TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

SMART RESUME PARSER, RANKER AND ANALYZER

BY

BHUWAN THAPA MAGAR (PUR076BCT018)

DEBENDRA PUN (PUR076BCT028)

GANESH GAUTAM (PUR076BCT032)

SAROJ PAUDEL (PUR076BCT077)

A PROJECT SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

PURWANCHAL CAMPUS

DHARAN, NEPAL

6th MARCH, 2024

SMART RESUME PARSER, RANKER AND ANALYZER

By

BHUWAN THAPA MAGAR (PUR076BCT018)

DEBENDRA PUN (PUR076BCT028)

GANESH GAUTAM (PUR076BCT032)

SAROJ PAUDEL (PUR076BCT077)

Project Supervisor

Asst. Prof. Bishnu Chaudhary

A project submitted to the Department of Electronics and Computer Engineering in partial fulfillment of the requirements for the Bachelor’s Degree in Computer Engineering

Department of Electronics and Computer Engineering
Purwanchal Campus, Institute of Engineering
Tribhuvan University
Dharan, Nepal

6th March, 2024

DECLARATION

We declare that the work hereby submitted for Bachelors of Engineering in Computer Engineering at Institute of Engineering, Purwanchal Campus entitled **"SMART RESUME PARSER, RANKER AND ANALYZER"** is our own work and has not been previously submitted by me at any university for any academic award.

We authorize Institute of Engineering, Purwanchal Campus to lend this report to other institution or individuals for the purpose of scholarly research.

Bhuwan Thapa Magar (PUR076BCT018)
Debendra Pun (PUR076BCT028)
Ganesh Gautam (PUR076BCT032)
Saroj Paudel (PUR076BCT077)

6th March, 2024

RECOMMENDATION

The undersigned certify that they have read and recommended to the Department of Electronics and Computer Engineering for acceptance, a project entitled **"SMART RESUME PARSER, RANKER AND ANALYZER"**, submitted by **BHUWAN THAPA MAGAR, DEBENDRA PUN, GANESH GAUTAM, SAROJ PAUDEL** in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering in Computer Engineering"**.

....

Asst. Prof. Bishnu Chaudhary
Supervisor
Department of Electronics and Computer Engineering
Purwanchal Campus, Institute of Engineering, Tribhuvan University

....

Prof. Subarna Shakya, (PhD)
External Examiner
Department of Electronics and Computer Engineering
Pulchowk Campus, Institute of Engineering, Tribhuvan University

....

Asst. Prof. Pravin Sangroula
Head of Department
Department of Electronics and Computer Engineering
Purwanchal Campus, Institute of Engineering, Tribhuvan University

6th March, 2024

ACKNOWLEDGEMENT

This project was done as per the requirement of the fourth year major project in Bachelor's in Computer Engineering, Institute of Engineering (IOE), Tribhuvan University, Nepal. We would like to express our deepest gratitude to Asst. Prof. Bishnu Chaudhary sir for supervising this project with full dedication, support and guidance. Without him, this project would not have been possible.

We would also like to thank the Department of Electronics and Computer Engineering, IOE for giving us this great opportunity to carry out a major project that will help us shape our career among all the members of the department. We would like to acknowledge all the authors of the research papers that helped us understand the required concepts as well as algorithms better, and that we utilized to prepare this report.

Every attempt has been made to include each and every aspect of the project in this report so that the reader can clearly understand our project. We would be pleased to get feedback on this project.

Sincerely,
Bhuwan Thapa Magar (PUR076BCT018)
Debendra Pun (PUR076BCT028)
Ganesh Gautam (PUR076BCT032)
Saroj Paudel (PUR076BCT077)

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 CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	: Application Programming Interface
BERT	: Bidirectional Encoder Representations from Transformers
Colab	: Colaboratory
MAE	: Mean Average Error
MSE	: Mean Square Error
NLP	: Natural Language Processing
NER	: Named Entity Recognition
NumPy	: Numerical Python
RMSE	: Root Mean Square Error
RoBERTa	: Robustly optimized BERT approach
SkLearn	: Scikit Learn
SQLite	: Structured Query Language Lite

CHAPTER 1

INTRODUCTION

1.1 Background

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.

1.2 Problem Statement

- Time-consuming and labor-intensive process that requires significant effort.
- Inefficiency in handling large volumes of resumes within tight timelines.
- Difficulty in tracking, organizing, and retrieving specific resumes accurately.
- Difficulty in objectively comparing candidates with diverse backgrounds and experiences.
- Limited scalability for growing applicant pools, leading to bottlenecks in the evaluation process.

1.3 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.

1.4 Application

- By automatically extracting and analyzing critical information from resumes, recruiters may expedite the candidate screening process and swiftly find qualified prospects.
- Assist recruiters in efficiently connecting candidate proficiencies with job requirements by processing resumes for specific talents.
- Candidates can assess how well their resumes align with a given job posted by the recruiter.
- Simplify database administration by automatically adding and updating candidate databases with data from parsed resumes, guaranteeing current and

accurate information for upcoming hiring requirements.

1.5 Features

- Automate extraction and analysis of critical information from resumes.
- Handle large volumes of resumes within tight timelines efficiently.
- Prioritization of candidate resumes based on relevance to specific job descriptions.
- Automatically updates the database with candidate's information extracted from parsed resumes.
- Rank the candidates based on matching between skills, experience, degree and past job using weighted sum algorithm.

CHAPTER 2

RELATED THEORY

2.1 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

2.2 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 preprocessing 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

2.3 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 organizing candidate information from large volumes of resumes.

2.4 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.

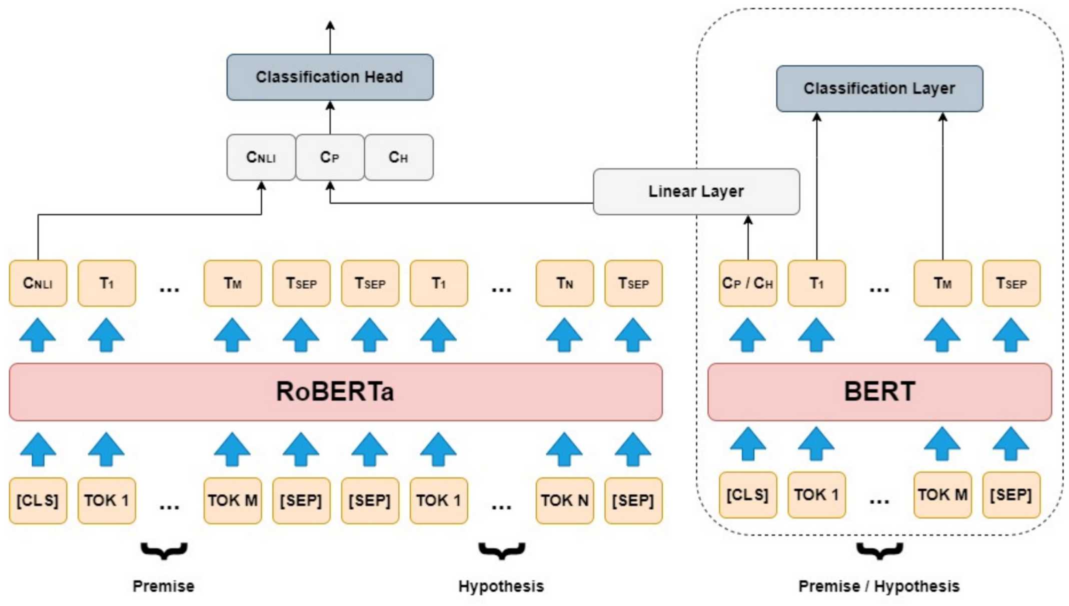


Figure 2.1: Roberta Based Transformer Architecture

Source: Adapted from labeller.com

2.5 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.

2.6 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.

CHAPTER 3

LITERATURE REVIEW

In the realm of Human Resources and recruitment processes, several noteworthy publications have contributed to our understanding of resume parsing and analysis. Ankit Raj and Nidhi Singh's paper, (2022) [1], presents the historical evolution of resume parsing, delves into diverse parsing approaches, and explores the challenges and opportunities within this domain.

Abhishek Gupta and Neha Gupta's paper, (2021)[2]. provides the transformative influence of intelligent resume parsing. The authors contend that such parsing methods can lead to significant time and cost savings, enhance hiring decision accuracy, and uncover hidden talents.

With a shift in emphasis to the methods used, (2020)[3] by Shubham Gupta and Ruchir Mittal introduces a thorough analysis of numerous resume parsing methods, addressing issues and their numerous solutions.

Ayush Agarwal and Mohit Agarwal's work from the same year, (2020)[4] provides an in-depth analysis of the features, cost structures, and advantages and disadvantages of the products that are currently on the market.

Lastly, Pranav Gupta and Rishabh Gupta's publication, (2020)[5] observes the trajectory of resume parsing. The authors contend that as the number of resumes rises, the incorporation of intelligent parsing and analysis is set to become more and more crucial in the future. Collectively, these articles offer a thorough and current summary of the situation, highlighting the development, significance, methods, resources, and possibilities for resume parsing and analysis in the HR and recruitment domains.

CHAPTER 4

METHODOLOGY

4.1 Overview

In our resume ranking process, we have implemented a sophisticated method of assessing applicants' suitability for a particular job. Using the Spacy natural language processing (NLP) model that excels in Named Entity Recognition (NER), we can efficiently extract important information from resumes and job descriptions.

Through NER, the system identifies and categorizes important elements such as experience, skills, degrees, and work history in each document. Each of these entities is assigned a weight based on the importance of the qualifications mentioned in the job description. For example, depending on the nature of the role, skills and experience may weigh more heavily than other factors.

Once the organizations are identified and balanced, our algorithm proceeds to examine the match between the organizations in the applicant's application and those specified in the job description. This assessment considers a variety of factors, including the relevance, depth, and specificity of the identified features. For example, a resume that most closely matches the required skills and experience outlined in the job description will receive a higher score.

