A Major Project Mid-term Report on

Smart Resume Parser and Analyzer

Submitted in Partial Fulfilment of the Requirements for

the Degree of **Bachelor of Engineering in Information Technology**

under Pokhara University

Submitted by:

Mr. Binod Dhamala, 191410

Mr. Arjun Uchai Thakuri, 191405

Mr. Prasiddha Pokhreal, 191425

Mr. Swastik Chaudhary, 191447

Under the supervision of :

Mr. Himal Acharya

Date: 14 June 2024

Department of IT Engineering

NEPAL COLLEGE OF INFORMATION TECHNOLOGY

Balkumari, Lalitpur, Nepal

ABSTRACT

The “Smart Resume Parser, Ranker and Analyzer” web application is a comprehensive solution designed to streamline the recruitment process by leveraging advanced natural language processing (NLP) techniques. Developed with ReactJS for the frontend and Django for the backend, the system integrates the SpaCy NLP model to parse both applicant resumes and job descriptions provided by recruiters. It utilizes the Named Entity Recognition (NER) component of SpaCy to label each entity thus enabling the parsing process of application to assess the compatibility between an applicant’s skills and a job’s requirements, resulting in a dynamic ranking system. Notably, the implementation leverages a weighted score algorithm for ranking, adding a layer of sophistication to the assessment process. This system not only enhances the efficiency of the hiring process but also promotes fair and unbiased candidate assessment. By automating the initial screening stages, recruiters can focus on more strategic aspects of the hiring process, leading to quicker and more informed decision-making.

Keywords: *SpaCy NLP model, ranking system, NER, NLP, parsing*

Table of Content:

List of Figure:

List of Table:

1. Introduction

Traditional methods of manually analyzing resumes can be time-consuming and inefficient for recruiters. To overcome this challenge, automated resume parsing and analysis systems have been developed. These systems utilize Natural Language Processing (NLP) techniques to extract key information from resumes, such as contact details, work experience, education, and skills. To address this, the custom Named Entity Recognition (NER) model is trained using dedicated resume datasets. By training the model from scratch, recruiters can achieve more accurate and tailored results, enabling efficient resume analysis and improved candidate evaluation. In this context, the use of SpaCy, a powerful NLP library offers the opportunity to develop a custom resume parsing and analysis solution that better aligns with recruiters' specific needs.

Building a custom resume parsing and analytics solution using SpaCy not only accelerates the start of the recruitment process but also increases the quality of candidate selection. Using SpaCy's advanced NLP capabilities, recruiters can customize their NER model to identify and categorize a wide variety of terms and keywords specific to their industries, job functions, and organizational culture That system this ensures that the system can identify microskills, certifications and experience well are highly relevant but may be overlooked by advanced parsing tools.

Furthermore, by automating the tedious and time-consuming task of re-screening and initial screening, recruiters can save time many have allowed potential employees to communicate, better understand their aspirations and capabilities, and make informed decisions receive individualized feedback and feedback.

In conclusion, creating custom resume parsing and analytics solutions using SpaCy provides many benefits for recruiters. From increased productivity and accuracy to the ability to analyze candidate profiles, such a system can dramatically improve the recruitment process. By leveraging SpaCy's powerful NLP features, organizations can stay ahead of the competitive landscape of talent acquisition.

2. Problem Statement

Manual resume sorting and ranking present several significant challenges that impact the efficiency and effectiveness of the recruitment process. This method is inherently time-consuming and labor-intensive, requiring recruiters to meticulously review each resume individually. During periods of high applicant volume, this can become overwhelming, making it difficult to manage within tight hiring timelines and leading to delays in identifying and engaging top candidates. Accurate tracking, organizing, and retrieving specific resumes is another major issue with manual sorting. Without a structured system, resumes can easily get lost or misfiled, resulting in missed opportunities and incomplete candidate data, complicating efforts to maintain a comprehensive talent pool for future needs. Objectively comparing candidates with diverse backgrounds and experiences is particularly challenging. Human biases, both conscious and unconscious, can inadvertently influence evaluations, potentially overlooking qualified candidates who do not fit conventional profiles. This lack of objectivity can lead to less equitable hiring practices.

Furthermore, scalability is a critical concern. As organizations grow and the volume of applications increases, the manual process becomes increasingly unsustainable. Recruiters may struggle to keep up with the influx of resumes, creating bottlenecks that slow down the entire hiring process and negatively impact the candidate experience. These challenges underscore the need for more streamlined, automated solutions to enhance the efficiency and fairness of recruitment practices.

3. Objectives

- To develop a custom resume parsing and analysis system that makes job recruitment much easier and faster.
- To implement weighted score algorithm that can adapt to varying job requirements and prioritize candidate resumes based on relevance to specific job descriptions.

4. Significance of Study

The Smart Resume Analyser project holds substantial significance in both the job-seeking and recruitment landscapes. For job seekers, the tool offers a transformative advantage by providing tailored feedback and recommendations to enhance their resumes. By leveraging NLP and ML technologies, the system ensures that resumes are optimized for specific job roles, significantly improving candidates' chances of securing interviews and landing their desired positions.

For employers, the project addresses critical inefficiencies in the traditional recruitment process. Manual sorting and ranking of resumes are often time-consuming, labor-intensive, and prone to errors. The automated approach of the Smart Resume Analyser streamlines these tasks, enabling faster, more accurate, and objective evaluation of applicants. This not only reduces the workload for recruiters but also ensures a more fair and consistent hiring process.

Furthermore, the scalability of the system allows organizations to handle growing applicant pools without bottlenecks, facilitating a smoother and more efficient recruitment process. Overall, the Smart Resume Analyser enhances the quality and efficiency of job applications and recruitment, benefiting both job seekers and employers.

5. Scope and Limitation

Our website provides the following scope and limitation:

5.1. Scope:

- The system will target the extraction and analysis of key information from resumes, including personal details, work experience, education, and skills.
- The scope also includes the automation of the resume parsing process, providing an efficient solution for recruiters to streamline candidate evaluation and selection.

5.2. Limitation:

- System's Accuracy may be affected if resumes do not provide complete or accurate information. Candidates may omit certain details, exaggerate their qualifications, or include irrelevant information.
- Job requirements and company needs to evolve over time. If the system is not updated regularly, it may become less effective in ranking resumes based on the latest criteria
- Dealing with resumes in different languages adds another layer of complexity.

6. Literature Review

CV Analysis Using Machine Learning

[1] This particular report published in the iJRASET (International Journal for Research in Applied Science and Engineering Technology) makes a similar model using NLP and tests its accuracy using various means. In particular it uses and shows a formula to calculate the score using various measures such as skillset, experience, required work years etc. Each are given different weightage and used in a formula to calculate the score. This inspired us to make a similar formula of our own to calculate a score.

Ayehigh resume shortlister

[2] This is the first resume short lister present on the internet when searching for one through a browser. It seems to want a job description for the job you are looking for and uses that as a basis for shortlisting your cv by finding out relevant terms related to the specific job or position. It gives you a score on your cv, a resume analysis, a guide for optimizing your resume and uses LLMs (large language models) to compare resume with job description. We want our app to work like this one but at a lower level to start before implementing more features.

Resume Parser And Analyzer Using NLP

[3] This particular report published in the International Research Journal of Modernization in Engineering Technology and Science (irjmets) has an interesting concept of analysing the field which are extracted from resumes. Then the system will suggest some areas where resume can be improve like required skills to be added in resume, quality of resume, enhancements required. This type of resume parser introduced the concept of quality of resume and how to improve it. We decided to not implement a quality improvement aspect for the time being but decided to assign a quality number to the project to give a quantitative answer regarding how good the resume was.

spaCy

spaCy [4] is a free, open-source Python library that provides advanced capabilities to conduct natural language processing (NLP) on large volumes of text at high speed. spaCy has its own deep learning library called thinc used under the hood for different NLP models. for most (if not all) tasks, spaCy uses a deep neural network based on CNN with a few tweaks. Specifically for Named Entity Recognition, spacy uses:

1. A **transition based approach** borrowed from shift-reduce parsers
2. A framework that's called **"Embed. Encode. Attend. Predict"**

    - **Embed**: Words are embedded using a Bloom filter, which means that word hashes are kept as keys in the embedding dictionary, instead of the word itself. This maintains a more compact embeddings dictionary, with words potentially colliding and ending up with the same vector representations.
    - **Encode**: List of words is encoded into a sentence matrix, to take context into account. spaCy uses CNN for encoding.
    - **Attend**: Decide which parts are more informative given a query, and get problem specific representations.
    - **Predict**: spaCy uses a multi layer perceptron for inference.

