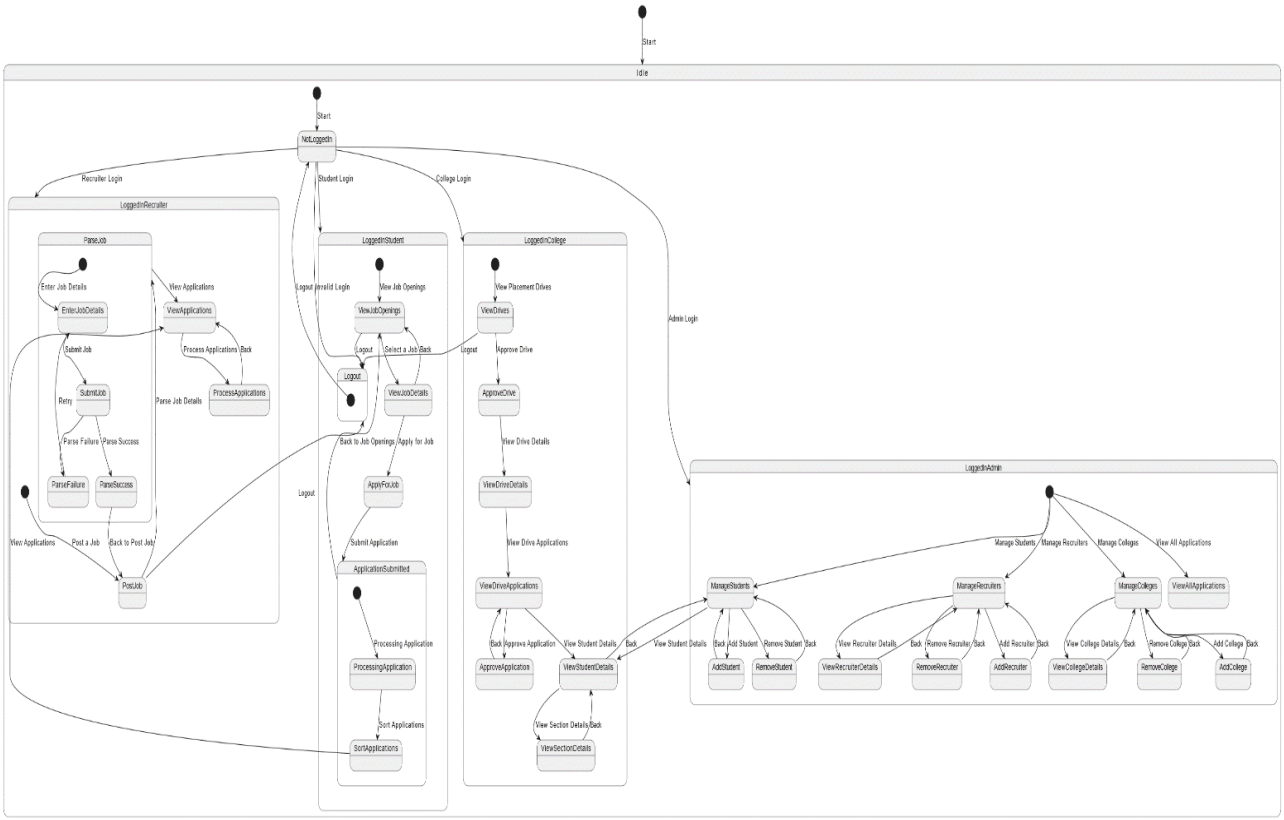


Fig 5.4: State Chart Diagram

5.2.5 DEPLOYMENT DIAGRAM

A deployment diagram in UML represents the physical deployment of software components and artifacts in a system. It illustrates the relationships and dependencies between software elements and the hardware components they run on, such as servers, PCs, or other devices. Nodes in a deployment diagram represent these hardware components, while components represent the software artifacts. Arrows between nodes and components indicate the deployment of artifacts on specific nodes. This diagram provides a clear visualization of how the software is distributed across hardware nodes, aiding in understanding the system's architecture and deployment configuration.

Deployment diagrams are beneficial for both developers and system administrators. Developers can use them to plan the deployment of their software, ensuring that components are distributed efficiently and that dependencies are managed correctly. System administrators can use deployment diagrams to understand the overall system architecture, plan hardware resource allocation, and manage software deployments effectively. Overall, deployment diagrams help in visualizing the deployment process and ensure that the software system is deployed correctly and runs smoothly in its intended environment.

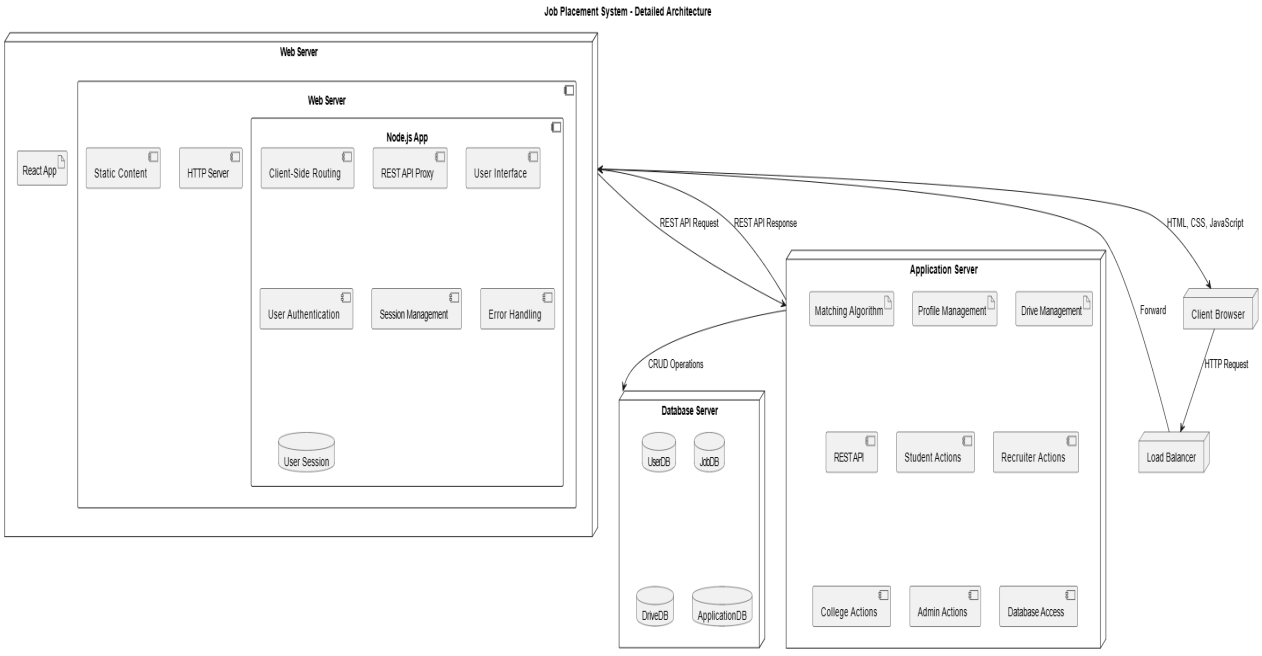


Fig 5.5: Deployment Diagram

5.2.6 ACTIVITY DIAGRAM

An activity diagram in UML is a behavioral diagram that represents the flow of activities in a system or a part of a system. It shows the sequence of activities and the flow of control from one activity to another, including decision points, branching, parallelism, and synchronization. Activity diagrams are useful for modeling business processes, workflows, and the logic of complex algorithms. They provide a visual representation of how activities are coordinated and executed in a system, helping stakeholders to understand, analyze, and improve the processes within the system.