Based on this comparison, a score is calculated for each application, reflecting the match between the applicant's qualifications and the job requirements The application with the highest score is prioritized and ranked well, which means it strongly matches position requirements.

By using this comprehensive approach, our system streamlines the recruitment process by identifying candidates whose profile best matches the job requirements. Not only does this save time and resources, it also ensures that the best candidates are selected for the further consideration, ultimately enhancing the quality and efficiency of the hiring process.

4.2 SDLC

Our project successfully implemented the Agile Software Development Methodology as our chosen Software Development Life Cycle (SDLC). Through the division of our project into short sprints, we consistently delivered working software, actively sought feedback from our supervisor, and adapted the project accordingly. This Agile approach not only enabled us to respond promptly to changes but also ensured the early delivery of tangible value in our development lifecycle. The regular feedback loops and continuous improvement processes embedded within Agile contributed to our satisfaction, aligning the final project closely with our expectations. Overall, our utilization of the Agile SDLC significantly contributed to the project's success by optimizing adaptability, collaboration, and the timely delivery of value. Agile Model diagram is shown in figure 4.1.

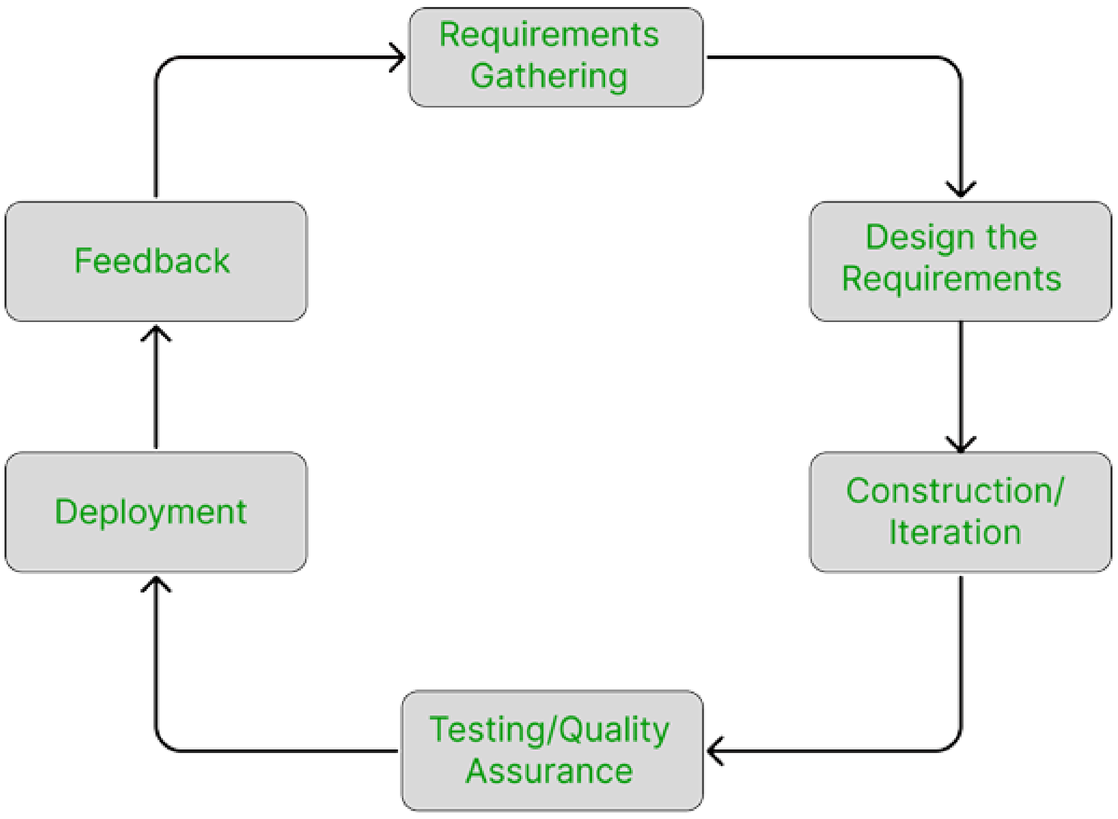


Figure 4.1: Agile Model

4.3 System Architecture

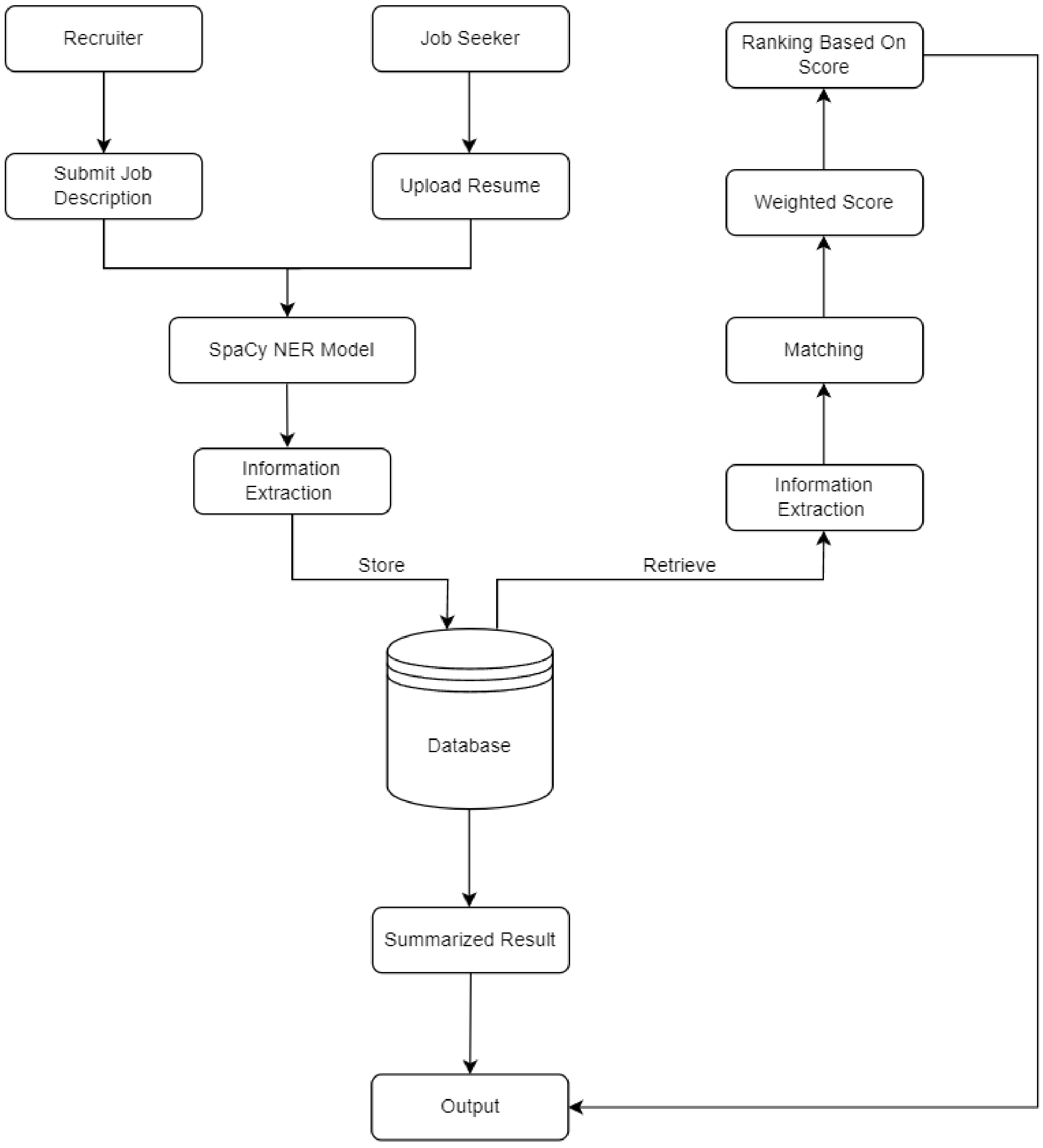


Figure 4.2: System Architecture

The system architecture shown in Figure 4.2 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.

4.4 UML Diagram

4.4.1 Use Case Diagram

Use Case Diagram of our system is shown in figure 4.3.