7. Related Theory

7.1. Natural Language Processing

Natural Language Processing (NLP) is a branch of artificial intelligence (AI) dedicated to enabling computers to understand, interpret, and optimize human speech NLP is about developing programs and techniques for natural language data, such as text and language processing and analysis To be, gain valuable insights and develop actions These tasks include classifying unstructured information into manageable units, understanding the meaning and structure of language, acquire natural language data or speech, analyze emotions, extract text, answer questions, classify text, translate speech NLP speech processing techniques are applied and analyzed by statistical and statistical machine learning models use, which are often trained on big data. These models range from traditional rule-based approaches to more advanced deep learning frameworks such as recurrent neural networks (RNNs) and transformers NLP powers a variety of applications such as virtual assistants, chatbots, search engines, sentiment analysis tools, and language translation applications As it comes, NLP will continue to evolve, allowing computers and humans to communicate intelligently and naturally By combining a wide range of data processing with an understanding of context and nuance, NLP holds promise to transform communication, decision-making and information capture in sectors ranging from health care finance to education and entertainment.

7.2. Tokenization

Tokenization is a fundamental process in Natural Language Processing (NLP) where a given text is broken down into individual words or tokens. Each token represents a unit of meaning, simplifying the analysis of language by providing a basis for further linguistic processing. For example, in the sentence.”The quick brown fox jumps,” tokenization would break it into individual tokens. This technique is crucial for tasks like part-of-speech tagging, named entity recognition, and other analyses that require understanding the structure and meaning of words within a text. Consider the following sentence from a resume:

”Experienced software developer with proficiency in Python, Java, and C++.” After tokenization, this sentence would be represented as a list of tokens: [”Experienced”, ”software”, ”developer”,”with”, ”proficiency”,”in”, ”Python”, ”,”, ”Java”, ”,”, ”and”, ”C++”, ”.”]

Each word, punctuation mark, or symbol is treated as a separate token. Tokenization is important for a variety of NLP tasks, including part-of-speech marking, named entity recognition, and semantic analysis. For example, in part-of-speech marking, syntactic categories (such as noun, verb, adjective) are assigned to each token to help us understand the syntactic structure of the text Named entity recognition identifies and classifies specific objects a mentioned in the text, such as names of people, institutions, or places. The goal of semantic analysis is to understand the meaning conveyed by the text, based on accurately tokenized representations of the input. Overall, tokenization is an important pre-processing step in NLP applications such as resume parsing, categorizing information into meaningful categories and providing a basis for subsequent processable analysis again and rendered by computer.

7.3. Document Understanding and Information Extraction

This title emphasizes the goal of extracting structured information from unstructured documents. This technique involves analyzing the content of documents to identify and extract relevant data, such as a person’s name, contact information, education history, work experience, skills, and qualifications, job post, email, and other related entities. Various parsing techniques can be employed, including rule-based parsing, template-based parsing, and machine learning-based parsing. Rule-based parsing relies on predefined rules and patterns to extract information, while template-based parsing uses predefined templates to guide the extraction process. Machine learning-based parsing utilizes algorithms trained on annotated data to automatically learn patterns and extract information from resumes. Overall, resume parsing plays a crucial role in automating the recruitment process by efficiently extracting and 5 organizing candidate information from large volumes of resumes.

7.4. Transformer

The Transformer is a deep learning architecture for sequential data processing, using self-attention mechanisms to capture relationships between elements in a sequence, such as words in a sentence. It enables parallelization and avoids sequential computations, making it highly efficient. Roberta, an advanced transformer-based model, is tailored for natural language processing (NLP) tasks. Built on the transformer architecture, it excels in capturing bidirectional contextual information within sequential data. During pre-training, RoBERTa employs a masked language model objective, predicting masked words to understand intricate contextual relationships. It outperforms its predecessor, BERT, by training on larger datasets, incorporating hyperparameter tuning, and removing the Next Sentence Prediction (NSP) task. With a focus on bidirectional context understanding, RoBERTa is effective in various NLP applications, demonstrating versatility and robust performance in tasks such as text classification, named entity recognition, and sentiment analysis.

7.5. Resume Matching and Ranking

Resume matching and ranking are key processes in modern recruitment, aimed at efficiently identifying the most suitable candidates for job roles. Resume matching involves comparing resume content with job descriptions to assess candidate suitability. This process typically utilizes keyword matching, semantic analysis, or machine learning algorithms. Resume ranking prioritizes resumes based on relevance to job requirements, considering factors like skillset, experience, and education. These processes streamline recruitment, saving time and resources while ensuring objective candidate evaluation. Leveraging technology, such as natural language processing and machine learning, enhances the accuracy and efficiency of matching and ranking algorithms, empowering organizations to make better hiring decisions.

7.6. Evaluation Technique

Evaluation techniques in machine learning are methods used to assess the performance and generalization ability of models on unseen data. In the context of resume parsing and ranking, common evaluation techniques include cross-validation, holdout validation, and train-test split. Cross-validation involves partitioning the dataset into multiple subsets, training the model on a subset while validating on the rest, and repeating this process multiple times to obtain robust performance estimates. Holdout validation splits the dataset into training and validation sets, where the model is trained on the training set and evaluated on the validation set. Train-test split divides the dataset into two parts, one for training the model and the other for testing its performance on unseen data. These evaluation techniques help assess the model's ability to generalize to new data and identify potential issues such as overfitting or underfitting. By employing appropriate evaluation techniques, practitioners can make informed decisions about model selection, hyperparameter tuning, and optimization strategies.

8. Methodology

8.1. Software Development Model

Agile is an iterative and incremental software development approach that focuses on flexibility, collaboration, and continuous improvement. It involves dividing the project into short sprints, regularly delivering working software, gathering feedback, and adapting the product based on customer needs. The process emphasizes close collaboration between stakeholders and development teams, allowing for quicker response to changes and delivering value early in the development lifecycle.

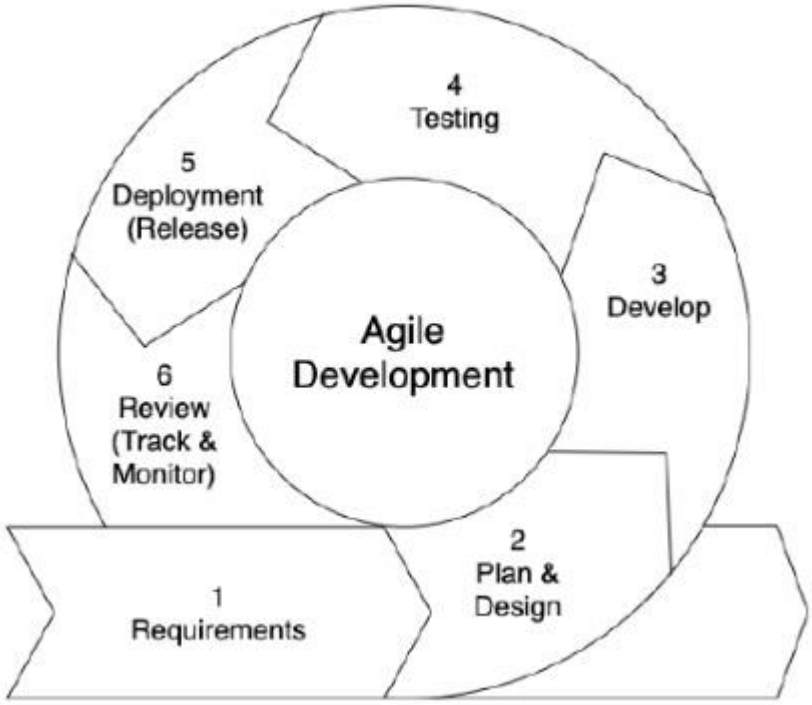


Figure 1: Step in Agile SDLC model

8.2. Requirement Analysis

8.2.1. Functional Requirements:

- Extract essential resume details accurately and comprehensively from various formats and structures.
- Effectively identify and extract specific entities like names, contact details, job titles, companies, institutions, and skills.
- The system should employ a matching algorithm to evaluate how well a candidate's resume aligns with the job description.
- Perform validation checks for data accuracy, consistency and include a ranking mechanism for efficient candidate evaluation and selection.

8.2.2. Non-Functional Requirements:

- Stable and consistent performance with minimal downtime
- The user interface should be visually appealing and easy to navigate, enhancing the overall user experience.
- Flexible to adapt to different resume formats and user requirements. 4.3. Hardware Requirements
- Adequate processing power, memory (RAM), and storage capacity to handle computational demands and store data efficiently.

8.2.3. Software Requirements:

- Natural Language Processing (NLP) Library: SpaCy
- Database Management System: PostgreSQL
- Programming Language: Python, Javascript
- Version Control: Git and Git hub
- Integrated Development Environment (IDE): Visual Studio Code, Jupyter Notebook

8.3. System Architecture

Architecture Description The architecture of the proposed system consists of the following components:

User Interface: The system provides a user-friendly web interface where job seekers can upload their resumes, and organizations/recruiters can list job descriptions.

Backend: The backend will handle the incoming requests from the frontend and manage the entire processing flow. It will host the NLP models, perform data processing, and implement the ranking algorithm.

Data Storage: A database is utilized to store the uploaded resumes, job descriptions, and the extracted resume information. It ensures efficient data retrieval and management.

Training Data Preparation: A diverse resume dataset is collected and prepared for training the spaCy NER model. The dataset includes various resume formats, layouts, and terminologies to enhance the model's adaptability.