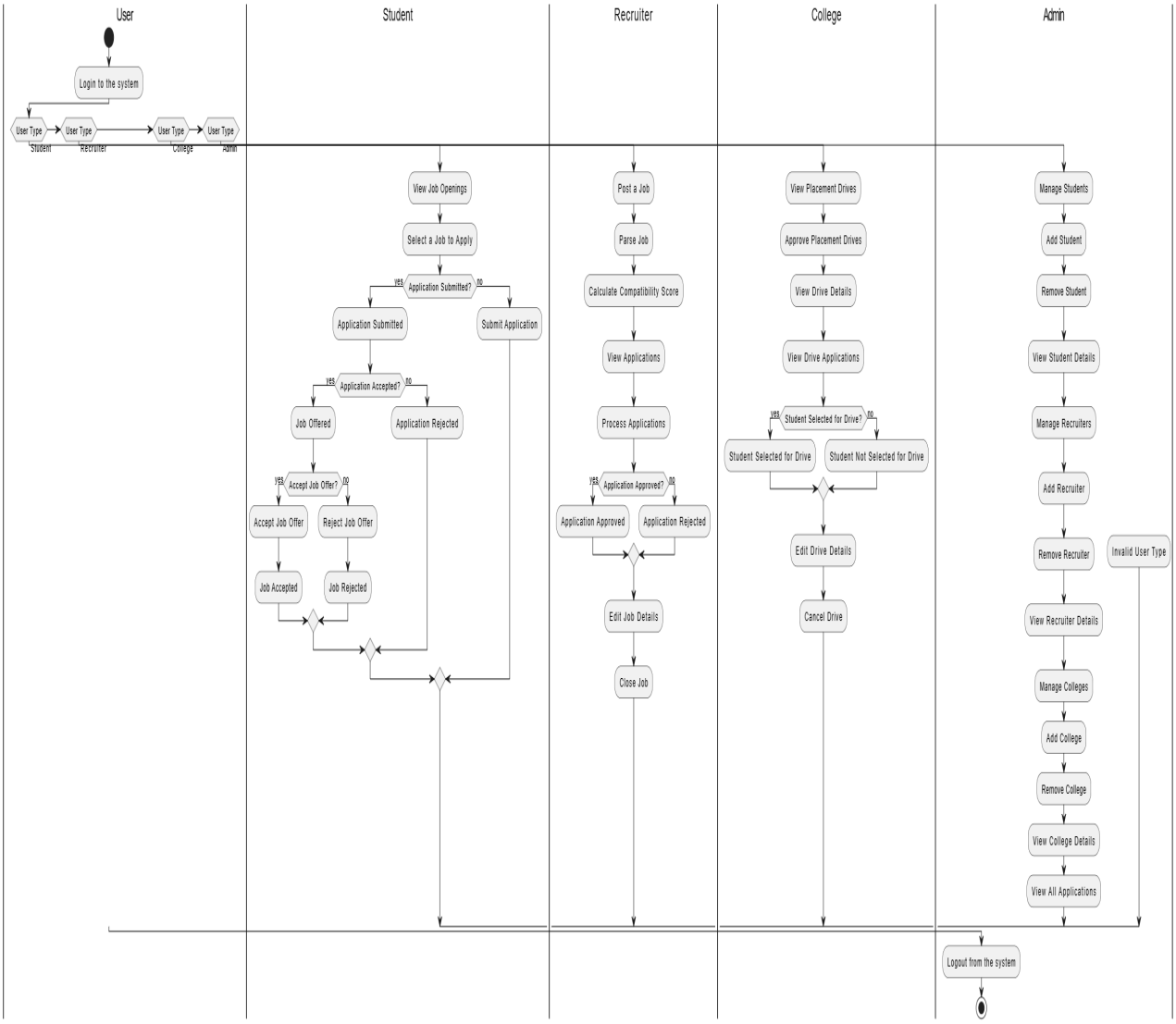


Fig 5.6: Activity Diagram

5.2.7 COMPONENT DIAGRAM

A component diagram in UML is used to model the physical components of a system and their dependencies. It shows how software components are connected and interact at runtime to achieve a specific functionality. Components represent modular parts of the system, such as classes, packages, files, and libraries, while the connections between them represent the dependencies and interactions between these components. Component diagrams are useful for understanding the structure of a system, identifying reusable components, and planning the development and deployment of software systems. They are commonly used during the design and architecture phases of software development to visualize the organization of components and their relationships in a system.

This aspect of component diagrams is particularly useful for understanding how the system's software components are distributed across a network of computers or devices. By showing the relationship between components and nodes, developers and system architects can gain insights into the system's physical architecture and how components interact across different parts of the system. This helps in optimizing the deployment of components for better performance, scalability, and reliability. Overall, component diagrams play a crucial role in the design and development of complex software systems, providing a visual representation of the system's structure, behavior, and deployment.

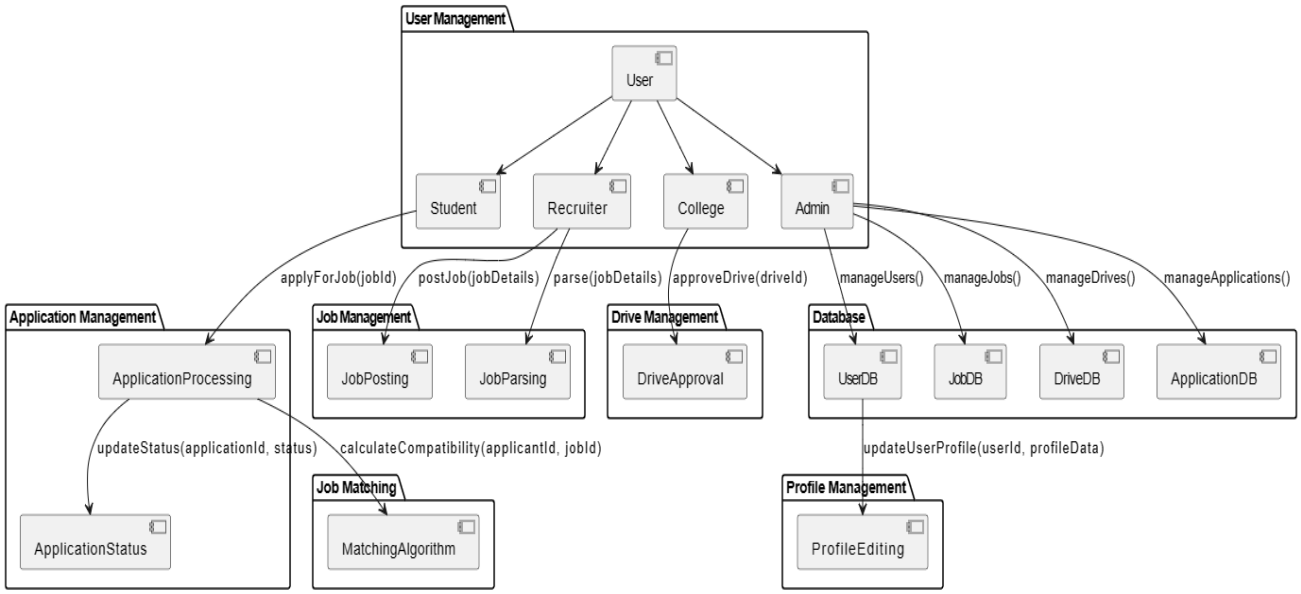


Fig 5.7: Component Diagram

CHAPTER 6

RESULTS

User Interface with Browse and Detect button. Browse is used to upload the image to be detected. Detect is used to identify humans in the given photograph.