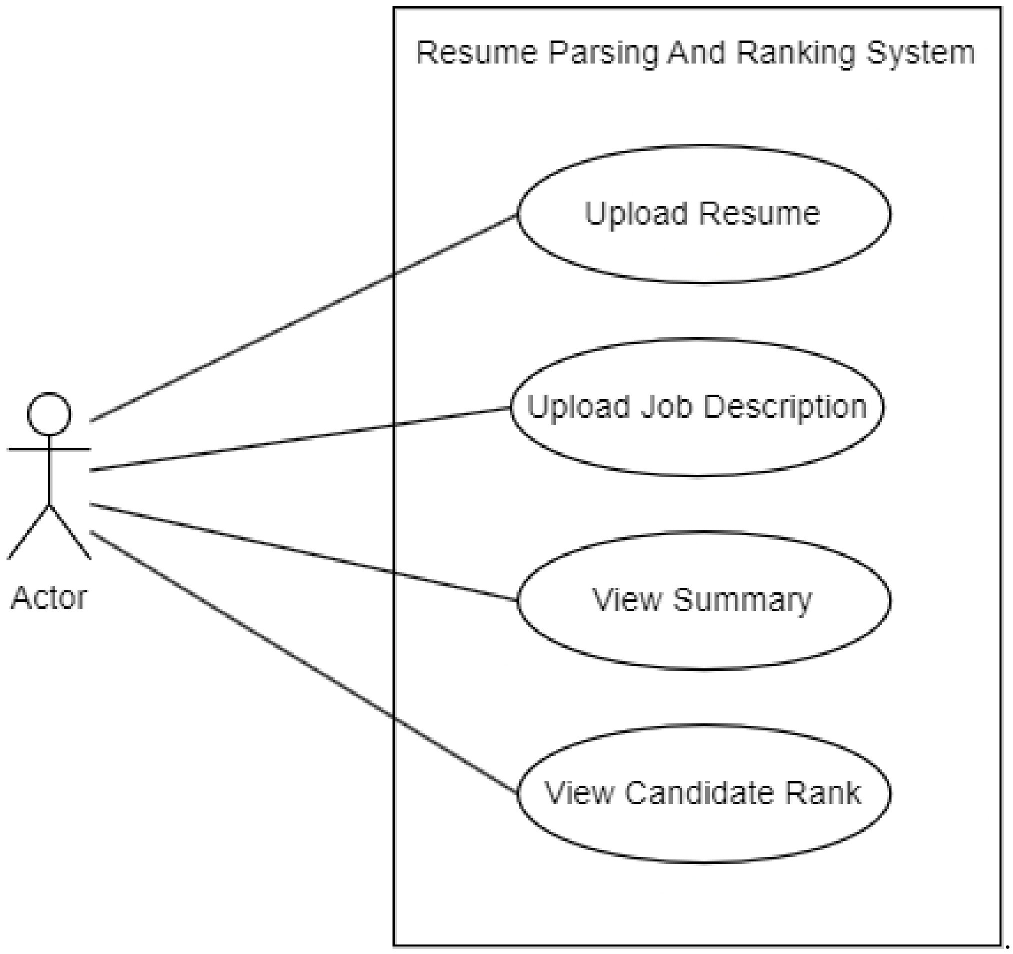


Figure 4.3: Use Case Diagram

4.4.2 Statechart Diagram

Statechart Diagram of our system is shown in figure 4.4.

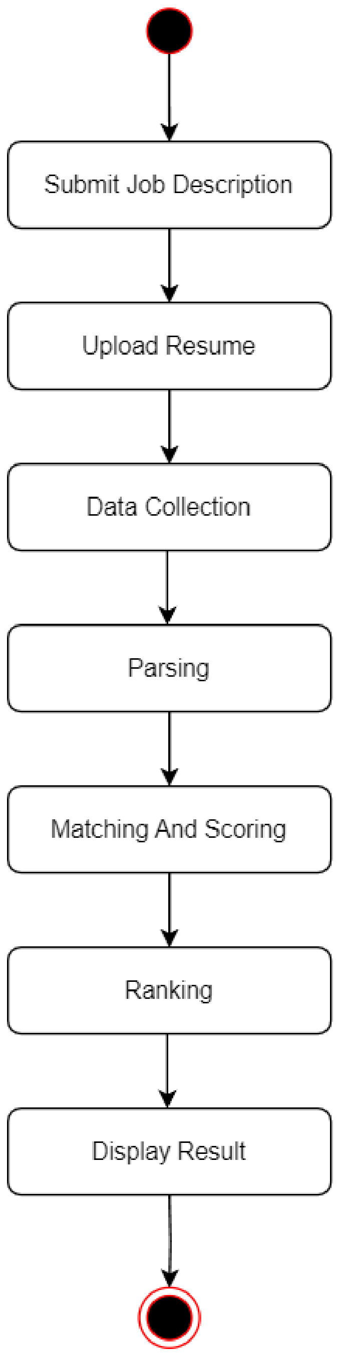


Figure 4.4: Statechart Diagram

4.4.3 Component Diagram

Component Diagram of our system is shown in figure 4.5.

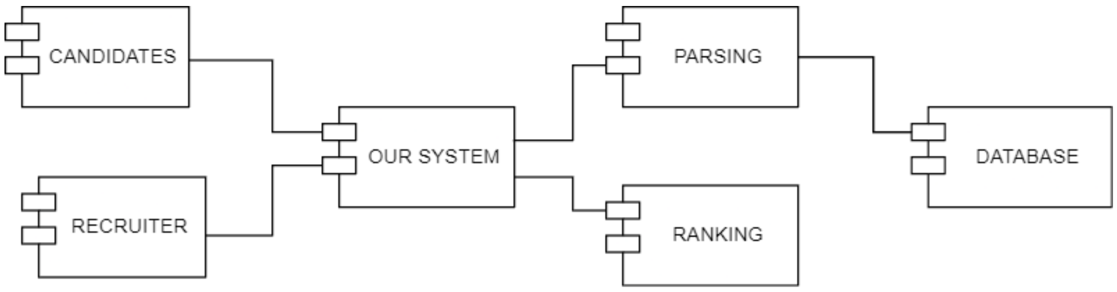


Figure 4.5: Component Diagram

4.5 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.

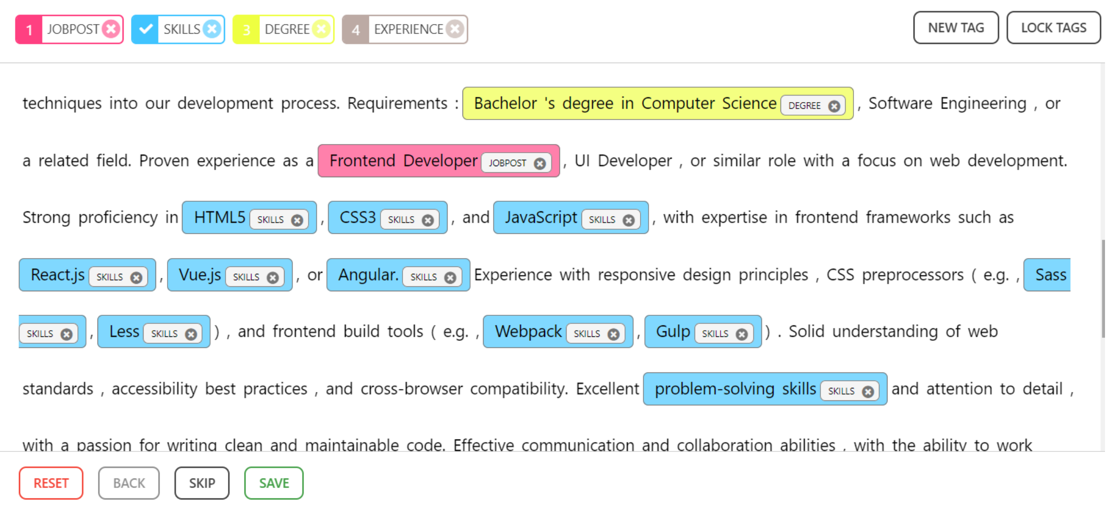
4.6 Data Pre-Processing

4.6.1 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 and symbols, such as HTML tags or non-alphanumeric characters, the text data is streamlined and made more uniform, facilitating easier processing and analysis.

4.6.2 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. Example is shown in figure 4.6.



techniques into our development process. Requirements : Bachelor 's degree in Computer Science DEGREE , Software Engineering , or a related field. Proven experience as a Frontend Developer JOBPOST , UI Developer , or similar role with a focus on web development. Strong proficiency in HTML5 SKILLS , CSS3 SKILLS , and JavaScript SKILLS , with expertise in frontend frameworks such as React.js SKILLS , Vue.js SKILLS , or Angular. SKILLS Experience with responsive design principles , CSS preprocessors (e.g. , Sass SKILLS , Less SKILLS) , and frontend build tools (e.g. , Webpack SKILLS , Gulp SKILLS) . Solid understanding of web standards , accessibility best practices , and cross-browser compatibility. Excellent problem-solving skills SKILLS and attention to detail , with a passion for writing clean and maintainable code. Effective communication and collaboration abilities , with the ability to work

Figure 4.6: NER Annotation

```
["Exciting Opportunity for a Full Stack Web Developer!
\r\n\r\nAre you a versatile and motivated Full Stack Web
Developer eager to tackle diverse challenges and contribu-
te to the development of dynamic web applications? Our for-
ward thinking tech startup is seeking a talented individual
```

to join our team and playa key role in building innovative solutions that delight our users. If you have a passion for coding, a strong foundation in both frontend and backend development, and a knack for problem-solving, then this position is perfect for you!\r\n\r\nResponsibilities:\r\n\r\nDesign, develop, and maintain robust and scalable web applications from concept to deployment.\r\nCollaborate with cross-functional teams, including designers and product managers, to understand project requirements and translate them into technical solutions.\r\nImplement frontend interfaces using modern web technologies such as HTML5, CSS3, JavaScript, and frontend frameworks like React, Angular, or Vue.js.\r\nDevelop backend APIs and services using server-side technologies such as Node.js, Python, Ruby on Rails, or Java.\r\nIntegrate third-party APIs and services to enhance application functionality and user experience. \r\nWrite clean, efficient, and well-documented code, adhering to coding standards and best practices.\r\nConduct code reviews, debug issues, and provide technical support to ensure the stability and performance of web applications.\r\n Requirements:\r\n\r\nBachelor’s degree in Computer Science, Engineering, or a related field.\r\nProven experience as a Full Stack Web Developer with a strong portfolio showcasing your frontend and backend development skills.\r\nProficiency in frontend technologies such as HTML, CSS, JavaScript, and frontend frameworks/libraries (e.g., React, Angular, Vue.js). \r\nExperience with backend development using server-side languages and frameworks (e.g., Node.js, Python/Django, Ruby on Rails, Java/Spring).\r\nFamiliarity with database systems such as MySQL, PostgreSQL, MongoDB, or Firebase.\r\nStrong problem-solving skills and attention to detail, with the ability to debug and troubleshoot complex issues.\r\nExcell-