Training Model: The prepared training data is used to train a custom NER model on a labeled dataset to recognize named entities (e.g., skills, education, experience). SpaCy is a good choice for implementing NER models.

Parsing and Analysis: When a resume is uploaded, the system applies the trained spaCy NER model to extract relevant information such as personal.details, work experience, education, and skills. The extracted data is stored in the database for further processing.

Job Description Matching: The system compares the extracted resume information with the requirements specified in the job descriptions. Factors such as skills, experience, and other relevant criteria are evaluated to assess compatibility.

Scoring and Ranking: Based on the evaluation, resumes are assigned scores and ranked to facilitate efficient sorting. This enables recruiters to identify top-matched candidates easily.

Output and Reporting: The system generates a summarized output, providing recruiters with a concise overview of the most suitable candidates. This assists in streamlined candidate selection and decision-making.

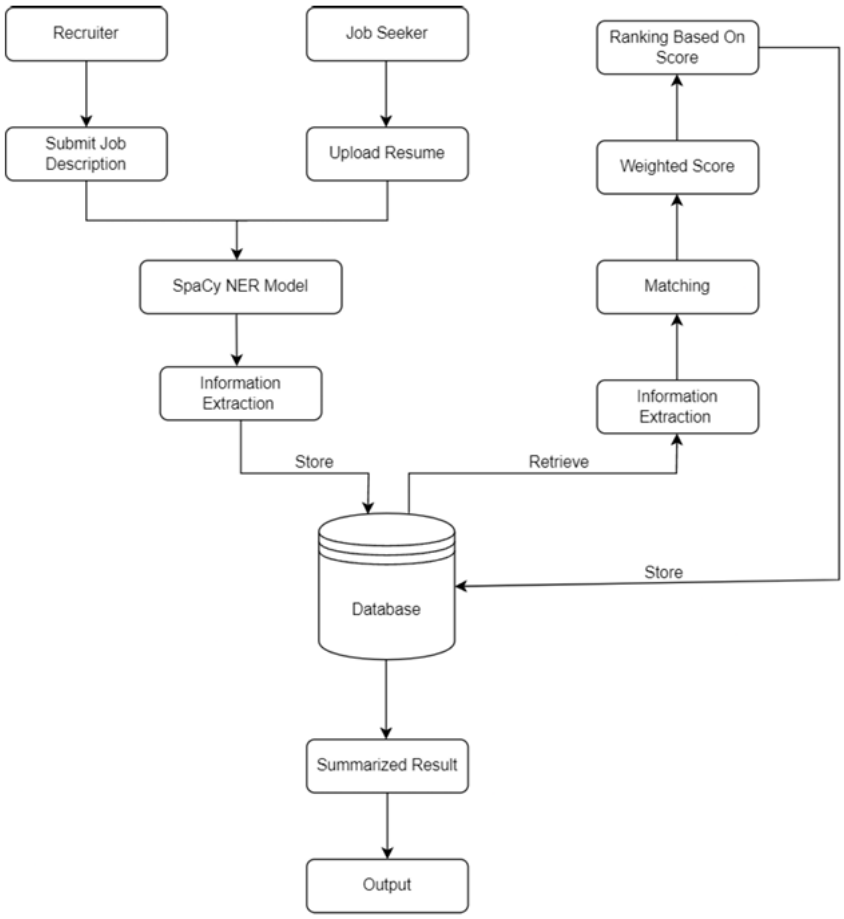


Figure 2: System Architecture

The system architecture shown in Figure 2 automates the process of screening and ranking resumes for job openings. Recruiters first submit a job description outlining the requirements. Job seekers then upload their resumes in response. Here, the model extracts key information from the resumes, such as skills and experience. This information is stored in a database and then compared against the job description requirements. Based on how well the candidate's qualifications match the job, a weighted score is assigned and resumes are ranked accordingly. Recruiters receive a summarized report with ranked lists of qualified candidates, streamlining the process of identifying potential hires.

8.4 Data Collection

The development of an effective resume parsing system involves meticulous data collection. Initially, a diverse set of resume samples is gathered, covering various backgrounds and experiences. Similarly, a comprehensive collection of job description samples is amassed, representing different positions, levels of specificity, and requirements. The synergy between the collected data ensures the system's parsing capabilities are enhanced, resulting in a more resilient and robust parsing system capable of accurately extracting relevant information from diverse resumes and job descriptions.

8.5. Data Pre-Processing

8.5.1. Text Cleaning

Data cleaning is a crucial step in the data preprocessing pipeline, aimed at ensuring that the dataset is free from any irrelevant or inconsistent information that could potentially affect the accuracy and reliability of subsequent analyses. In the context of natural language processing (NLP), data cleaning involves the removal of irrelevant characters, symbols, and formatting elements from the text data. This includes removing special characters, punctuation marks, and whitespace, as well as standardizing text to ensure consistency in parsing. By removing irrelevant characters 14 and symbols, such as HTML tags or non-alphanumeric characters, the text data is streamlined and made more uniform, facilitating easier processing and analysis.

8.5.2. Data Annotation

Annotation involves the process of identifying and labeling key entities within both resumes and job descriptions using Named Entity Recognition (NER) annotation tools, leveraging the NER feature of Spacy. For resumes, this includes elements such as applicant names, education details, work experience, skills, email, location, and LinkedIn links. Similarly, job description annotation entails identifying essential elements within job postings, such as job titles, required skills, required experience, and required degree. These annotated datasets provide structured information essential for further processing and analysis.

8.5.3. Language Processing Pipeline

When we call nlp on a text, spaCy first tokenizes the text to produce a Doc object. The Doc is then processed in several different steps – this is also referred to as the processing pipeline. The pipeline used by the trained pipelines typically include a tagger, a lemmatizer, a parser and an entity recognizer. Each pipeline component returns the processed Doc, which is then passed on to the next component. The language processing pipeline is shown in figure.

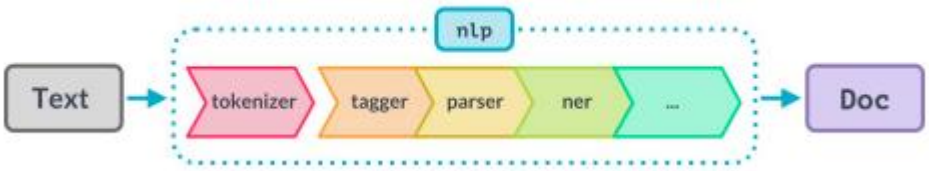


Figure 3: Language processing pipeline

Tokenization

This step breaks down the raw text into individual units called tokens, such as words or punctuation marks. Each token represents a discrete unit of text and forms the basis for further processing.

Lemmatization

This step involves reducing words to their base or dictionary form, known as the lemma. This is useful for standardizing words and reducing inflectional forms to their common base form.

Tagger (Part-of-Speech Tagging)

In this step, the tagger assigns grammatical categories, known as parts of speech, to each token in the text. This helps identify the role of each word in the sentence, such as nouns, verbs, or adjectives.

Parser (Dependency Parsing)

In this step, the parser analyzes the syntactic structure of sentences and establishes relationships between words based on their grammatical dependencies. It determines how words are connected in a sentence, such as subject-verb or modifier relationships.

NER (Named Entity Recognition)

In this step, the NER component identifies and classifies entities in the text, such as person names, locations, emails, degrees, dates, etc. It detects spans of text that represent named entities and assigns them a label indicating their entity type, providing important contextual information.

8.5.4. Data Splitting

Splitting an annotated dataset into training and evaluation processes is important in machine learning, especially for tasks such as natural language processing (NLP). This separation is crucial for training the model on one subset and evaluating its performance on another, ensuring robust testing of unseen data. To achieve this, we split the data by assigning 70% of the data set for training and the remaining 30% for analysis. These considerations strike a balance between providing sufficient data to train robust models and ensuring that a larger set of analyses is established for more accurate performance analysis By experimenting with training a range of models and unseen cases so we prevent overfitting and obtain reliable estimates of model real-world performance generalize well to and perform reliably in practical application.

8.5.5. Structured Data Representation

In this step, we convert annotated data into a structured representation, capturing key entities such as applicants' names, educational backgrounds, work experience, and skills. This structured process is the cornerstone for training and evaluating the performance of the system in extracting relevant information from the unstructured data.

By sorting the data, we convert the raw text into machine-understandable form. This requires text to include a label or labels that identify specific elements in the text and their location. For example, when using the named company identification (NER) for identifying and sorting information such as names of persons, dates, organizations, and places of naming Once data is organized and annotated, it is invaluable for training and evaluation of the effectiveness of NLP models optimized for resume parsing, information extraction, and so on

Overall, these steps are key to developing a robust NLP system. By providing customized and labeled data for training and analysis, enables the system to learn from annotated examples and make text comprehension more general. As a result, the system

is able to efficiently process and extract information from different types of information, making it more efficient and accurate in different applications.

8.6. Model Training

8.6.1. Setup Config

File In this step, we use the SpaCy base default configuration to create a custom configuration file. By creating a custom configuration file, we fine-tuned parameters like maximum steps, evaluation frequency, and learning rate to optimize performance. This approach allowed us to efficiently adapt spaCy's powerful capabilities to our specific needs, resulting in enhanced accuracy and streamlined development.