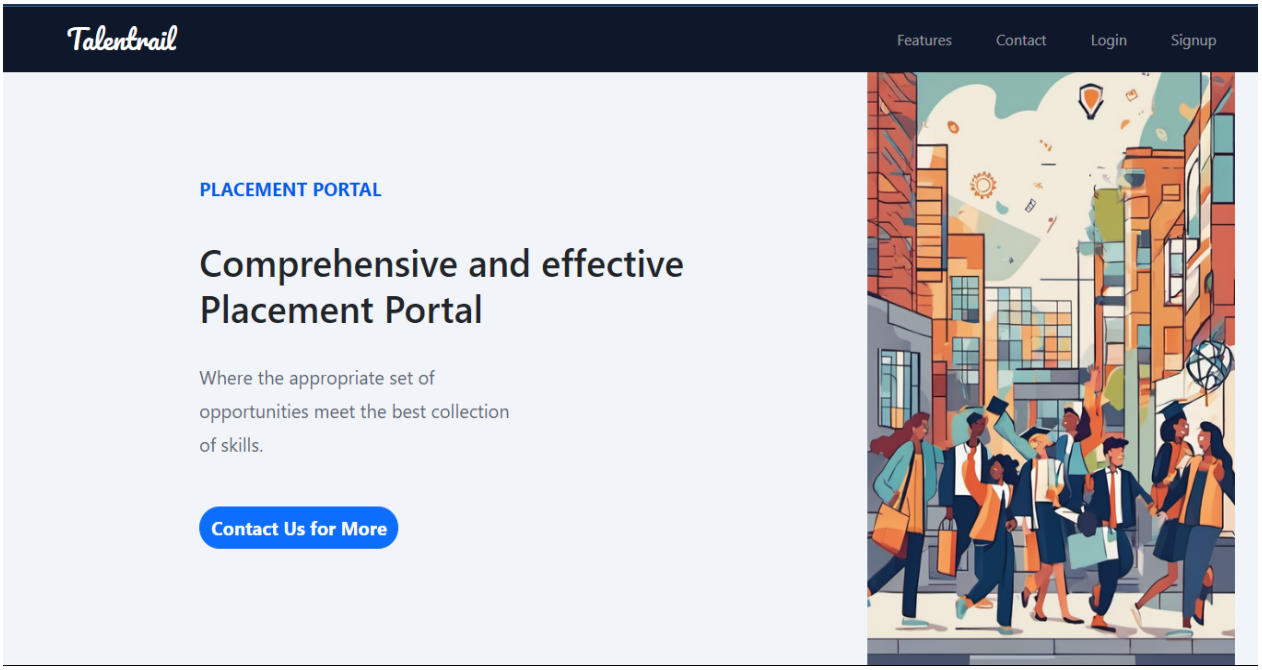


Fig 6.1: Home Page

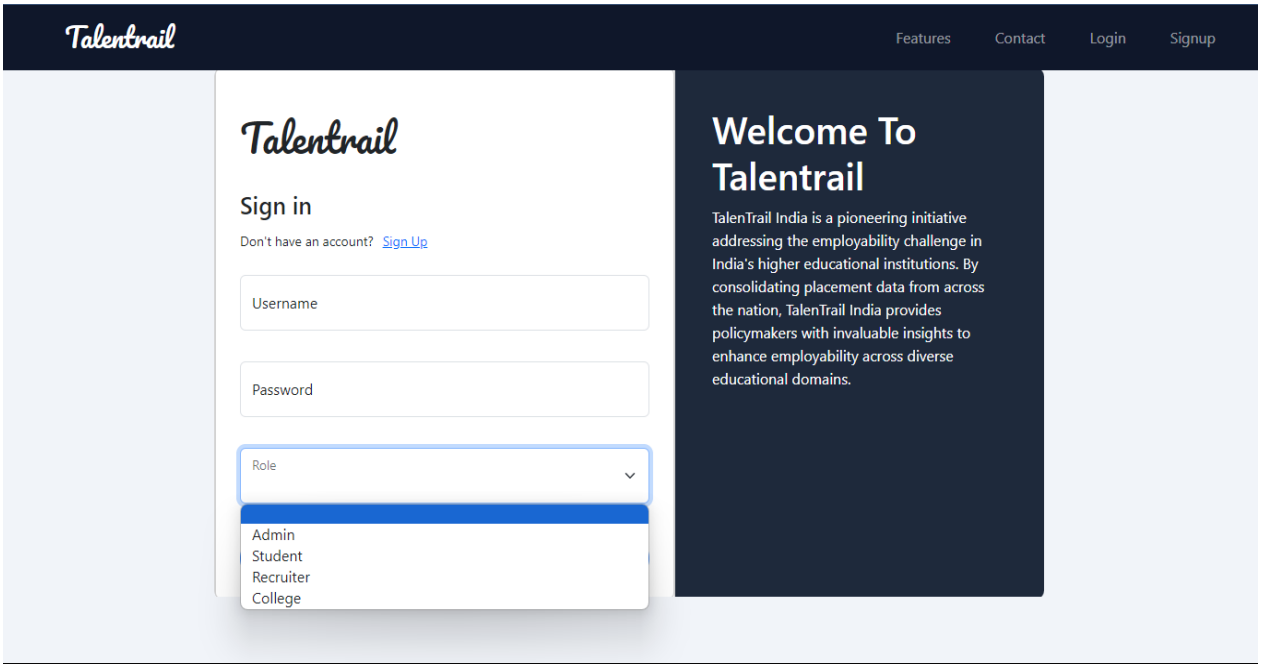


Fig 6.2: Login Page

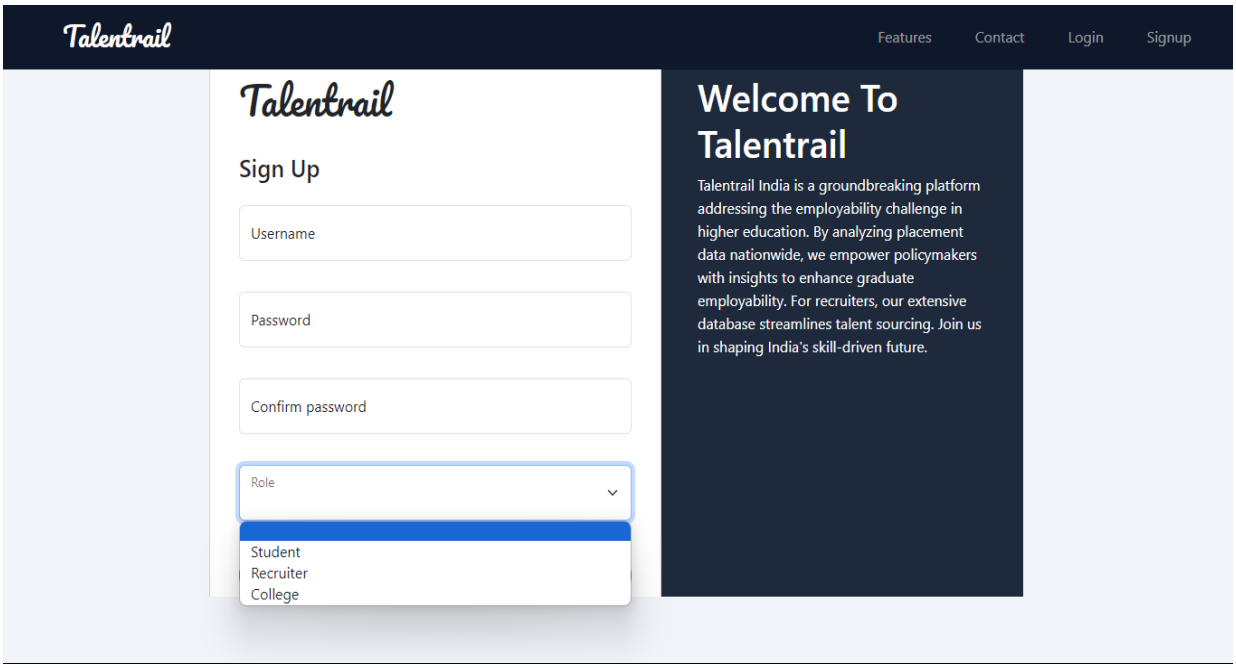


Fig 6.3: Sign Up Page

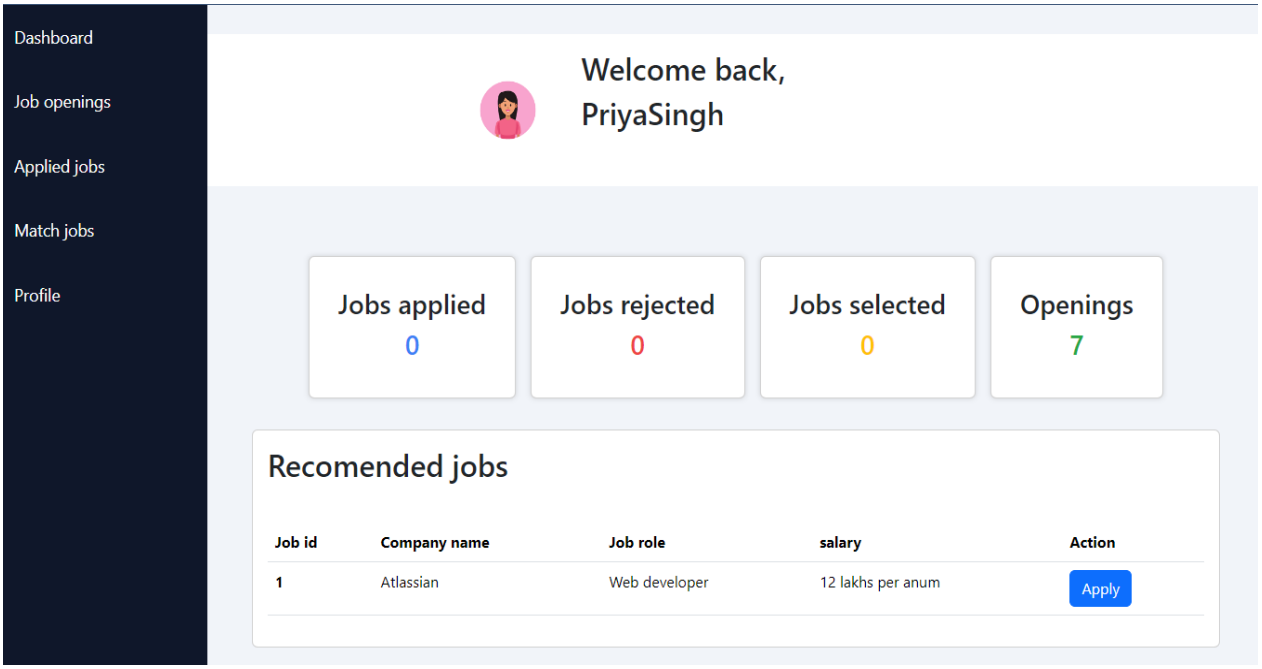


Fig 6.4: Student Dash Board

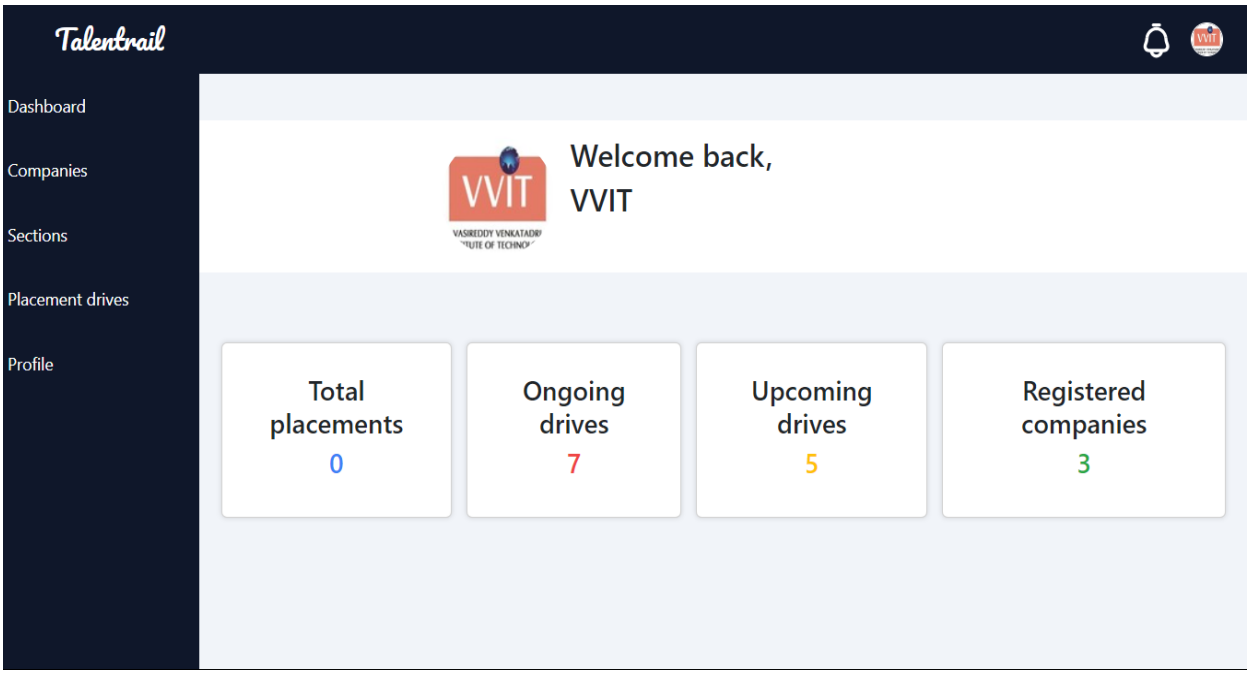


Fig 6.5: College Dash Board

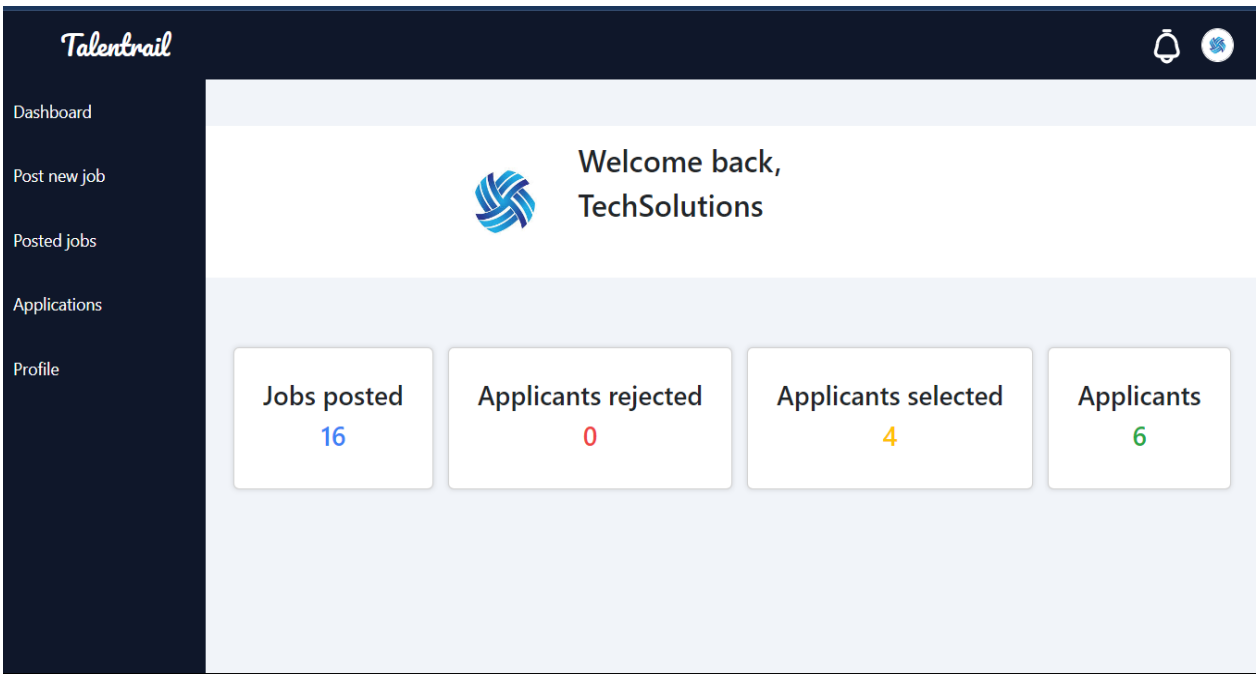


Fig 6.6: Recruiter's Dash Board

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

The study highlights how advanced technologies, especially Natural Language Processing (NLP), can transform traditional recruitment methods. It presents a detailed framework involving techniques like resume parsing with Hugging Face, NLP for job description analysis, and cosine similarity for candidate-job matching. Through extensive testing with various algorithms like Named Entity Recognition (NER) and models such as BERT , the framework proved to be effective and scalable. Resume parsing extracted relevant information from different resume styles seamlessly, while NLP successfully analyzed job descriptions to identify essential qualifications. Using cosine similarity for matching ensured candidate profiles aligned with job requirements. Although specific accuracy metrics were not the focus, the methodology's credibility and flexibility were evident. This research contributes recruitment to discussions practices, on modern highlighting the importance of technology-driven solutions for improving efficiency, accuracy, and inclusivity in talent acquisition. Future research opportunities include exploring new NLP algorithms, enhancing matching strategies, and integrating with emerging technologies for better performance in dynamic recruitment environments. Overall, the study emphasizes the need for innovative approaches to adapt to changing recruitment trends.

CHAPTER 8

REFERENCES

[1] E. Albaroudi, T. Mansouri, and A. Alameer, "A Comprehensive Review of AI Techniques for Addressing Algorithmic Bias in Job Hiring," AI, vol. 5, no. 1, pp. 383–404, Feb. 2024, doi: 10.3390/ai5010019.

[2] B. Lund, "Researchers Matching for Collaboration: A Novel Algorithm Based Approach Using Cosine Similarity," SSRN Electronic Journal, 2023, Published,

[3] "TRANSFORMING HR PRACTICES: RESUME RANKING USING BERT EMBEDDINGS," International Research Journal of Modernization in Engineering Technology and Science, Nov. 2023, Published, doi: 10.56726/irjmets45589.

[4] L. Kadri, "ROLE WISE JOB DESCRIPTION MAPPING & ITS USEFULNESS IN RECRUITMENT," International Journal of Management, Public Policy and Research, vol. 2, no. 4, pp. 39–44, Oct. 2023, doi: 10.55829/ijmpr.v2i4.187.