ent communication and collaboration abilities, with a passion for working in a team-oriented environment.\r\nBonus Points: \r\n\r\nExperience with cloud platforms (e.g., AWS, Azure, Google Cloud) and containerization technologies (e.g., Docker, Kubernetes).\r\nKnowledge of DevOps practices and tools for continuous integration and deployment (CI/CD).\r\n Understandingn of web security principles and best practices. \r\nContributions to open-source projects or participation in developer communities. \r\nIf you are a motivated Full Stack Web Developer with a driveto build innovative web solutions and thrive in a fast-paced, collaborative environment, we want to hear from you! Join us and be part of a team that's shaping the future of web development.Apply now to kickstart your exciting journey with us!\r\n\r\n",]
{"entities":[
[27,51,"JOBPOST"],
[90,114,"JOBPOST"],
[894,899,"SKILLS"],
[901,905,"SKILLS"],
[907,917,"SKILLS"],
[948,953,"SKILLS"],
[955,962,"SKILLS"],
[967,974,"SKILLS"],
[992,996,"SKILLS"],
[1049,1056,"SKILLS"],
[1058,1064,"SKILLS"],
[1066,1079,"SKILLS"],
[1084,1089,"SKILLS"]
[1435,1472,"DEGREE"],
[1531,1555,"JOBPOST"],
[1683,1687,"SKILLS"],
[1689,1692,"SKILLS"],

```
[1694,1704,"SKILLS"],
[1747,1752,"SKILLS"],
[1754,1761,"SKILLS"],
[1763,1769,"SKILLS"],
[1859,1866,"SKILLS"],
[1868,1881,"SKILLS"],
[1883,1896,"SKILLS"],
[1898,1909,"SKILLS"],
[1955,1960,"SKILLS"],
[1962,1972,"SKILLS"],
[1974,1981,"SKILLS"],
[1986,1995,"SKILLS"],
[2004,2026,"SKILLS"],
[2122,2163,"SKILLS"],
[2281,2284,"SKILLS"],
[2286,2291,"SKILLS"],
[2293,2305,"SKILLS"],
[2348,2354,"SKILLS"],
[2356,2366,"SKILLS"],
[2383,2389,"SKILLS"],
[2453,2458,"SKILLS"]
]}]
```

4.6.3 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 4.7.

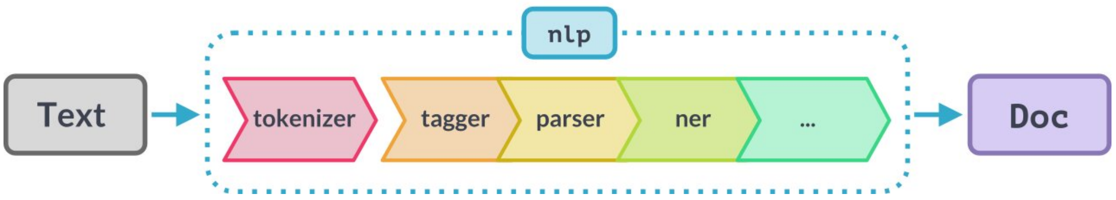


Figure 4.7: Language Processing Pipeline

Source: Adapted from spacy.io

- **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. Example,
  Text: ”I am a software engineer with experience in Java, Python, and SQL.”
  Tokenized Output: [”I”, ”am”, ”a”, ”software”, ”engineer”, ”with”, ”experience”, ”in”, ”Java”, ”,”, ”Python”, ”,”, ”and”, ”SQL”, ”.”]

- **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. Example,
  Text: ”The cats are running.”
  Lemmatized Output: ”The cat be run.”

- **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.
  Tagged Output: [(”I”, ”Personal Pronoun”), (”am”, ”Verb”), (”a”, ”Determiner”), (”software”, ”Noun”), (”engineer”, ”Noun”)]

- **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.
  Dependency Relationships: ”software” is connected to ”engineer” with a modifier relationship, ”engineer” is connected to ”am” with a subject relationship, etc.

- **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.

  Named Entities Identified:
  ”Java”, ”Python”, and ”SQL” are identified as skills.
  ”software engineer” is identified as a job title.
  ”experience” might be identified as a qualification.

4.6.4 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 applications.

4.6.5 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

4.7 Model Training

4.7.1 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.

4.7.2 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.

4.7.3 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 4.8.

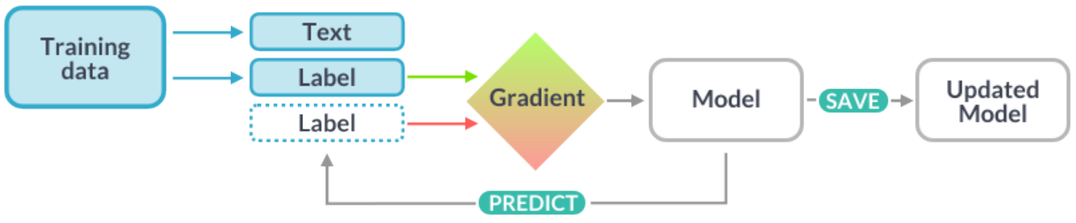


Figure 4.8: NER Training

Source: Adapted from spacy.io

- **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,

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

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

where,
n is the number of classes
y_i is the true label for class i
p_i is the predicted probability for i-th class

4.8 Model Evaluation

In the training process of Named Entity Recognition (NER), precision, recall, and F1-score are evaluated at each step of the evaluation. These metrics provide insights into the model's performance and its ability to accurately identify named entities in text data. By monitoring precision, recall, and F1-score throughout the training process, we ensure that the model continually improves and achieves optimal performance on the task of NER.

Moreover, it's worth noting that there's no need to write additional code for evaluation, as the NER training process inherently provides evaluation metrics such as precision, recall, and F1-score at every step. This built-in evaluation mechanism enables continuous monitoring of the model’s performance without the need for manual intervention

4.8.1 Metrics

- **Precision(P)**: The ratio of correctly predicted named entities to the total predicted named entities.

  $$P = \frac{TP}{TP+FP}$$

  where,
  True Positives (TP) are the named entities correctly identified by the model.
  False Positives (FP) are the named entities incorrectly identified by the model.

- **Recall(R)**: The ratio of correctly predicted named entities to the actual named entities in the text.

  $$R = \frac{TP}{TP+FN}$$

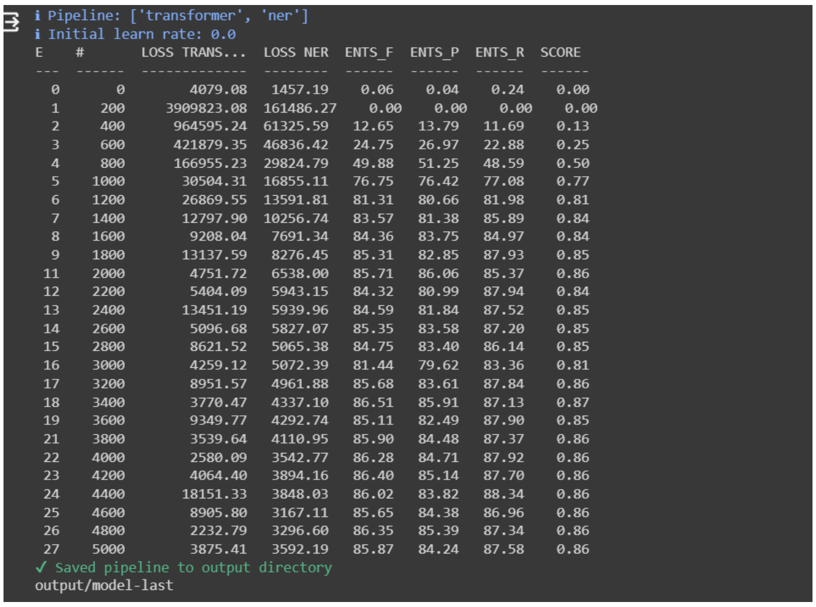
  where,
  True Positives (TP) are the named entities correctly identified by the model.
  False Negatives (FN) are the named entities in the text that were not identified by the model.

- **F1 Score(F1)**: The harmonic mean of precision and recall, providing a balanced evaluation metric.

  $$F1 = \frac{2*P*R}{P+R}$$

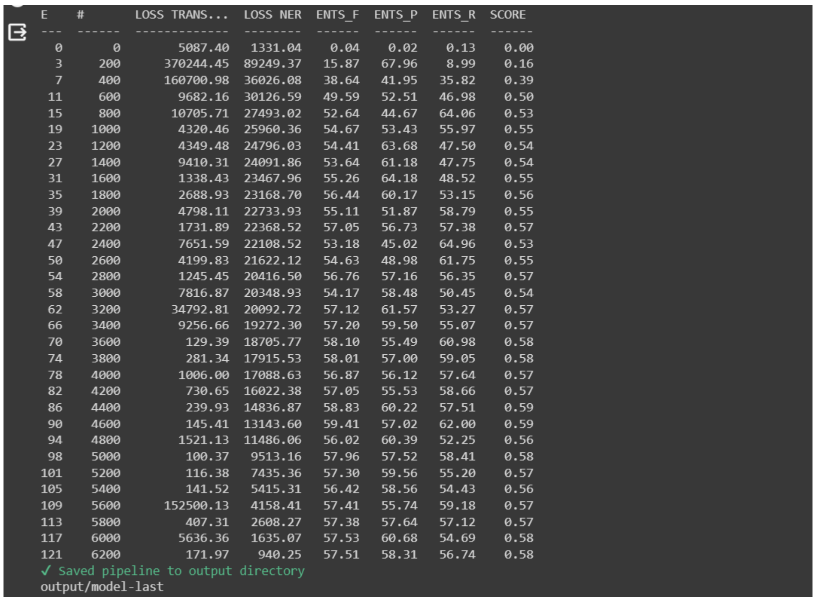
  where,
  P is precision.
  R is recall.