8.6.2. Initializing the Model with Custom Configuration

In our project, we initialized the spaCy NER model with custom configurations to optimize its performance for entity recognition tasks.

8.6.3 Train Model

Train the model on the annotated data using spaCy's training command, specifying the configuration file, training, and evaluation datasets. It involves teaching a model to identify entities like names of people, organizations, etc., in text. It begins with a dataset containing labeled examples of text. During training, the model learns to predict entity labels by adjusting its internal parameters through gradient descent and backpropagation, minimizing the error between predictions and annotations. The goal is not memorization but generalization across different contexts, ensuring the model performs well on unseen data. Evaluation data is used to assess the model's performance. Through iterative adjustments, the model improves its ability to accurately recognize entities in various contexts. NER Training figure is shown figure.

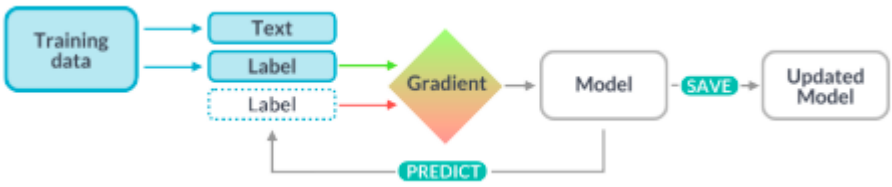


Figure 4: NER Training

Loss Calculation

NER training involves two primary loss functions. The first, NER loss, and the other, is transformer loss.

NER Loss:

This loss function is crucial for the accurate identification of named entities. It compares the predicted labels generated by the NER model with the ground truth annotations provided in the training data. The aim is to minimize the discrepancy between the predicted and actual entity labels, ensuring precise entity recognition.

Transformer Loss (RoBERTa):

This loss function is crucial for evaluating the model's language understanding. This loss function focuses on predicting the next tokens in sequences, assessing the model's ability to comprehend and generate coherent text. RoBERTa employs an entropy loss function to measure the uncertainty of predictions. Minimizing this uncertainty enhances the model's proficiency in language understanding tasks, as it becomes more accurate and confident in predicting subsequent tokens.

Entropy loss is calculated as,

$$\text{Loss} = -\sum_{i=1}^{n} \text{yi} \cdot \log(\text{pi}) + (1 - \text{yi})\log(1 - \text{pi})$$

$$\text{Gradient} = \partial\text{Loss} / \partial\text{Parameters}$$

where,

n is the number of classes

yi is the true label for class i

pi is the predicted probability for i-th class

8.7. Model Evaluation

Confusion matrix will be used for performance evaluation.

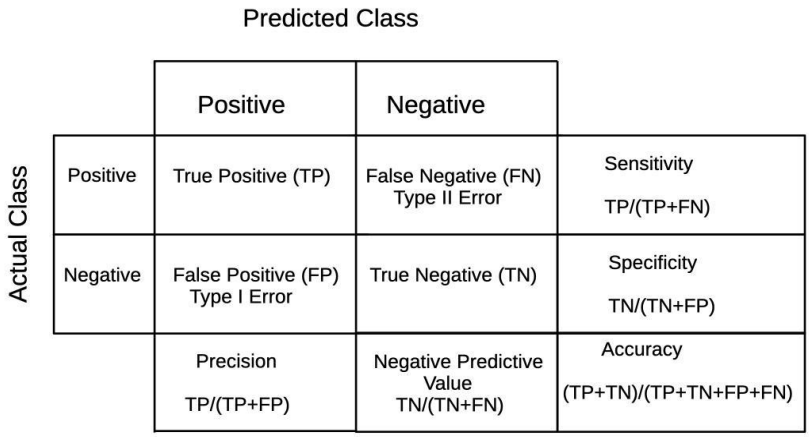


Figure 5:Confusion matrix

A confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a classifier. It is used to measure the performance of a classification model. It can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy, precision, recall, and F1-score.

8.8. UML Diagram

8.8.1. Use case Diagram

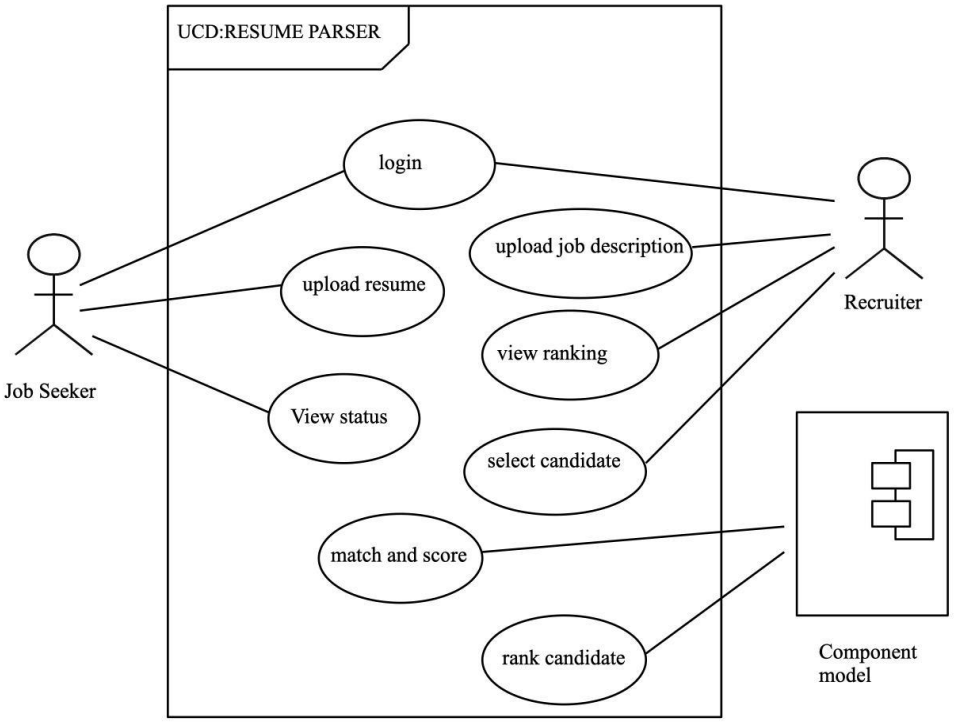


Figure 6: Use Case Diagram

Brief

Use Case Name: Login (UC#1)

A user opens the web application of Smart Resume Parser and Analyzer. The user tries to login into the website. If the user hasn't registered their account, then at first, they must register their account and login into their account. After that, the user enters the dashboard. An recruiter also gets logged in. After logging in, the recruiter also enters the dashboard.

Use Case Name: Upload Resume (UC#2)

The user uploads the resume and waits for the system to process information. At last, the system validates the resume.

Use Case Name: View Status (UC#3)

The candidate uploads the resume and waits for the system to process information. After that recruiter check the status of the resume analysis. At last, the system provides with the status.

Use Case Name: Upload job description (UC#4)

The recruiter uploads job description. the system then validates the job description been provided.

Use Case Name: View Ranking (UC#5)

When the candidate uploads resume. The system then analyzes the resume. The recruiter then tries viewing ranking of uploaded resume.

Use Case Name: Match and Score (UC#6)

System after analyzing the resume, match the resume with job description and score accordingly.

Use Case Name: Rank Candidate (UC#7)

The system after analyzing the resume, then ranks the candidate who matches with given description provided by recruiter.

Casual

Use Case Name: Login (UC#1)

Main Success Scenario:

The users open the web app and login to the webpage. The system checks

the validation. Now, the users are provided with home page.

Alternate Scenario:

1. If the account is not registered.
2. If user fills wrong info
3. If the password is forgotten.
4. If system failure occurs.

Use Case Name: Upload Resume (UC#2)

Main Success Scenario:

The candidate gets an option upload the resume after verification procedure. then they upload it and waits for the system to respond.

Alternate Scenarios:

1. If the account is not registered.
2. If user fills wrong info
3. If the password is forgotten.
4. If the system failure occurs.
5. If the resume uploaded doesn't match the format.

Use Case Name: View Status (UC#3)

Main Success Scenario:

The user uploads the resume and waits for the system to process information. At last, the system validates the resume.

Alternate Scenarios:

1. If the account is not registered.
2. If user fills wrong info
3. If the password is forgotten.
4. If system failure occurs.
5. If the resume uploaded doesn't match the format.

Use Case Name: Upload job description (UC#4)

Main Success Scenario:

The recruiter uploads job descriptions and then validates the job description been provided.

Alternate Scenarios:

1. If the account is not registered.
2. If user fills wrong info
3. If the password is forgotten.
4. If system failure occurs.

5. If the job description uploaded doesn't match the format.

Use Case Name: View Ranking (UC#5)

Main Success Scenario:

When the candidate uploads resume. Then the system analyses resume and the recruiter then tries viewing ranking of uploaded resume .

Alternate Scenarios:

1. If the account is not registered.
2. If user fills wrong info
3. If the password is forgotten.
4. If system failure occurs.
5. If the resume has not been uploaded properly.
6. If the job description uploaded doesn't match the format
7. If the model is not tuned properly.