[5] V. Mittal, P. Mehta, D. Relan, and G. Gabrani, "Methodology for resume parsing and job domain prediction," Journal of Statistics and Management Systems, vol. 23, no. 7, pp. 1265–1274, Jul. 2020, doi: 10.1080/09720510.2020.1799583.

[6] A. K. Sinha, M. A. K. Akhtar, and M. Kumar, "Automated Resume Parsing and Job Domain Prediction using Machine Learning," Indian Journal Of Science And Technology, vol. 16, no. 26, pp. 1967–1974, Jul. 2023, doi: 10.17485/ijst/v16i26.880.

[7] I. Journal, "THE SCIENCE OF HIRING : Predicting Personality through CV Parsing & Machine Learning," INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT, vol. 07, no. 11, pp. 1 11, Nov. 2023, 10.55041/ijsrem27060. doi:

[8] J. Quan, "A Corpus-based Analysis of Job Description Discourse," International Journal of Linguistics and Translation Studies, vol. 4, no. 4, pp. 144–158, Oct. 2023, doi: 10.36892/ijlts.v4i4.382. doi: 10.2139/ssrn.4346965.

APPENDIX

E - Certificate

CERTIFICATE
OF PUBLICATION

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Referred, Multidisciplinary, Scholarly Indexed, Open Access Journal since 2012)

IJIRSET

Impact Factor 8.423

The Board of IJIRSET is hereby awarding this certificate to

RANKELA SAI SRI HARSHA

Department of CSE (Artificial Intelligence & Machine Learning), Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh, India

In Recognition of publication of the paper entitled

"Resume Parsing Using Named Entity Recognition and Hugging Face Model"

in IJIRSET, Volume 13, Issue 3, March 2024

Google scholar | Crossref | Mendeley | doi | INNO SPACE SJIF Scientific Journal Impact Factor

e-ISSN: 2319-8753
p-ISSN: 2347-6710

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

P. Kumar
Editor-in-Chief

www.ijirset.com ijirset@gmail.com

CERTIFICATE
OF PUBLICATION

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Referred, Multidisciplinary, Scholarly Indexed, Open Access Journal since 2012)

IJIRSET

Impact Factor 8.423

The Board of IJIRSET is hereby awarding this certificate to

BUSI JOSEPH

Department of CSE (Artificial Intelligence & Machine Learning), Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh, India

In Recognition of publication of the paper entitled

"Resume Parsing Using Named Entity Recognition and Hugging Face Model"

in IJIRSET, Volume 13, Issue 3, March 2024

Google scholar | Crossref | Mendeley | doi | INNO SPACE SJIF Scientific Journal Impact Factor

e-ISSN: 2319-8753
p-ISSN: 2347-6710

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

P. Kumar
Editor-in-Chief

www.ijirset.com ijirset@gmail.com

CERTIFICATE
OF PUBLICATION