The model training for resume parser model and job description model is shown in figure **??** and figure **??** respectively.



```
ℹ Pipeline: ['transformer', 'ner']
ℹ Initial learn rate: 0.0
E    #       LOSS TRANS...  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ------  -------------  --------  ------  ------  ------  ------
  0       0        4079.08   1457.19    0.06    0.04    0.24    0.00
  1     200     3909823.08 161486.27    0.00    0.00    0.00    0.00
  2     400      964595.24  61325.59   12.65   13.79   11.69    0.13
  3     600      421879.35  46836.42   24.75   26.97   22.88    0.25
  4     800      166955.23  29824.79   49.88   51.25   48.59    0.50
  5    1000       30504.31  16855.11   76.75   76.42   77.08    0.77
  6    1200       26869.55  13591.81   81.31   80.66   81.98    0.81
  7    1400       12797.90  10256.74   83.57   81.38   85.89    0.84
  8    1600        9208.04   7691.34   84.36   83.75   84.97    0.84
  9    1800       13137.59   8276.45   85.31   82.85   87.93    0.85
 11    2000        4751.72   6538.00   85.71   86.06   85.37    0.86
 12    2200        5404.09   5943.15   84.32   80.99   87.94    0.84
 13    2400       13451.19   5939.96   84.59   81.84   87.52    0.85
 14    2600        5096.68   5827.07   85.35   83.58   87.20    0.85
 15    2800        8621.52   5065.38   84.75   83.40   86.14    0.85
 16    3000        4259.12   5072.39   81.44   79.62   83.36    0.81
 17    3200        8951.57   4961.88   85.68   83.61   87.84    0.86
 18    3400        3770.47   4337.10   86.51   85.91   87.13    0.87
 19    3600        9349.77   4292.74   85.11   82.49   87.90    0.85
 21    3800        3539.64   4110.95   85.90   84.48   87.37    0.86
 22    4000        2580.09   3542.77   86.28   84.71   87.92    0.86
 23    4200        4064.40   3894.16   86.40   85.14   87.70    0.86
 24    4400       18151.33   3848.03   86.02   83.82   88.34    0.86
 25    4600        8905.80   3167.11   85.65   84.38   86.96    0.86
 26    4800        2232.79   3296.60   86.35   85.39   87.34    0.86
 27    5000        3875.41   3592.19   85.87   84.24   87.58    0.86
✓ Saved pipeline to output directory
output/model-last
```

Figure 4.9: Model Training and Evaluation for Resume Parsing Model



```
E    #       LOSS TRANS...  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ------  -------------  --------  ------  ------  ------  ------
  0       0        5087.40   1331.04    0.04    0.02    0.13    0.00
  3     200      370244.45  89249.37   15.87   67.96    8.99    0.16
  7     400      160700.98  36026.08   38.64   41.95   35.82    0.39
 11     600        9682.16  30126.59   49.59   52.51   46.98    0.50
 15     800       10705.71  27493.02   52.64   44.67   64.06    0.53
 19    1000        4320.46  25960.36   54.67   53.43   55.97    0.55
 23    1200        4349.48  24796.03   54.41   63.68   47.50    0.54
 27    1400        9410.31  24091.86   53.64   61.18   47.75    0.54
 31    1600        1338.43  23467.96   55.26   64.18   48.52    0.55
 35    1800        2688.93  23168.70   56.44   60.17   53.15    0.56
 39    2000        4798.11  22733.93   55.11   51.87   58.79    0.55
 43    2200        1731.89  22368.52   57.05   56.73   57.38    0.57
 47    2400        7651.59  22108.52   53.18   45.02   64.96    0.53
 50    2600        4199.83  21622.12   54.63   48.98   61.75    0.55
 54    2800        1245.45  20416.50   56.76   57.16   56.35    0.57
 58    3000        7816.87  20348.93   54.17   58.48   50.45    0.54
 62    3200       34792.81  20092.72   57.12   61.57   53.27    0.57
 66    3400        9256.66  19272.30   57.20   59.50   55.07    0.57
 70    3600         129.39  18705.77   58.10   55.49   60.98    0.58
 74    3800         281.34  17915.53   58.01   57.00   59.05    0.58
 78    4000        1006.00  17088.63   56.87   56.12   57.64    0.57
 82    4200         730.65  16022.38   57.05   55.53   58.66    0.57
 86    4400         239.93  14836.87   58.83   60.22   57.51    0.59
 90    4600         145.41  13143.60   59.41   57.02   62.00    0.59
 94    4800        1521.13  11486.06   56.02   60.39   52.25    0.56
 98    5000         100.37   9513.16   57.96   57.52   58.41    0.58
101    5200         116.38   7435.36   57.30   59.56   55.20    0.57
105    5400         141.52   5415.31   56.42   58.56   54.43    0.56
109    5600      152500.13   4158.41   57.41   55.74   59.18    0.57
113    5800         407.31   2608.27   57.38   57.64   57.12    0.57
117    6000        5636.36   1635.07   57.53   60.68   54.69    0.58
121    6200         171.97    940.25   57.51   58.31   56.74    0.58
✓ Saved pipeline to output directory
output/model-last
```

Figure 4.10: Model Training and Evaluation for Job Description Parsing Model

The model evaluation for the resume parser and job description is shown in figure 4.11, figure 4.12, figure 4.13, figure 4.14, figure 4.15, figure 4.16 respectively.