Use Case Name: Match and Score (UC#6)

Main Success Scenario:

System after analyses the resume, match the resume with job description and score accordingly.

Alternate Scenarios:

1. If the account is not registered.
2. If user fills wrong info
3. If the password is forgotten.
4. If system failure occurs.
5. If the resume has not been uploaded properly.
6. If the job description uploaded doesn't match the format
7. If the model is not tuned properly.

Use Case Name: Rank Candidate (UC#7)

Main Success Scenario:

The system after analyses the resume, then ranks the candidate who matches with given description provided by recruiter.

Alternate Scenarios:

1. If the account is not registered.
2. If user fills wrong info
3. If the password is forgotten.
4. If system failure occurs.
5. If the resume has not been uploaded properly.
6. If the job description uploaded doesn't match the format
7. If the model is not tuned properly.

Fully Dressed

Use Case Name: Login (UC#1)

Scope: Smart Resume parser and Analyzer

Level: User goal

Primary actor: Candidate and Recruiter

Stakeholders and Interests:

Candidate: Wants to login to the system.

Component model: Wants a properly filled login details.

Recruiter: Wants to login to the system and access the functionalities.

Preconditions: The user opens the application.

Success guarantee/Post conditions: The system has correctly processed the information provided by the user and logs in to the system.

Main success scenario/Basic Flow:

1. The user opens the application.
2. The system provides a login form to fill.
3. The user fills up the form providing accurate information.
4. The user submits the form and waits for the system to verify.

5.The system identifies the role of the user (Candidate or Recruiter).

6.If credentials are valid:

a. If the user is a candidate, the system redirects them to the candidate dashboard.

b.If the user is a recruiter, the system redirects them to the recruiter dashboard.

Extensions/Alternate Scenario:

1.If the system fails: User reloads the webpage.

2.If username or password is wrong: User re-enter username and password.

3.If the login failed: User reloads the login page and re-enter the username and password.

Special Requirements: A good working computer device and a good internet connection.

Technology and data variation list: Web based login page with username and password credentials.

Frequency of Occurrence: Depends on the flow of users.

Miscellaneous: None

Use Case Name: Upload Resume (UC#2)

Scope: Smart Resume parser and Analyzer

Level: User goal

Primary actor: Candidate

Stakeholders and Interests:

Candidate: Wants to properly upload the resume.

Component model: Wants a proper resume to further analyze.

Preconditions:

1. The user is authenticated and logged in.

2. The user has the necessary permissions to upload images.

Success Guarantee/Post conditions: The resume is successfully uploaded, and the system has parsed the resume, and the correct result is provided to the user.

Main success scenario/Basic Flow:

1. The user logs in or signs up to the system.
2. The system provides option for the user to upload resume.
3. The user uploads the resume.
4. The user waits for the information to process and get the result.

Extensions/Alternate Scenario:

1. At any time, system fails: User restarts the system, logs in and fills the form from the beginning.
2. If the uploaded resume is in an unsupported format: User uploads a resume in a supported format.

Special Requirements:

1. The system must support document formats (pdf).
2. The system should limit the maximum resume size to ensure performance and storage efficiency.

Technology and data variation list:

1. Resume selection is done by file picker.
2. Resume should be in supported format like pdf.

Frequency of Occurrence: Depends on the flow of users.

Miscellaneous: None

Use Case Name: View Status (UC#3)

Scope: Smart Resume parser and Analyzer

Level: User goal

Primary actor: Candidate, Component model

Stakeholders and Interests:

Candidate: Wants to check the resume status.

Component model: Wants proper analysis of resume.

Preconditions:

1. The user is authenticated and logged in.
2. The user has properly uploaded resume.

Success Guarantee/Post conditions: The resume is successfully analyzed and the candidate can view the status.

Main success scenario/Basic Flow:

1. The user logs in or signs up to the system.
2. The user uploads the resume.
3. The resume is analyzed and can be clearly stated.

Extensions/Alternate Scenario:

1. At any time, system fails: User restarts the system, logs in and fills the form from the beginning.
2. If the uploaded resume is in an unsupported format: User uploads a resume in a supported format.

Special Requirements:

1. The system must support document formats (pdf).
2. The system should limit the maximum resume size to ensure performance and storage efficiency.

Technology and data variation list:

1. Resume selection is done by file picker.
2. Resume should be in supported format like pdf.

Frequency of Occurrence: Depends on the flow of users.

Miscellaneous: None

Use Case Name: Upload job description (UC#4)

Scope: Smart Resume parser and Analyzer

Level: User goal

Primary actor: Recruiter

Stakeholders and Interests:

Recruiter: Wants to upload job description.

Preconditions:

1. The user is authenticated and logged in.
2. Job description in supported format.

Success Guarantee/Post conditions: The Job descriptions is successfully analyzed.

Main success scenario/Basic Flow:

1. The user logs in or signs up to the system.
2. The user uploads the job description.

Extensions/Alternate Scenario:

1. At any time, system fails: User restarts the system, logs in and fills the form from the beginning.
2. If the uploaded job description is in an unsupported format: User uploads a job description in a supported format.

Special Requirements:

1. The system must support document formats (pdf).
2. The system should limit the maximum description size to ensure performance and storage efficiency.

Technology and data variation list:

1. Job description is analyzed by file picker.
2. Job description should be in supported format like pdf.

Frequency of Occurrence: Depends on the flow of users.

Miscellaneous: None

Use Case Name: View Rank(UC#5)

Scope: Smart Resume parser and Analyzer

Level: User goal

Primary actor: Recruiter, Component model

Stakeholders and Interests:

Recruiter: Wants to view resume ranking.

Component model: Wants to rank the resume after smart parsing.

Preconditions:

1. The candidate uploads resume in correct format.
2. The recruiter uploads description properly

Success Guarantee/Post conditions: The resume is analyzed and ranked properly.

Main success scenario/Basic Flow:

1. The user logs in or signs up to the system.
2. The candidate uploads the resume.
3. The recruiter uploads job description.
4. The system analyzes and rank well.

Extensions/Alternate Scenario:

1. At any time, system fails: User restarts the system, logs in and fills the form from the beginning.
2. If the uploaded job description and resume is in an unsupported format: Appropriate user uploads them in a supported format.

Special Requirements:

1. The system must support document formats (pdf).
2. The system should limit the maximum description size to ensure performance and storage efficiency.

Technology and data variation list:

1.Document is analyzed by file picker.

2.Document should be in supported format like pdf.

Frequency of Occurrence: Depends on the flow of users.

Miscellaneous: None

Use Case Name: Match and score (UC#6)

Scope: Smart Resume parser and Analyzer

Level: User goal

Primary actor: System

Stakeholders and Interests:

Recruiter: Wants to upload job description.

Component model: Wants to parse the resume on basis of job description.

Preconditions:

1.The resume and job description are uploaded properly.

Success Guarantee/Post conditions: The resume analysis is successful.

Main success scenario/Basic Flow:

2.The user logs in or signs up to the system.

3.The system parses the resume properly.

Extensions/Alternate Scenario:

1.At any time, system fails: User restarts the system, logs in and fills the form from the beginning.

2.If the uploaded documents are in an unsupported format: User uploads them in a supported format.

Special Requirements:

1.The system must support document formats (pdf).

2.The system should limit the maximum description size to ensure performance and storage efficiency.

Technology and data variation list:

1.Resume is analyzed by file picker.

2.System performs accurately.

Frequency of Occurrence: Depends on the flow of users.

Miscellaneous: None

Use Case Name: Rank Candidate (UC#7)

Scope: Smart Resume parser and Analyzer

Level: User goal

Primary actor: System

Stakeholders and Interests:

Candidate: Uploads resume for parsing.

Component model: Wants to parses the resume and rank on basis of job description.

Preconditions:

1.The resume and job description are uploaded properly.

Success Guarantee/Post conditions: The resume is ranked successful.

Main success scenario/Basic Flow:

2.The user logs in or signs up to the system.

3.The system parses the resume properly.

Extensions/Alternate Scenario:

1.At any time, system fails: User restarts the system, logs in and fills the form from the beginning.

2.If the uploaded documents are in an unsupported format: User uploads them in a supported format.

Special Requirements:

1.The system must support document formats (pdf).

2.The system should limit the maximum description size to ensure performance and storage efficiency.

Technology and data variation list:

1.Resume is analyzed by file picker.

2.System ranks according to job description.

Frequency of Occurrence: Depends on the flow of users.

Miscellaneous: None

Two Column Variation

Use Case Name: Login (UC#1)

Actor Action	System Response
1.The use case begins when the user opens the application.	
	2.The system provides a login form to the user.
3.The user fills and submit the form.	
	4.The system verifies the login.

Table 1: Two column variation for Login

Use Case Name: Upload Resume (UC#2)

Actor Action	System Response
1.The use case begins when candidate is logged in. 3.The candidate uploads the resume.	2.The system provides an upload option

Table 2: Two column variation for Upload Resume.