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Referred, Multidisciplinary, Scholarly Indexed, Open Access Journal since 2012)

IJIRSET

Impact Factor 8.423

The Board of IJIRSET is hereby awarding this certificate to

VARA LAKSHMAIAH DUGGINENI

Department of CSE (Artificial Intelligence & Machine Learning), Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh, India

In Recognition of publication of the paper entitled

"Resume Parsing Using Named Entity Recognition and Hugging Face Model"

in IJIRSET, Volume 13, Issue 3, March 2024

Google scholar | Crossref | Mendeley | doi | INNO SPACE SJIF Scientific Journal Impact Factor

e-ISSN: 2319-8753
p-ISSN: 2347-6710

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

P. Kumar
Editor-in-Chief

www.ijirset.com ijirset@gmail.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 3, March 2024 ||

| DOI:10.15680/IJIRSET.2024.1303223|

Resume Parsing Using Named Entity Recognition and Hugging Face Model

Rimmalapudi Rajesh, Rankela Sai Sri Harsha, Busi Joseph, Vara Lakshmaiah Duggineni, Janardhana Rao Alapati

Department of CSE (Artificial Intelligence & Machine Learning), Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh, India

Asst. Professor, Department of CSE (Artificial Intelligence & Machine Learning), Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh, India

ABSTRACT: New methods in networking and communication have paved the way for enhancing the recruitment process by creating e-recruitment recommender systems. The growing popularity of online recruitment has led to a high volume of resumes being saved in recruitment platforms. Resumes are typically designed in various styles to grab the attention of recruiters, using different sizes, colors, and table formats. Nevertheless, the diversity of formats significantly impacts data mining tasks like extracting resume information, matching profiles automatically, and ranking applicants. Rule-based, supervised, and semantics-based techniques have been developed for precise extraction of resume facts, but they rely heavily on extensive annotated data, which can be challenging to gather. Moreover, the time-consuming nature of these techniques and the incomplete knowledge they possess greatly impact the accuracy of resume parsing. In this manuscript, we introduce a framework for parsing resumes that addresses the constraints encountered in previous methods. Initially, the unprocessed content is taken from CVs and sections are divided by classifying text blocks. Moreover, the entities are identified using named entity recognition and enhanced using Hugging Face. The resume parser proposed accurately grabs details from resumes which directly help in selecting the top candidate.