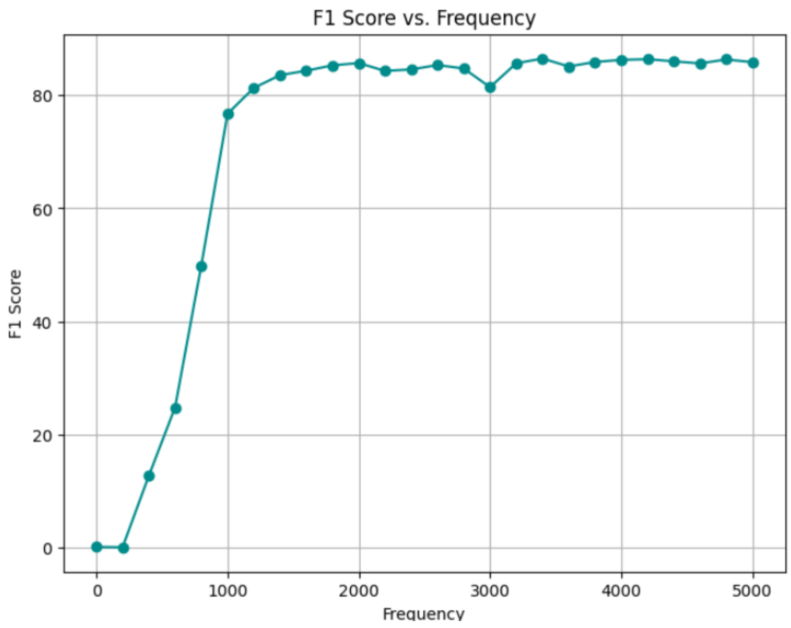


Figure 4.11: F1 Score vs frequency

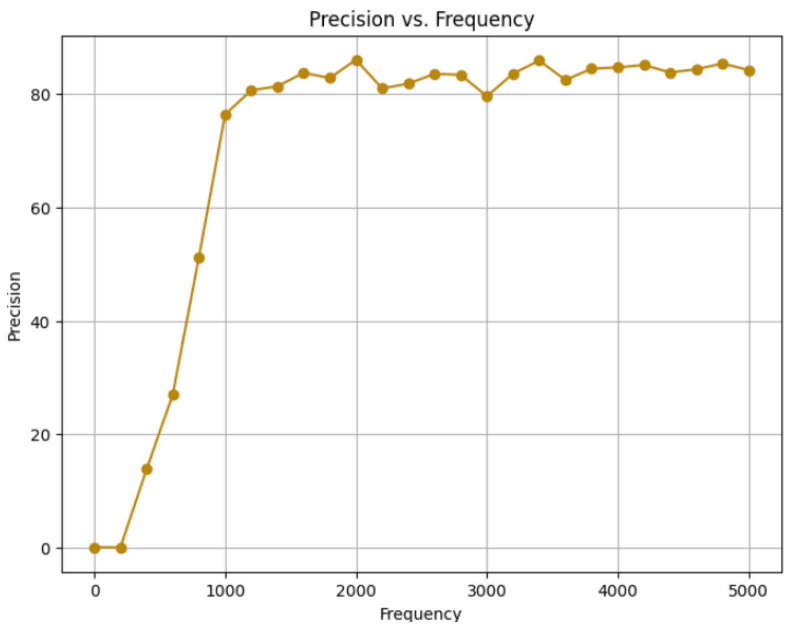


Figure 4.12: Precision vs Frequency

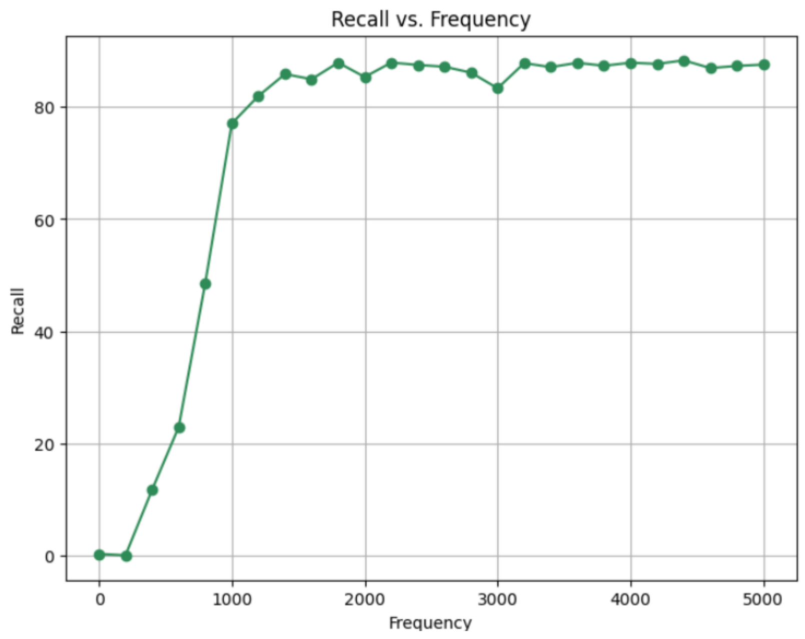


Figure 4.13: Recall vs Frequency

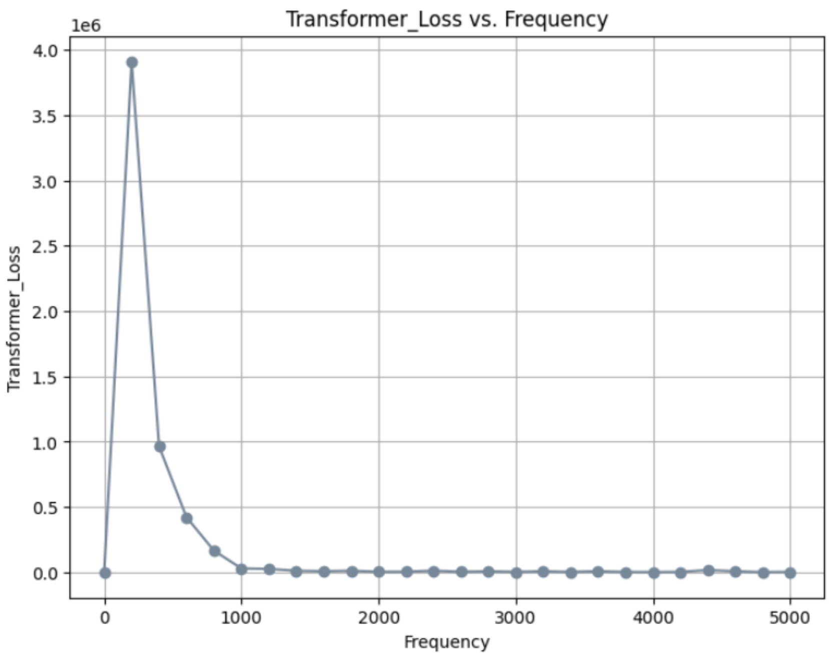


Figure 4.14: Transformer Loss vs Frequency

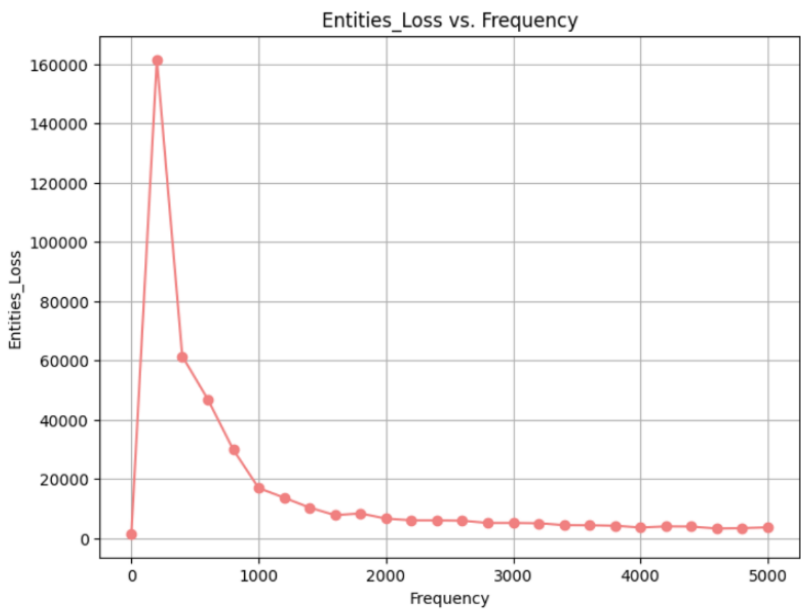


Figure 4.15: Entities Loss vs Frequency

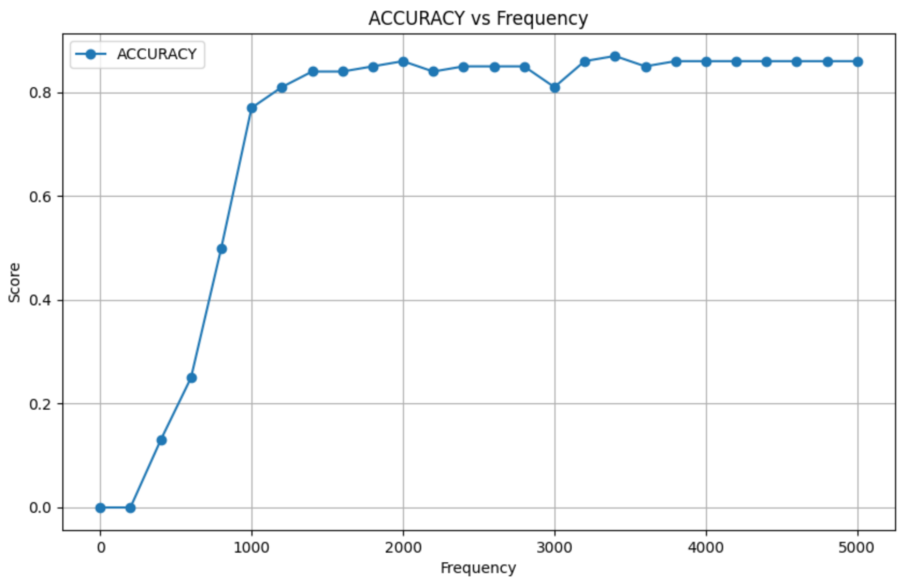


Figure 4.16: Accuracy vs Frequency

4.8.2 Testing

- **Unit Tests**: In this step individual components, including model, frontend, and backend functionalities, are tested in isolation to verify their correctness. To assess the performance of our Named Entity Recognition (NER) resume parsing model, we conducted resume parser model testing and this test case involved comparing the model's predictions with the ground truth annotations for a single example. The example, sourced from an unseen resume in our dataset, contained information such as the candidate's name, educational background, skills, and experience. Using assertions, we verified the correctness of predicted entities and calculated evaluation metrics such as precision, recall, and F1-score.

  While it's essential to note that the evaluation of a single data point may not fully represent the overall performance of a model, it can provide valuable insights into its behavior. In this context, the evaluation conducted on the provided example serves as an illustration of the model's capabilities rather than a comprehensive assessment. It's crucial to emphasize that we have previously evaluated the model's performance using a diverse test dataset(30 % of total data) to obtain a more comprehensive understanding of its effectiveness and generalization capabilities. This particular instance is used here solely for illustrative purposes and does not constitute a complete evaluation of the model.

```
predicted entities={
    "Name": "Sushil Sharma",
    "Degree": "Bachelor of Science in Computer Science",
    "Years of experience": "4 years",
    "College Name": "Tribhuvan University",
    "Skills": "Languages: HTML, CSS, JavaScript, PHP;
               Web Technologies: React.js, Angular,
               Node.js; Database Management: MySQL,
               MongoDB; Frameworks: Laravel, Express;
               Version Control: Git, GitHub;
               Other Tools: Webpack, npm, Docker;
               Design Tools: Adobe Photoshop, Figma",
    "Designation": "Web Developer",
    "Degree": "E-commerce"
}


actual entities = {
    "Name": "Sushil Sharma",
    "Location": "Kathmandu",
    "Email": "sushil789@gmail.com",
    "Years of experience": "4 years",
    "Degree": "Bachelor of Science in Computer Science",
    "College Name": "Tribhuvan University",
    "Skills": "HTML, CSS, JavaScript, PHP;
               Web Technologies: React.js,Angular,
               Node.js; Database Management: MySQL,
               MongoDB; Frameworks: Laravel, Express;
               Version Control: Git, GitHub;
               Other Tools: Webpack, npm, Docker;
               Design Tools: Adobe Photoshop, Figma",
    "Designation": "Web Developer",
    "Degree": "E-commerce",

}
```

Given the following parameters:

- True Positives (TP): 7(Name, Degree, College Name, Skills, Designation, Years of experience, Degree)
- True Negatives (TN):0
- False Positives (FP): 0
- False Negatives (FN): 2(email, location)

We can calculate the precision, recall, and F1-score as follows:

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{7}{7+0} = \frac{7}{7} \approx 1$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{7}{7+2} = \frac{7}{9} \approx 0.77$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \approx 0.87$$

The accuracy can be calculated as:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{7+0}{7+0+0+2} = \frac{9}{11} \approx 0.818$$

- **Integration Tests**: Integration testing is a crucial phase following unit testing, where individual components such as frontend, backend, and models are tested in concert. We have conducted integration testing to ensure that these different parts of the system work together seamlessly, facilitating smooth communication and coherent functionality across the software ecosystem. By merging and testing various modules, integration testing identifies and resolves integration-related issues, bolstering the reliability and robustness of the software product. This phase is essential for preemptively detecting and addressing any glitches that may arise from integrating disparate parts, ultimately leading to the creation of a stable and dependable software solution

4.8.3 Optimization

The optimization process involves fine-tuning the model parameters to minimize a predefined loss function, thereby enhancing the model's ability to correctly identify and classify entities within resumes. In our resume parsing project, the Adam optimizer, along with other optimization techniques like gradient clipping and warmup linear schedule, plays a pivotal role in training our NER model to achieve high accuracy and robustness in extracting information from resumes. The formula for updating the parameters using the Adam optimizer is as follows:

$$
\begin{aligned}
m_t &= \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \\
v_t &= \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\
\theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t}+\varepsilon} \cdot \hat{m}_t
\end{aligned}
$$

Where,

m_t: First moment estimate of the gradients.

v_t: Second moment estimate of the gradients.

$\hat{m}_t$: Bias-corrected first moment estimate.

$\hat{v}_t$: Bias-corrected second moment estimate.

θ_t : Updated parameters.

η : Learning rate.

β_1: Exponential decay rate for the first moment estimates.