Use Case Name: View Status (UC#3)

Actor Action	System Response
1.The use case begins when candidate upload resume. 2.The recruiter uploads job description. 4. The recruiter views the status.	3.The system analyze the resume.

Table 3: Two column variation for View Status.

Use Case Name: Upload job description (UC#4)

Actor Action	System Response
1.The use case begins when recruiter is logged in. 2. The recruiter uploads the job description.	 3.The system verifies the job description.

Table 4: Two column variation for Upload job description.

Use Case Name: View Ranking (UC#5)

Actor Action	System Response
2. The recruiter tries viewing ranking.	1. The use case begins when system analyzes the resume. 3.The system provides ranked resume.

Table 5: Two column variation for View Ranking.

Use Case Name: Match and Score (UC#6)

Actor Action	System Response
1.The use case begins when both resume and job description are uploaded.	 2.The system analyze the resume. 3.The system then matches and score resume with respect to job description.

Table 6: Two column variation for Match ad score resume .

Use Case Name: Rank Candidate (UC#7)

Actor Action	System Response
1.The use case begins when both resume and job description are uploaded.	2.The system then matches and score resume with respect to job description. 3.The system ranks candidate based on resume score.

Table 7: Two column variation for rank candidate

8.8.2. System Sequence Diagram (SSD)

A system sequence diagram is a fast and easily created artifact and illustrates input and output events related to the systems under discussion. It is a picture that shows, for a particular scenario of a use case, the events that external actors generate, their order, and inter-system events. All systems are treated as a black box; the emphasis of the diagram is events that cross the system boundary from actors to systems.

Use Case Name: Login (UC#1)

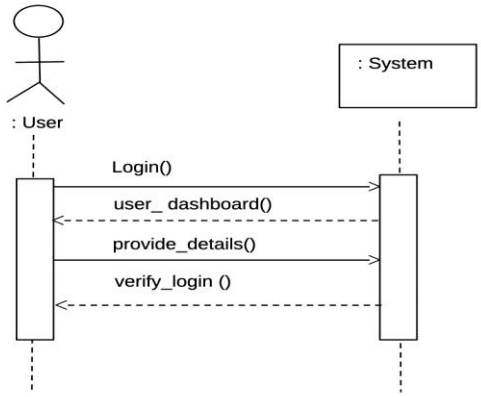


Figure 7: SSD for login

The SSD for use case Login is as shown in Figure. At first the user launches the application, then the system provides a login page to the user. The user fills all the necessary information in the login form and submits the form. Then the system identifies role of the user and verifies login

Use Case Name: Upload Resume (UC#2)

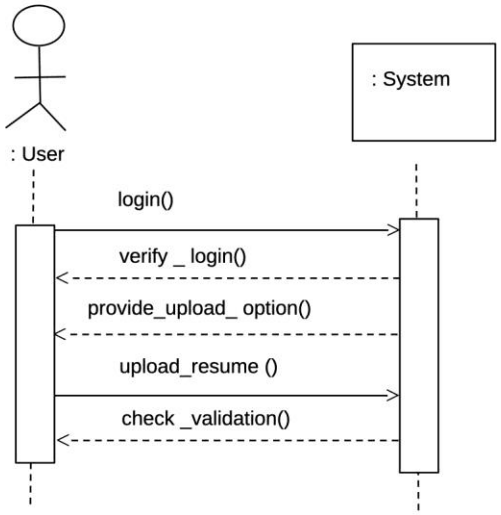


Figure 8: SSD for Upload Resume

The SSD for use case Upload Resume is as shown in Figure. At first the candidate logs in or gets registered in the system and the system verifies login. Then, the system provides an option for the user to upload resume. The candidate uploads the resume and waits for the system to process information. At last, the system validates the resume.

Use Case Name: View Status (UC#3)

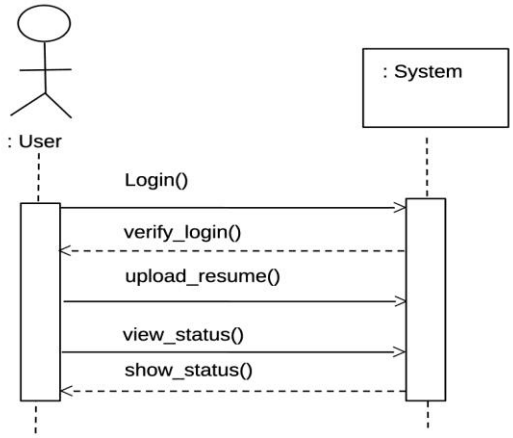


Figure 9: SSD for view status

The SSD for use case view status is as shown in Figure. At first the user logs in or gets registered in the system and the system verifies login. Then, the system provides an option for the candidate to upload resume. The candidate uploads the resume and waits for the system to process information. After that recruiter checks the status of the resume analysis. then the system provides with the status.

Use Case Name: Upload job description(UC#4)

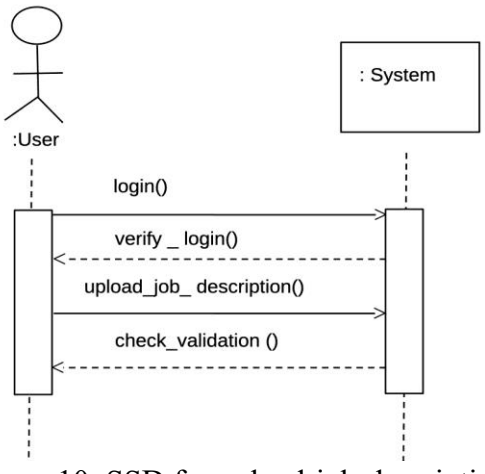


Figure 10: SSD for upload-job-description

The SSD for use case upload job description is as shown in Figure. At first the user logs in or gets registered in the system and the system verifies login. Then the recruiter uploads job descriptions. Then the system validates the job description been provided.

Use Case Name: View Ranking(UC#5)

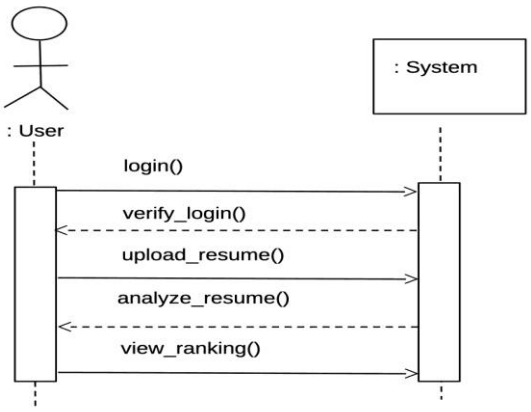
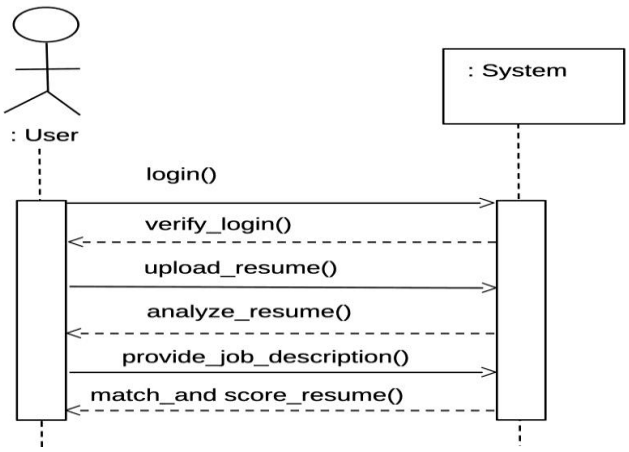


Figure 11: SSD for view ranking

The SSD for use case upload job description is as shown in Figure. At first the user logs in or gets registered in the system and the system verifies login. then the candidate uploads resume. Then the system analyzes the resume, then the recruiter tries viewing ranking of uploaded resume.

Use Case Name: Match and Score (UC#6)



The SSD for use case upload job description is as shown in Figure. At first the user logs in or gets registered in the system and the system verifies login, then the candidate uploads resume, Then the recruiter provides job descriptions, then the system analyze the resume and finally match the resume with job description and score accordingly.

Use Case Name: Rank Candidate (UC#7)

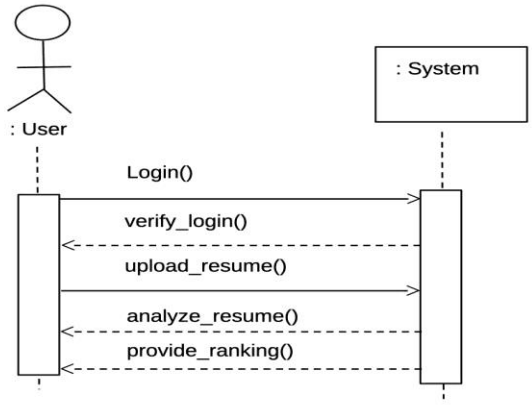


Figure 13: SSD for rank candidate

The SSD for use case upload job description is as shown in Figure. At first the user logs in or gets registered in the system and the system verifies login, then the candidate uploads resume, Then the recruiter provides job description, Then the system analyzes the resume and ranks the candidate who matches with given description.