KEYWORDS: Resume parsing, Data Enrichment, Text Extraction, Hugging Face, Boolean Naive Bayes

I. INTRODUCTION

Recent advancements in networking and communication have led to increased information exchange on the internet, with job seekers using online job boards to share their resumes, resulting in a large number of job postings and CVs online. A study found that over one million resumes are submitted annually to platforms like Monster.com and LinkedIn.com. Recruiters typically spend around 2 minutes reviewing each resume, leading to the quick scanning for key information and potential oversight of important details. This has driven the development of e-recruitment recommender systems, which have become the primary method of hiring in many industries. While these systems have helped streamline the hiring process and reduce time and costs, they also pose challenges such as candidates applying for unsuitable roles due to the limitations of parsing resumes accurately.

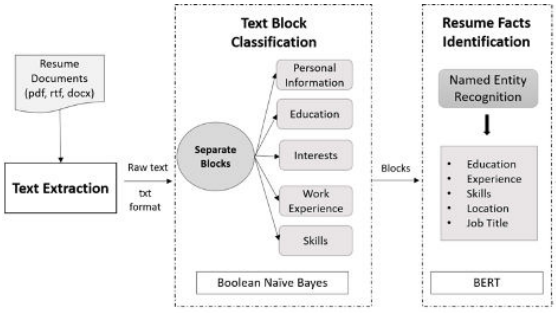


Fig. 1: Proposed Framework for Resume Parsing

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 3, March 2024 ||

| DOI:10.15680/IJIRSET.2024.1303223|

Resume parsing, crucial for effective hiring, encounters challenges due to varying formats (PDF, docx, rtf) and layouts (list-style, table-style) with elements like info-graphics requiring image processing. This complexity makes candidate recommendation parsing crucial for evaluation. Machine parsing struggles due to complexity, prompting research into plain text, heuristic, and neural network methods. Gathering data across domains remains a focus, despite research gaps in layout details and domain knowledge. This study introduces a framework converting unstructured resumes into structured, data-enriched outputs. It employs Named Entity Recognition and a domain-neutral ontology to enhance skill entities, improving candidate profile matching and ranking.

II. LITERATURE REVIEW

Research has long been centered around resume parsing. Various methods have been developed for precise information extraction and are categorized as follows.

A. Heuristics-Based Methods:

Chen et al. [7] introduced a two-phase algorithm for extracting data from unformatted resumes, leveraging the 'Writing Style' approach, Naive Bayes, and NER. Gunaseelan et al. [5] developed a machine learning-based system for segment extraction from resumes, employing two categorization methods. Despite successes, challenges persisted in handling diverse document structures. Additionally, Chen et al. [12] proposed a hierarchical method for PDF resume extraction, utilizing heuristic rules and a CRF model. While effective for list-formatted resumes, it lacked compatibility with table-formatted ones and primarily focused on specific resume sections like personal details and education.

B. Framework Based Methods:

A different method for parsing resumes and matching them with job profiles was suggested by Deepak et al [13]. The system for parsing resumes had 4 stages: text segmentation, NER, merging co-references and conflict resolution, and text normalization. Initially, the resume was divided into sections containing related data and then named entities were categorized using NER. Additionally, text normalization was carried out to ensure that the extracted entities were consistent and trustworthy. Finally, o-reference resolution was employed to clarify the abbreviations in the resume and examine their linguistic expressions. The resume parser's results were utilized to rank resumes based on a specific job description.

C. Skill Extraction Methods:

Various researchers have focused on skill extraction in resume parsing, particularly aiming to identify implicit skills. Kameni and NoHugging Face [14] devised a method utilizing a CNN-based multi-label classification model to predict high-level skills from unprocessed CV content. They transformed the content into matrices and employed binary CNN classifiers to categorize resumes into occupation classes. Similarly, Chifu et al. [12] proposed an ontology-driven system to extract professional skills from natural language resumes. Their method incorporated pattern induction and skill detection modules to identify unfamiliar skills mentioned in resumes. Additionally, they addressed the ambiguity of skill phrases using Wikipedia..

D. Deep Learning Based Methods:

Zu and Wang [6] proposed a text block classification method for precise resume information extraction. Utilizing neural network-based text classifiers, their approach focused on accurately categorizing text blocks and lines within them. They introduced a novel block segmentation method leveraging line details and word representations. This paired with line type and label classification facilitated precise text block categorization. Meanwhile, Nguyen et al. [14] introduced a four-phase method for resume data extraction. It involved text segmentation, rule-based NER entity extraction, and DNN-based entity sequencing. Despite effectiveness, limited training data hindered DNN performance. Additionally, the study targeted IT resumes, potentially impacting applicability across different fields.

III. PROPOSED FRAMEWORK

This part discusses the input, output, and thorough explanation of the suggested resume parsing framework. Fig.1 displays the process of parsing resumes.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 3, March 2024 ||

| DOI:10.15680/IJIRSET.2024.1303223|

A .Input and Output:

This pipeline aims to extract information from resumes without taking into account the document's font styles, font sizes, or standard structure.

1. Input : The resume parser takes in unstructured documents of various file types such as pdf, docx, and doc files.
2. Output : Output is presented in a structured format detailing all relevant information and details about a candidate obtained from their resume.

B . Text Extraction:

Resumes exhibit diverse formats, including document type, layout, writing style, and structure, attributed to the lack of standardized formats. To enable text mining techniques, raw text extraction from resumes is necessary. The proposed method utilizes PDFbox to retrieve unprocessed text from PDF resumes, as depicted in Fig. 2. However, this approach is limited by PDFbox's inability to extract text from non-PDF document formats, necessitating conversion to PDF before extraction.

Fig. 2: Conversion of PDF file to Txt file

1) Data Preprocessing: The gathered information undergoes preprocessing to eliminate noise introduced by PDFbox, such as non-ascii characters, unnecessary punctuation, missing and extra spaces, and newlines.

i) Missing or Extra Spaces: Because of the varied font styles and sizes found in resumes, the extracted words may have additional spaces or lack spaces altogether. For example, if spaces are missing, 'I am a competent person' will become 'Iamacompetentperson' or 'I a m a c o m p e t e n t p e r'. In the event of additional spaces. This creates disturbances in the text and is removed prior to classifying text blocks.

ii)*Punctuation Marks:* The extracted text was full of punctuation marks, particularly at the start and in the middle. They create disturbance in the data by altering the meaning of a word which hinders the process of extracting information. For example, Adobe Illustrator and Adobe Illustrator are considered the same skill, but are recognized as two distinct words by the model. It is eliminated during data-preprocessing to prevent bias in similar types of text.

iii) *Non-Ascii characters:* PDFbox produces certain non-ascii characters in the resulting text document when extracting text in regards to bullet points, emoticons, or icons. Before the text block classification training, any non-ascii characters in the extracted text are eliminated. An instance of non-ascii characters being produced during text extraction can be seen in Figure 3.