β_2: Exponential decay rate for the second moment estimates.

g_t: Gradient of the loss function with respect to the parameters.

t: Time step or iteration number.

ε: Small constant added to the denominator for numerical stability.

4.9 Ranking Algorithm with Weighted Score

Step 1: Initialize

- Create an empty list named `Ranked Resumes` to store prioritized resumes.

Step 2: Retrieve Data

- Fetch all resumes and job descriptions from the respective databases.
- Create structured entities for each resume (`resume_entities`) and job

description (`job_description_entities`).

Step 3: Normalization Functions

- Define functions for normalizing data points:
  - `normalize_years_of_experience`: Extracts and normalizes years of experience.
  - `normalize_skills`: Matches and counts the skills present in the resume and required by the job.
  - `normalize_degree`: Check if the resume degree matches the required degree.
  - `normalize_worked_as`: Checks if the worked job title matches the required job title.

Step 4: Score Calculation Function: `calculate_score`

- Receive a resume, job description, and weights as parameters.
- Normalize various data points using the defined normalization functions.
- Calculate an overall score using the weighted sum of normalized criteria.
- Return the calculated score.

Step 5: Ranking Function: `rank_resumes`

- Initialize an empty list named `Ranked Resumes`.
- Iterate through each resume and job description.
- For each pair, call `calculate_score` to compute the score.
- Append relevant details (resume ID, name, email, score, job title) to the `Ranked Resumes` list.
- Sort the list in descending order based on scores.

Step 6: Weight Assignment:

- Define a dictionary named `weights` with assigned weights for different criteria:
  - 'experience': 0.3

    - 'skills': 0.4
    - 'degree': 0.05
    - 'worked_as': 0.25
- Adjust weights based on the importance of each criterion.

Step 7: Rank Resumes:

- Call the `rank resumes` function with `resume entities`, `job description entities`, and `weights` as arguments.
- Receive a list of ranked resumes in descending order of scores.

Step 8: Print and Return:

- Print the ranked resumes with relevant details (ID, name, email, job title, score).
- Return the ranked resumes as a JSON response.

4.10 Implementation of the proposed system design

4.10.1 Frontend (ReactJS)

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React allows developers to create applications by creating reusable components that you can think of as independent Lego blocks. These components are individual pieces of a final interface, which, when assembled, form the application's entire user interface.

We utilized the power of React to develop a user-friendly interface for resume and job description submission to ensure seamless interaction with the backend for data processing. Additionally, we made an interface for displaying the ranking of resumes, to enhance the overall user experience.

4.10.2 Backend (Django)

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It is maintained by the Django Software Foundation (DSF), an independent organization established in the US as a non-profit. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. Django follows the model–template–views (MTV) architectural pattern.

In our project, we made use of the Django restframework to set up API endpoints for receiving resumes, job description data, and resume ranking. SpaCy model is integrated for NLP processing of resume and job description texts.

4.10.3 DataBase(SQLite)

SQLite is a database engine written in the C programming language. It is not a standalone app; rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases. It is the most widely deployed database engine, as it is used by several of the top web browsers, operating systems, mobile phones, and other embedded systems. SQLite was designed to allow the program to be operated without installing a database management system or requiring a database administrator.

We've set up a SQLite database, featuring distinct tables dedicated to housing extracted data from resumes and job descriptions. This database also plays a crucial role in seamlessly manipulating and presenting data to the front, ensuring a streamlined and effective data management process

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Output

The figure for output is shown in figure**??**, figure**??**, figure**??**, figure**??** and figure**??**.

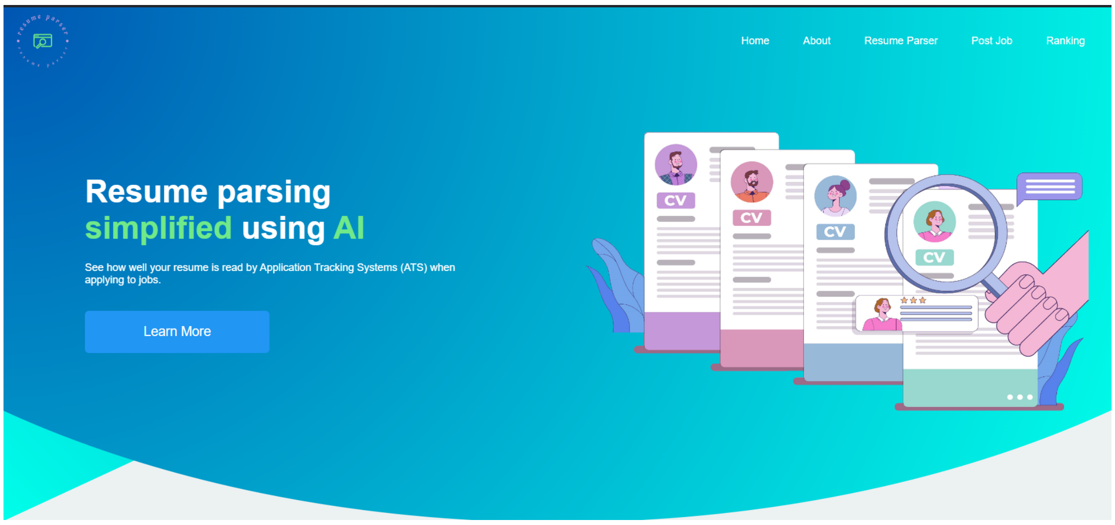

Home
About
Resume Parser
Post Job
Ranking
Resume parsing simplified using AI
See how well your resume is read by Application Tracking Systems (ATS) when applying to jobs.
Learn More
CV

Figure 5.1: Landing Page Interface

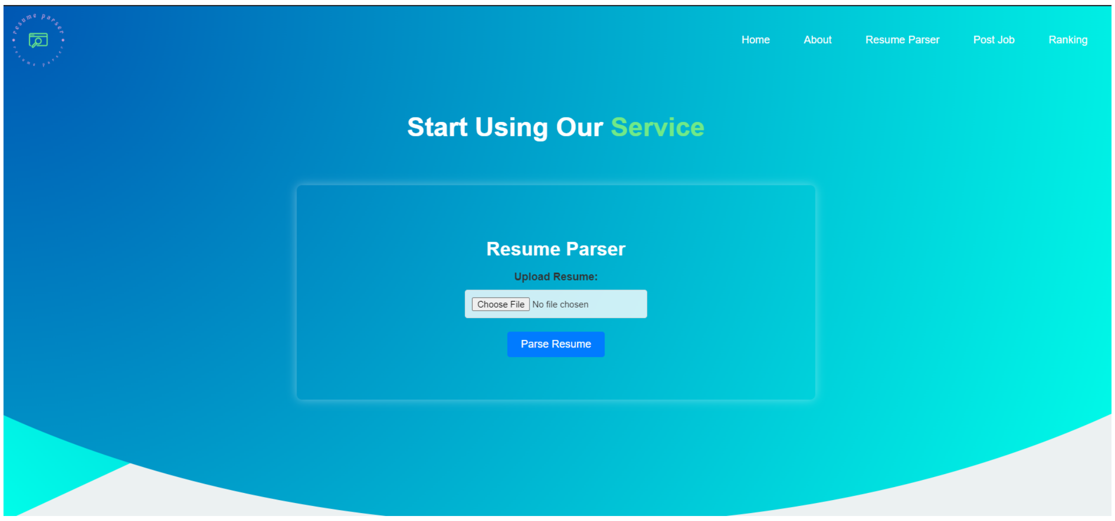
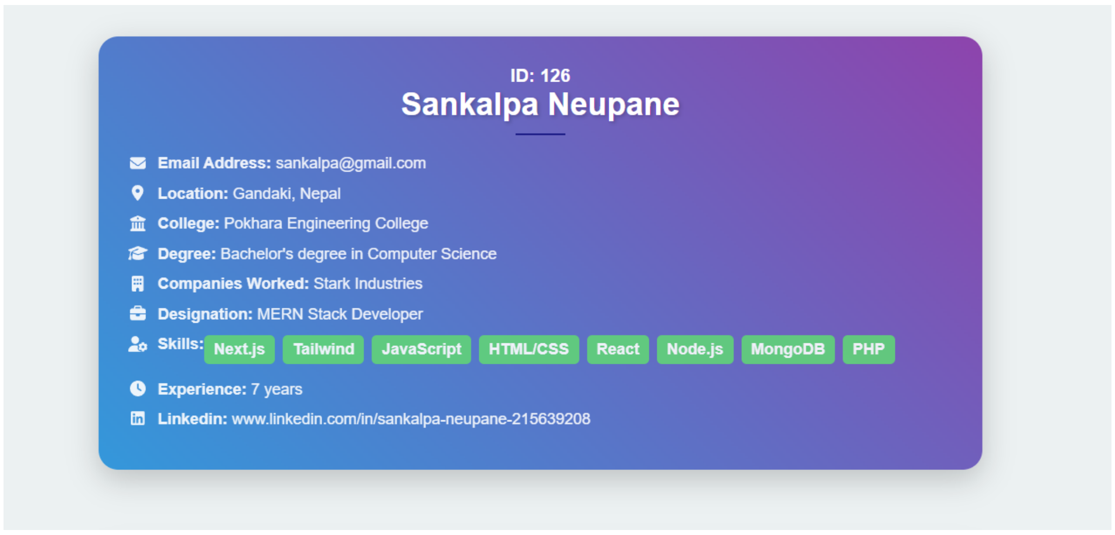