8.8.3. Class Diagram:

A class diagram is a diagram that describes the structure of a system. It shows the system's classes, attributes, operations (or methods), and relationships among the classes. A class diagram is used to describe structure and behavior in the use cases. It provides a conceptual model of the system in terms of entities and their relationships. It is often used for requirement capture, end-user interaction.

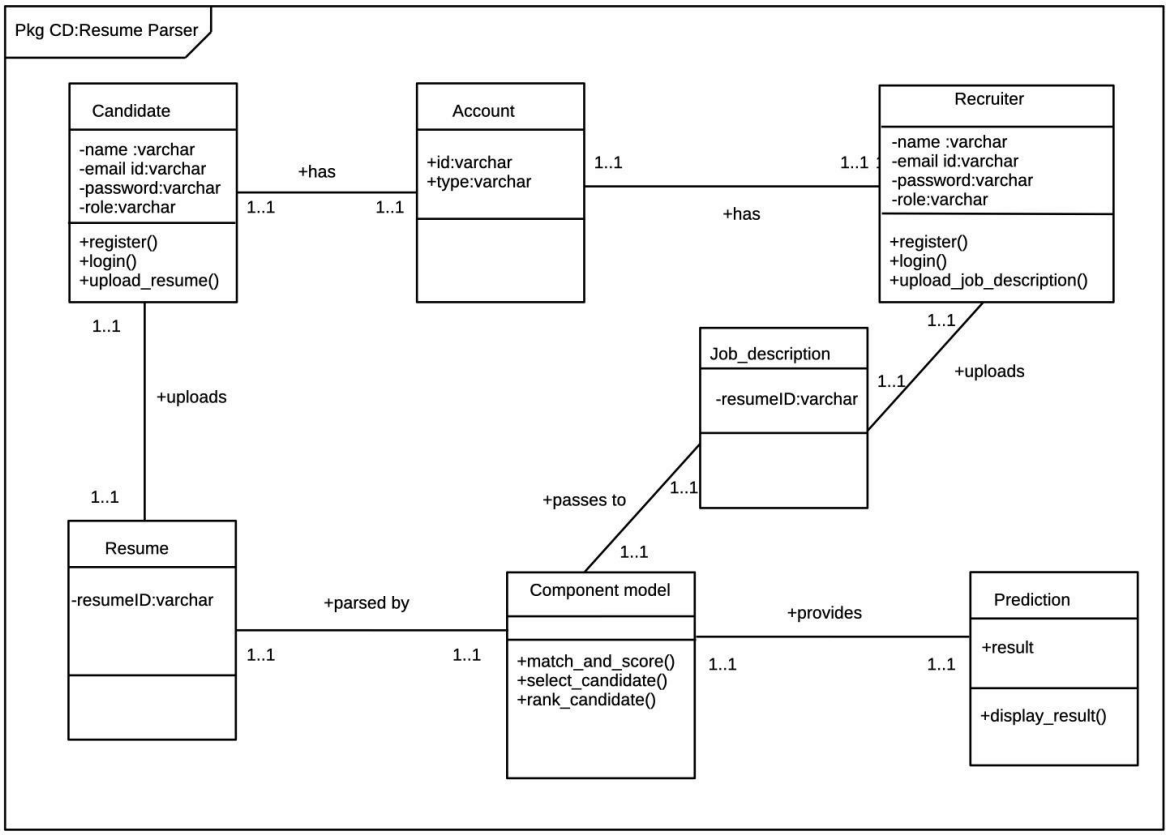


Figure 14: Class Diagram

8.8.4. Sequence Diagram:

A sequence diagram simply depicts interaction between objects in a sequential order i.e the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function.

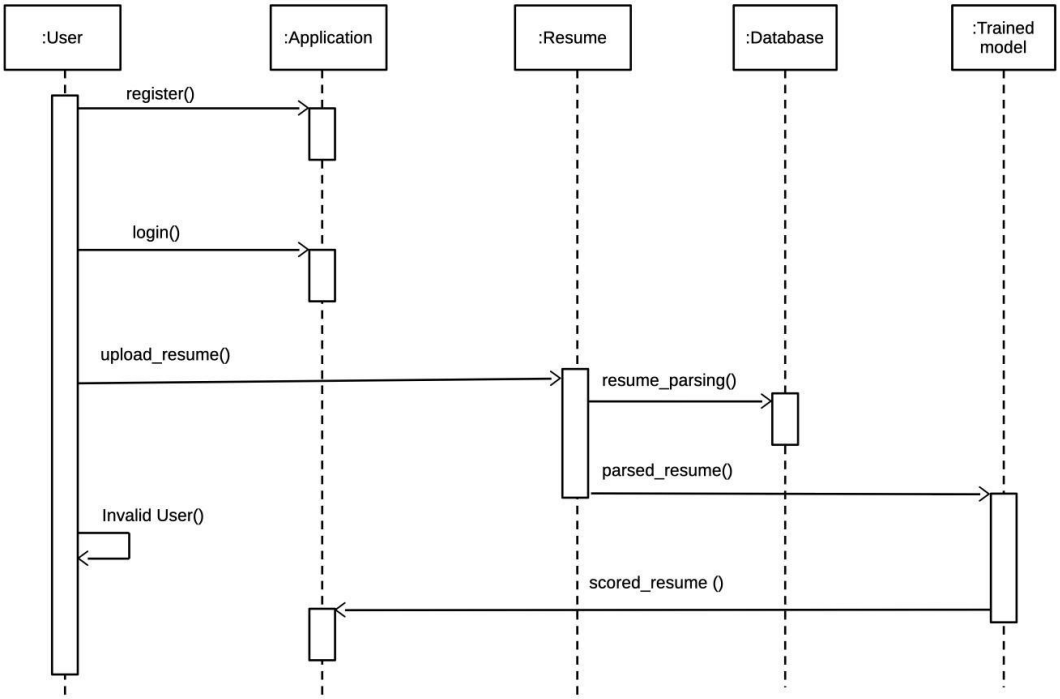


Figure 15: Sequence Diagram

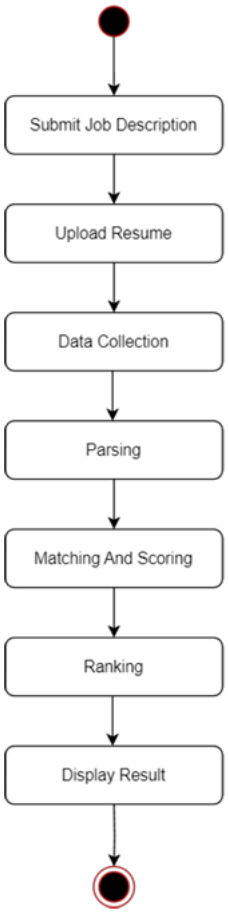


Figure 16: State Chart Diagram

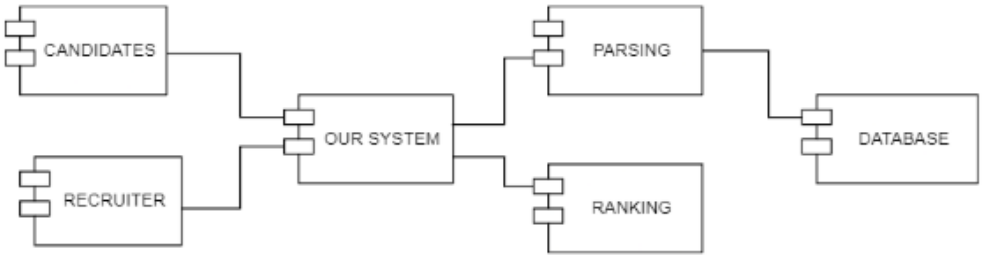


Figure 17: Compound Diagram

8.9. Tools and Technology

- Languages:
    a. Python
    b. Javascript
- Framework:
    a. Django
- Database:
    a. PostgreSQL
- Libraries:
    a. ReactJS
    b. SpaCy
    c. Pathlib
- Text editor:
    a. VS Code
    b. Jupyter Notebook, Google Colab
    c. Visualization tools: Matplotlib

9. Task Done So Far

We have successfully completed several key components to lay a solid foundation for resume parser and analyzer system. Firstly, we designed a user-friendly front-end interface, which allows users to upload their resumes and enables recruiters to submit job descriptions with ease. We have also integrated login authentication mechanism to ensure secure access for both job seeker and recruiters. Additionally, we have developed and trained a sophisticated model specifically for extracting and organizing information from resumes, which accurately and efficiently identifies critical details such as the candidate's name, skills, and experience etc. Lastly, we established a seamless connection to our database, ensuring that all extracted information is stored securely and can be efficiently retrieved for further processing.