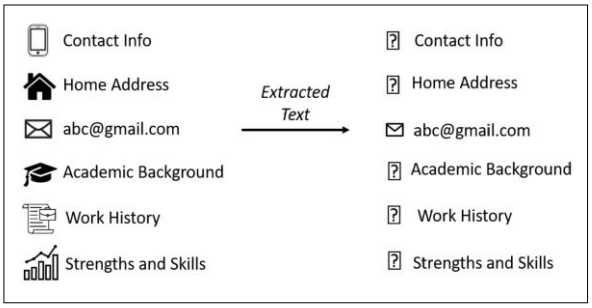


Fig. 3: Ascii characters generation

C . *Text Block Classification:*

Resume parsing relies heavily on categorizing text blocks, crucial for subsequent modules. Resumes typically follow a hierarchical structure, with blocks encompassing diverse topics like personal details, employment history, skills, and

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 3, March 2024 ||

| DOI:10.15680/IJIRSET.2024.1303223|

interests. Each block contains specific information related to its category, such as location in personal details or educational background in academic history. This study focuses on five key categories, including academic background and skills, as indicators of an individual's qualifications. While both machine learning and deep learning methods are employed for block categorization, neural network models excel in automatic feature identification and pattern recognition. However, substantial data is essential for effective neural network training.

The pipeline we suggest uses the Boolean Naive Bayes (BNB) algorithm, a probabilistic model, to categorize sections of resumes. The text data that has been pre-processed is utilized to create the vocabulary for the five specified information fields, which are then provided as input to the text block classification module.

1)Boolean Naive Bayes Alogorithm: BNB is a probabilistic model that is supervised and makes classifications based on the highest probability from the possible classes. Moreover, it eliminates duplications in the text prior to determining probabilities. It relies on Bayes' theorem assuming that features are independent, meaning each feature is not related to any other feature when considering the class label. The calculation of the probability for each class in Naive Bayes is shown in equation (1) .

$$\hat{C} = argmax_{c \in C} \frac{P(x|c) \times P(c)}{P(x)} \quad (1)$$

P(x|c) is the likelihood probability of variable z given class c and P(c) is the prior probability of a certain class c among the set of classes C where C = c_1 ,c_2,....,c_n.

2) BNB in resume parsing: The method involves training 5 language models for 5 classes (education, skill, experience, interest, and personal information) using the multinomial boolean naive bayes algorithm. Each model categorizes particular data sections found in the resumes. Table I provides the specifics of these information blocks.

TABLE I: **Details of Information Blocks**

Models	**Information Block**
Education Model	Academic Qualification, Certifications, Publica- tions
Experience Model	Work History, Professional Experience, Organi- zation, Responsibilities
Skills Model	Linguistic Skills, Technical Skills, Soft Skills, Computer Skills
Interests Model	Hobbies, Interests, Extra Curricular Activities
Personal Info Model	Personal Data, References

3) *Training:* The goal of BNB training is to create five language models using the five different vocabularies - education, skills, personal information, experience, and interests - produced during data-preprocessing, as illustrated in Fig. 4.

All of these words are combined together to create one comprehensive vocabulary V. Each word w in the vocabulary has a conditional probability P(w/c) computed for a specific class c, with this process being repeated for all classes C. The probability of a word given a certain class P(w/c) is determined using Laplace Smoothing (LS) formula instead of a simple count ratio as demonstrated in Eq (2). LS has the benefit of ensuring that no word will be assigned a zero probability for any particular class. If a word is not found in the vocabulary of a specific class c out of the five classes

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 3, March 2024 ||

| DOI:10.15680/IJIRSET.2024.1303223|

C, a small probability is still assigned to avoid having a total zero probability during testing. The probability of each word in the vocabulary is calculated conditionally as follows:

$$P(w_i|c) = \frac{count(w_i, c) + 1}{count(w, c) + |V|} \quad (2)$$

Count of a specific word in a class's vocabulary is represented as count(w_i , C), while the total number of words in the vocabulary of that class is represented as count(w, c). |V| signifies the total number of words in the overall vocabulary.

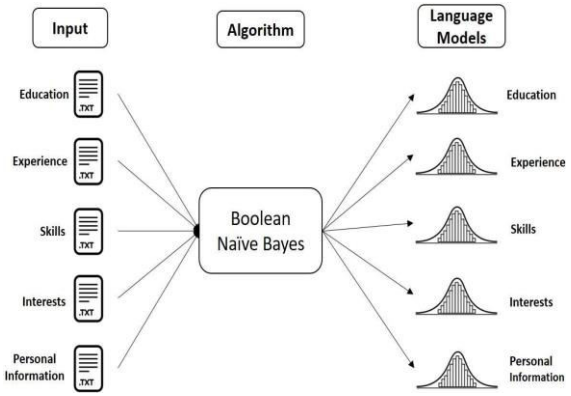


Fig. 4: Training of Boolean NB Algorithm

A probability is computed for the given word in the vocabulary for each class and saved in its corresponding model. This leads to the creation of 5 models of the same size as the shared vocabulary, each with unique probabilities.

4) Testing: Initially, the unprocessed text from a potential candidate's resume is extracted and prepared in the same manner as described in the previous section. Later, each section in the resume is categorized into one of the five classes. At first, the BNB algorithm eliminates repeated words in each block as required. Five language models are used to determine the conditional probability P(w|c) of each unique word w in a given block b to be classified.

Algorithm 1 Test BooleanNB(C, t, $prior$, $condprob$, B)

Input:
C: Set of classes
t: All words in a single block
B: Set of blocks in a resume
$prior$: Prior probability of a certain class
$condprob$: Conditional probability of a word given a certain class

procedure BNBTest(C,t,$prior$,$condprob$,B)
for each $b \in B$ **do**
  $W \leftarrow ExtractUniqueWordsfromBlock$(t,b)
  for each c $\in C$ **do**
    $score[c] \leftarrow log(prior[c])$
    for each $w_i \in W$ **do**
      $score[c]$ += $log(condprob[w_i][c])$
  Ĉ ←argmax$_{c \in C}$ $score[c]$
  Assign label Ĉ to b

Output:
Set of class labels assigned to B

Moreover, the likelihood probability for a given block is calculated by multiplying the conditional probabilities of each unique word in the block. It is then combined with the previous probability P(c) of a specific class e in order to

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 3, March 2024 ||

| DOI:10.15680/IJIRSET.2024.1303223|

calculate the ultimate probability score prevent a specific group from taking action. Nevertheless, the issue of underflow arises due to the multiplication of probabilities. The log adjusted formula is employed to determine the ultimate probability score of a specific block for each model in order to address this limitation. Furthermore, the block is assigned the class with the highest probability.

D. *Resume Facts Identification:*

Following the classification of the text blocks, distinct blocks are generated for personal details, educational history, skills, job experience, and hobbies. NER utilizes them to identify the named entities in the text passages. NER is a crucial component of the process of extracting information, aiming to identify, extract, and categorize named entities in text into specific groups . The chosen entities for this research project are:

- Education - title of degree and area of study for the applicant on their resume
- Experience - duration a candidate was employed at a particular job and name of the organization
- Skills - the technical abilities listed on the CV
- Job Title - candidate's most recent job title
- Location - the job seeker's place of location

Initially, NER methods heavily depended on rule and dictionary-based strategies, which required rule sets created by experts in specific fields. Nevertheless, the regulations do not encompass every instance of language use, the creation procedure is excessively lengthy, and is less probable to be transferable . Lately, the progress of deep learning in numerous Natural Language Processing (NLP) activities has sparked a transformation in the information extraction field, obtaining state-of-the-art outcomes without requiring manually created features. Additionally, transfer learning has proven to be successful in various NLP tasks by leveraging pre-trained models to apply previously acquired knowledge to tasks specific to a particular domain. HUGGING FACE is a language model that utilizes transformer architecture and focuses on context. Positional embedding is utilized in lieu of positional encoding to produce the contextualized embedding of a full sentence [11]. In this document, a HUGGING FACE model that has been fine-tuned is employed to identify and categorize the named entities described earlier. The pre-trained HUGGING FACE language model parameters are extensively trained with annotated data to encode information about our problem domain. Different types of data sets are utilized for training various sections of a resume to prevent class imbalance issues.