Figure 5.2: Parsing Page Interface



ID: 126

Sankalpa Neupane

Email Address: sankalpa@gmail.com

Location: Gandaki, Nepal

College: Pokhara Engineering College

Degree: Bachelor's degree in Computer Science

Companies Worked: Stark Industries

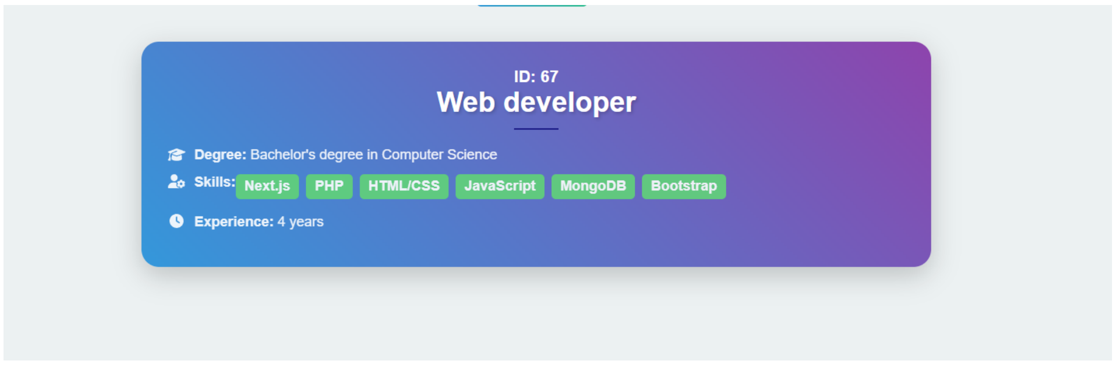
Designation: MERN Stack Developer

Skills: Next.js Tailwind JavaScript HTML/CSS React Node.js MongoDB PHP

Experience: 7 years

Linkedin: www.linkedin.com/in/sankalpa-neupane-215639208

Figure 5.3: Extracted Entities from Resume Sample



ID: 67

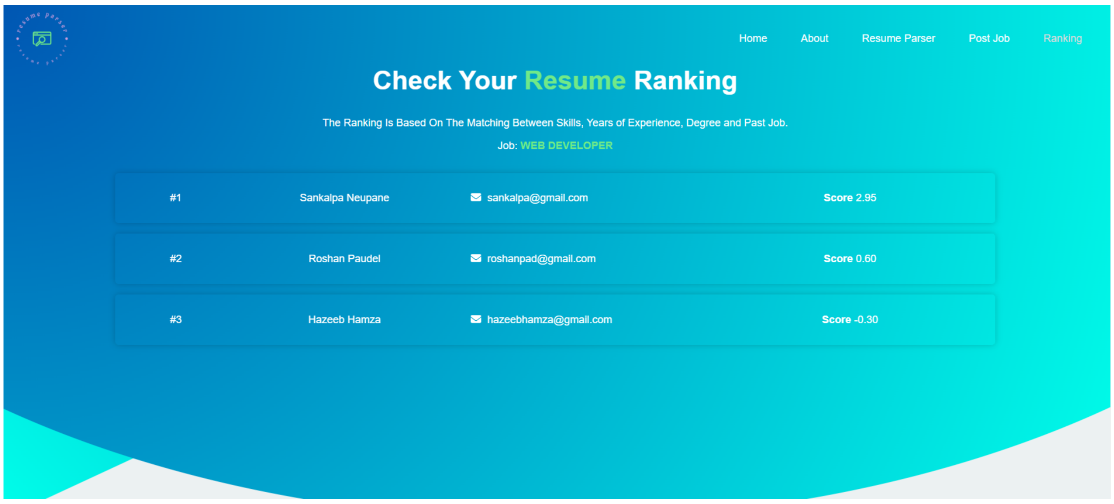
Web developer

Degree: Bachelor's degree in Computer Science

Skills: Next.js PHP HTML/CSS JavaScript MongoDB Bootstrap

Experience: 4 years

Figure 5.4: Extracted Entities from Job Description



Check Your Resume Ranking

The Ranking Is Based On The Matching Between Skills, Years of Experience, Degree and Past Job.

Job: WEB DEVELOPER

#1	Sankalpa Neupane	sankalpa@gmail.com	**Score** 2.95
#2	Roshan Paudel	roshanpad@gmail.com	**Score** 0.60
#3	Hazeeb Hamza	hazeebhamza@gmail.com	**Score** -0.30

Figure 5.5: Ranking Page Interface

5.2 Discussion

The RoBERTa model and the integration of Named Entity Recognition (NER) into our resume parser represent a major advance in natural language processing (NLP). Using state-of-the-art techniques and models, our program aims to change the way resumes are evaluated and processed.

Known for its skills in writing under context, the RoBERTa example is the backbone of our resume parser. With bidirectional context-recognition capabilities, RoBERTa enables our system to extract nuanced information from resumes with unparalleled accuracy. By training extensive datasets and fine-tuning its parameters, RoBERTa ensures that our parser captures both the semantic and syntactic nuances of resume data.

The addition of Named Entity Recognition (NER) with RoBERTa further enhances the capabilities of our parser. NER allows our system to identify and categorize specific elements of a resume, such as name, skills, and educational qualifications. This allows recruiters to quickly identify relevant information and make informed decisions when selecting candidates. Furthermore, NER integration enables our system to adapt to different iterative formats and structures, making it robust to perform on different data sets.

Through rigorous testing and validation, our integrated approach has demonstrated exceptional performance. Combining RoBERTa with NER significantly improved parsing accuracy, our system achieved impressive levels of accuracy and recall.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In conclusion, our study on resume ranking and parsing using spaCy NER reveals promising results. Through rigorous training, our resume parsing model achieves a commendable accuracy of 87% after 5000 training steps, demonstrating its ability to accurately extract entities such as name, degree, skills, and designation from resumes. The convergence of transformer loss and NER loss from high to low values further validates the effectiveness of our model in understanding and parsing resumes.

However, when it comes to job description parsing, the accuracy converges to only 59%, primarily due to the limitations of a small and less diverse dataset. The transformer loss and NER loss exhibit similar convergence patterns, starting from relatively high values and gradually decreasing over epochs until stabilization.

To address the disparity between resume and job description parsing accuracies, we leverage a weighted sum method for entity comparison. By assigning appropriate weights to each entity extracted from both resumes and job descriptions, we effectively rank resumes based on their relevance to the job requirements. This innovative approach enhances the accuracy and relevance of our resume ranking system, thereby facilitating efficient talent acquisition and recruitment processes. explain the mentioned loss below in the above paragraph: explain the conclusion for resume ranking and parsing using spacy ner where, while training our resume parsing model the accuracy converges to a maximum 87% for the best model while training in 5000 steps similarly transformer loss and ner loss converges to low value from high for resume. similarly, the accuracy for description covers only 59% due small dataset and not diverse dataset while transformer loss and ner loss start from 5087.40 and 1331.04 similarly for the next 200 wpoch it increases to 370244.45 and 89249.37 and finally ate 6200 epochs it converges to 171.97 and 940.25. Mentioning these things and additional by comparing the entities of resume and job description using weighted sum method ranking is done.

6.2 Future Work

Our project, Smart Resume Parser, Ranker, and Analyzer, lays out many options for future careers and development. To maximize the effectiveness of the system, we propose the following areas for exploration:

- **Enhanced Candidate Screening:** Develop advanced candidate screening algorithms and methods to reduce the risk of potential voters being overlooked. This may include using fuzzy matching techniques, semantic similarity algorithms, or incorporating feedback mechanisms to refine the system candidate evaluation process.
- **Language and Wording Adaptation:** Enhance systems understanding of language variations by incorporating advanced natural language processing techniques. This includes an analysis of strategies for translating common languages and terms used in the resume to improve the extraction of information.
- **User Feedback Mechanism:** Use a user feedback system to gather feedback from recruiters and hiring managers on program performance. By collecting information on the accuracy, utility, and usefulness of the analyzed iteration, we can iteratively improve the design and meet the user's specific needs and preferences.
- **Enhanced Technical Stability:** Improve the technical stability and reliability of the parsing software to minimize the impact of technical difficulties on data extraction and analysis. This includes implementing robust error handling mechanisms, conducting thorough testing, and regular updates to address compatibility issues and ensure smooth operation.

REFERENCES

[1] A. 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.

[2] A. 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.

[3] S. Gupta and R. Mittal, "A survey on resume parsing techniques," *Journal of Big Data*, vol. 7, no. 1, p. 13, 2020.

[4] A. Agarwal and M. Agarwal, "A review of resume parsing and analysis tools," *International Journal of Information Management*, vol. 49, p. 102070, 2020.

[5] 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.

APPENDIX

APPENDIX A: Sample Dataset

```
{
    "text": "Ramu Roy is a skilled data scientist with
    expertise in machine learning.",
    "labels": [
        {"start": 0, "end": 8, "label": "PERSON"},
        {"start": 32, "end": 46, "label": "JOB_TITLE"},
        {"start": 50, "end": 68, "label": "SKILL"},
        {"start": 72, "end": 88, "label": "SKILL"}
    ]
},
{
    "text": "XYZ Corporation is seeking a software
    developer proficient in Python and Java.",
    "labels": [
        {"start": 0, "end": 16, "label": "ORG"},
        {"start": 27, "end": 43, "label": "JOB_TITLE"},
        {"start": 57, "end": 63, "label": "SKILL"},
        {"start": 68, "end": 74, "label": "SKILL"}
    ]
},
{
    "text": "Sara Dhakal, a UX/UI designer, has a passion
    for creating intuitive user experiences.",
    "labels": [
        {"start": 0, "end": 10, "label": "PERSON"},
        {"start": 18, "end": 30, "label": "JOB_TITLE"},
        {"start": 55, "end": 66, "label": "SKILL"},
        {"start": 76, "end": 88, "label": "SKILL"}
    ]
```

```
  },
  {
  "text": "John Smith is a software developer specializing
  in machine learning.",
  "labels": [
    {"start": 0, "end": 10, "label": "PERSON"},
    {"start": 22, "end": 38, "label": "JOB_TITLE"},
    {"start": 56, "end": 75, "label": "SKILL"},
    {"start": 82, "end": 99, "label": "SKILL"}
  ]
}
```