10. Result and Discussions

```
============================= Training pipeline =============================
ℹ Pipeline: ['transformer', 'ner']
ℹ Initial learn rate: 0.0
E    #       LOSS TRANS...  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ------  -------------  --------  ------  ------  ------  ------
  0       0         936.39   1747.73    0.00    0.00    0.00    0.00
  1     200      202331.05  69729.36   25.88   89.42   15.13    0.26
  2     400       84462.67  23924.82   75.44   76.62   74.29    0.75
  3     600       30603.29  14949.82   83.66   82.77   84.56    0.84
  4     800       12000.44   8973.61   85.13   85.91   84.37    0.85
  5    1000        5353.80   7593.93   83.99   80.30   88.03    0.84
  6    1200       13997.15   7135.12   85.79   85.90   85.68    0.86
  7    1400        3883.18   6247.74   86.83   86.30   87.37    0.87
  8    1600        3546.61   6124.35   86.52   86.73   86.30    0.87
  9    1800        3402.56   5261.59   86.61   84.74   88.56    0.87
 10    2000        3007.26   5120.49   86.05   85.20   86.91    0.86
 12    2200        2776.79   4868.20   86.16   84.92   87.44    0.86
 13    2400        5386.81   4254.52   86.56   86.26   86.87    0.87
 14    2600        9937.09   4273.29   86.57   86.85   86.28    0.87
 15    2800        2478.09   4262.70   86.39   85.11   87.72    0.86
 16    3000        2183.80   3862.81   86.27   85.76   86.79    0.86
✓ Saved pipeline to output directory
content/drive/MyDrive/Project3/ResumeParser/model/output/model-last
```

```
Company Name City , State kitchen Manager / Catering and events chef  --->>> COMPANIES WORKED AT
Company Name City , State Kitchen manager / Executive chef  --->>> COMPANIES WORKED AT
Company Name City  --->>> COMPANIES WORKED AT
State Executive chef  --->>> WORKED AS
Company Name City , State Lead line cook  --->>> COMPANIES WORKED AT
Lyndon B Johnson , City , State , USA  --->>> UNIVERSITY
Serve safe
AA in applied science  --->>> CERTIFICATION
Station cook certificate  --->>> CERTIFICATION
Four dietary certificate  --->>> CERTIFICATION
Bilingual  --->>> LANGUAGE
Spanish  --->>> LANGUAGE
Skills  --->>> SKILLS
Grill cook specialist  --->>> SKILLS
Fry cook specialist  --->>> SKILLS
```

```
=============================== NER (per type) ===============================

                           P        R       F
LINKEDIN LINK          98.36    96.77   97.56
SKILLS                 86.61    75.23   80.52
LANGUAGE               76.54    80.89   78.66
CERTIFICATION          79.18    82.05   80.59
NAME                   94.46    94.77   94.62
LOCATION               78.98    89.86   84.07
COMPANIES WORKED AT    91.75    93.91   92.82
WORKED AS              89.18    92.10   90.62
YEARS OF EXPERIENCE    89.01    98.82   93.66
UNIVERSITY             70.72    65.69   68.11
DEGREE                 81.51    83.24   82.37
COLLEGE NAME           54.36    69.28   60.92
YEAR OF GRADUATION     41.67    28.17   33.61
AWARDS                 74.63    72.46   73.53
EMAIL ADDRESS          55.56   100.00   71.43


================================= Results =================================

TOK      100.00
NER P    85.76
NER R    86.79
NER F    86.27
SPEED    6211
```

11. Task Remaining

We need to develop a model for analyzing job descriptions and extracting key requirements. The next step involves creating a candidate ranking algorithm that considers various factors such as skills relevance and experience. We must conduct extensive testing and validation, optimize performance, and ensure data security and privacy compliance.

12. Deliverables:

- It should provide user-friendly interface where users can upload their resumes and recruiter can submit job descriptions.
- The complected system should provide the automated output from the resume with organized information, accurately and efficiently extracting details such as name, skills, experiences, and much more.
- The details obtained from the above process should be used by our system to rank the candidates based on which recruiters can focus on the most promising candidate, thereby reducing time and effort.

13. Project task and time Schedule:

The following chart presents the tentative schedule for the project.

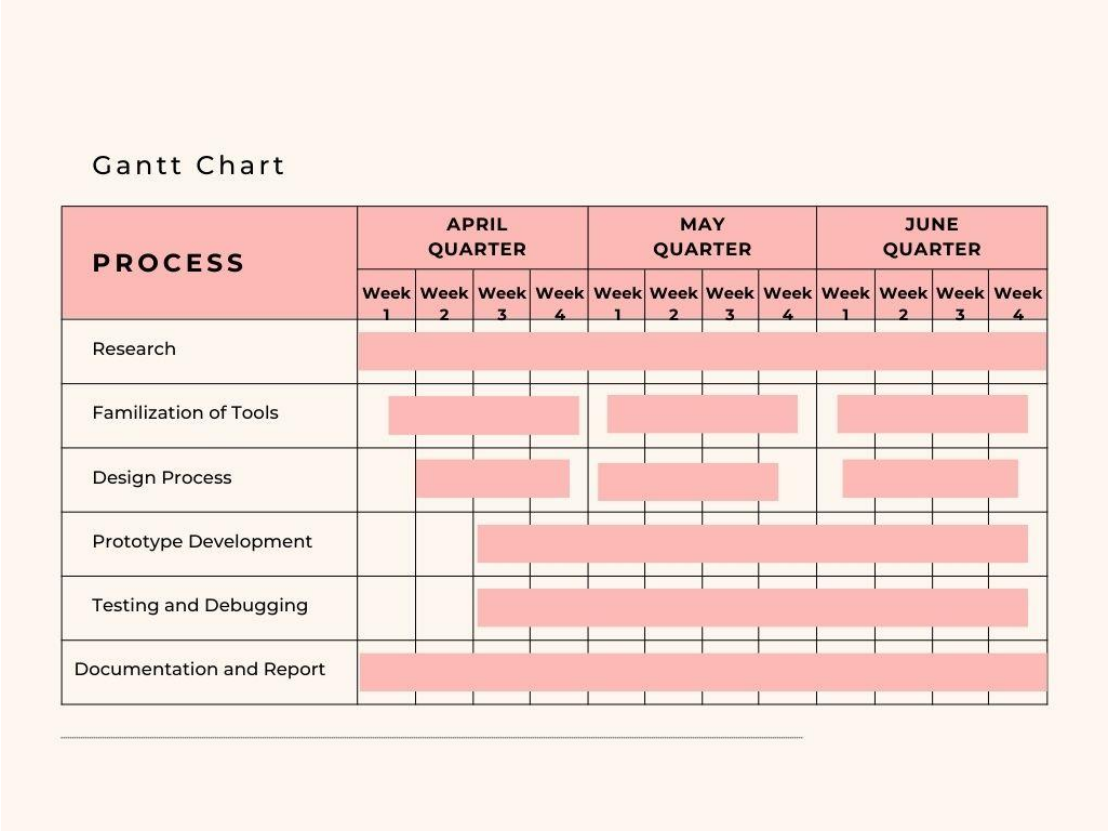


Figure 18:Gantt chart Diagram

Reference:

[1] Anand, A. and Dubey, *S. IJRASET Journal for Research in Applied Science and Engineering Technology, CV Analysis Using Machine Learning*. Available at: https://www.ijraset.com/research-paper/cv-analysis-using-machine-learning (Accessed: 17 May 2024).

[2*] AyeHigh Ai Resume shortlister, AyeHigh*. Available at: https://ayehigh.com/resume-shortlister (Accessed: 17 May 2024).

[3] Patil, N. and Yadav, *S. Resume parser and analyzer using NLP.* Available at: https://www.irjmets.com/uploadedfiles/paper//issue_4_april_2023/37428/final/fin_irjmets1683342426.pdf (Accessed: 17 May 2024).

[4] Raj and N. Singh, "*A Systematic Literature Review (SLR) On The Beginning of Resume Parsing in HR Recruitment Process & SMART Advancements in Chronological Order,*" Journal of Computational Science, vol. 48, pp. 100-115, 2022.

[5] Gupta and N. Gupta, "*The Impact of Smart Resume Parsing on the Recruitment Process*," Human Resource Management Review, vol. 31, no. 3, p. 100426, 2021.

[6] S. Gupta and R. Mittal, "*A Survey on Resume Parsing Techniques*," Journal of Big Data, vol. 7, no. 1, p. 13, 2020.

[7] Agarwal and M. Agarwal, "*A Review of Resume Parsing and Analysis Tools*," International Journal of Information Management, vol. 49, p. 102070, 2020.

[8] P. Gupta and R. Gupta, "*The Future of Resume Parsing and Analysis*," Future Computing and Information Systems, vol. 5, no. 1, pp. 1-10, 2020.

[9] M. Thejaswee, V. Srilakshmi, K. Anuradha, G. Karuna, Performance Analysis of Machine Learning Algorithms for Text Classification, in Proceedings of the Advanced Informatics for Computing *Research (ICAICR 2020),* A. K. Luhach, D. S. Jat, K. H. Bin Ghazali, Gao, P. Lingras, (eds), Comm. Comp. Inform. Sci. Springer, Singapore 1393 (2021).

[10] GeeksforGeeks, "*Confusion Matrix in Machine Learning*," GeeksforGeeks, [Online].Available:https://www.geeksforgeeks.org/confusion-matrix-machine-learning/. [Accessed: Apr. 25, 2024].