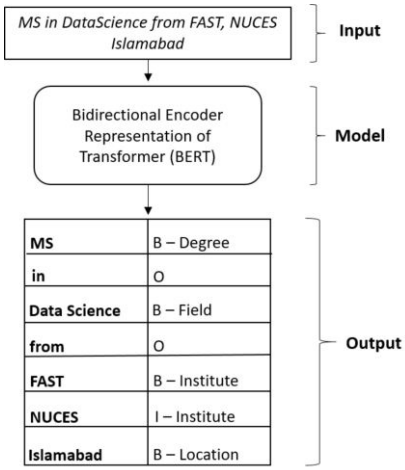


Fig. 5: Process of Named Entity Recognition

Following the adjustment of the HUGGING FACE model, the unstructured input goes through processing and is labeled with IOB tags as well as the tags employed during the training of the HUGGING FACE model. The outcome was a structured output where each word was labeled with the named entity, as depicted in Figure 5. IOB formatting, which stands for Inside, Outside, Beginning, is a format for tagging tokens that is commonly utilized in applications like NER. They are akin to part-of-speech tags, but they also provide details on where an entity is situated in the text.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 3, March 2024 ||

| DOI:10.15680/IJIRSET.2024.1303223|

IV. RESULT

The suggested resume parsing framework showed efficient performance in retrieving organized data from various resume styles and designs. Using the Boolean Naive Bayes algorithm for text block classification accurately sorted resume sections into categories such as education, skills, work experience, personal details, and interests. The categorization models, which were trained using specialized vocabularies for each category, successfully achieved a high level of accuracy in dividing the unorganized resume text into separate labeled sections that correspond to various types of information.

Using a fine-tuned HUGGING FACE model, the named entity recognition module successfully detected and extracted important entities such as degree names, organizations, skills, locations, and job titles from the categorized text blocks. Including a personalized skill ontology improved the identified skill entities by creating semantic connections and inferring additional skills not directly mentioned in the resume. This additional data enrichment step boosted the depth of the candidate skill profiles produced by the framework. In general, the outcomes confirmed the effectiveness of the resume parsing process in managing different resume formats and creating organized, enhanced candidate profiles that are appropriate for ranking and matching in electronic recruitment systems.

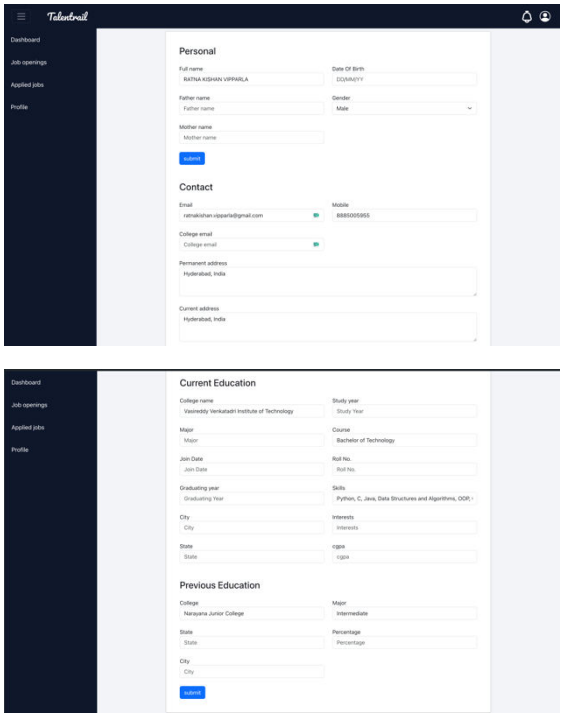


Fig. 6: Parsed Data

V. CONCLUSION

This paper introduces a framework for parsing resumes that can extract information from resumes with a variety of text formats and layouts. The objective of this study is to improve the precision of extracting information from individual resumes to help in selecting the most suitable candidate. In the suggested framework, the key processing steps include extracting text, classifying text blocks, and identifying resume facts using NER. Moreover, the skills that are taken out are enhanced with a specifically designed ontology. This framework streamlines the laborious task of manually creating custom features for text block classification and NER by saving time. The second addition is the method utilized for categorizing text blocks, involving the employment of Boolean Naive Bayes to categorize blocks into various classes. Additionally, individual NER training is conducted for each type of block to prevent class imbalance. Finally, the ultimate achievement is the creation of a personalized skill ontology for enhancing data. The focus is on various areas to encompass diversity and enhance reusability in the realm of e-recruitment. Plans for future work include

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 3, March 2024 ||

| DOI:10.15680/IJIRSET.2024.1303223|

incorporating an experiments and evaluation section to showcase results of the proposed framework using actual e-recruitment data.

REFERENCES

The template will number citations consecutively within the bracket

[l] W. Abdessalem and S. Amdouni, "E-recruiting supp011 system based on text mining methods," *International Journal of Knowledge and Learning,* vol. 7, pp. 220-232, 2011.

[2] E. K. Ryland and B. Rosen, "Personnel professionals' reactions to chronological and functional resume formats," *Career Development Quarterly,* vol. 35, no. 3, p. 228-238, 1987.

[3] S. Alotaibi and Y. Mourad, "Job recommendation systems for enhancing e-recruitment process," in *Proceedings of the International Conference on Ii1formatio11 and Knowledge Engineering,* 2012.

[4] "Analysis and shortcomings of e-recruitment systems: Towards a semantics-based approach addressing knowledge incompleteness and limited domain coverage," *Journal of lr!formation Science,* vol. 45, no. 12, p. 016555151881144, 2018.

[5] S. K. Kopparapu, "Automatic extraction of usable information from unstructured resumes to aid search," in *2010 IEEE International Con- ference on Progress in b(fonnatics and Computing,* vol. 1, pp. 99-103, 2010.

[6] S. Zu and X. Wang, "Resume information extraction with a novel text block segmentation algorithm," *Linguistics,* vol. 8, no. 5, pp. 29-48, 2019.

[7] J. Chen, C. Zhang, and Z. Niu, "A two-step resume information extraction algorithm," *Mathematical Problems in Engineering,* 2018.

[8] V. Bhatia, P. Rawat, A. Kumar, and R. R. Shah, "End-to-end resume parsing and finding candidates for a job description using bert," 2019.

[9] K. Yu, G. Guan, and M. Zhou, "Resume information extraction with cascaded hybrid model," in *Association for Computational Linguistics,*
p. 499-506, 2005.

[10] A. Singh, C. Rose, K. Yisweswariah, V. Chenthamarakshan, and
N. Kambhatla, "Prospect: a system for screening candidates for recruit- ment," in *Proceedings of the 19th ACM international conference on ltiformation and knowledge management,* p. 659-668, 2010.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL- HLT,* 2019.

[12] J. Chen, L. Gao, and Z. Tang, "Information extraction from resume documents in pdf format," in *Document Recognition and Retrieval,* 2016.

[13] G. Deepak, V. Teja, and A. Santhanavijayan, "A novel firefly driven scheme for resume parsing and matching based on entity linking paradigm," *Journal of Discrete Mathematical Sciences and Cryptogra- phy,* vol. 23, no. 1, pp. 157-165, 2020.

[14] K. F. F. Jiechieu and N. Tsopze, "Skills prediction based on multi-label resume classification using cnn with model predictions explanation," *Neural Computing Applications,* vol. 33, pp. 5069-5087, 2021.

[15] E. S. Chifu, V. R. Chifu, I. Popa, and I. Salomie, "A system for detecting professional skills from resumes written in natural language," in *2017 13th IEEE International Co1iference on Intelligent Computer Communication and Processing (ICCP),* pp. 189-196, 2017.