**IBM Project SRS Report**

**Named Entity Recognition Using Transformers architecture**

**Team Members: Guided By:**

Raj Kariya (21162101011) Prof Pritesh Andharia (Internal)

Jhanvi Patel (22162122008) Prof Nirav Rajgor (IBM Guide)

Dushyant Vyas (22162172006)

### 

### **1. Introduction**

#### **1.1 Overview of the Project**

Named Entity Recognition (NER) is a crucial task in natural language processing (NLP) that involves identifying and classifying entities such as names, locations, organizations, dates, and more within unstructured text. In this project, we aim to leverage the \_power of transformer-based models to develop an efficient NER system specifically tailored for processing resumes. The project integrates Python for the model, Django as the frontend framework, and MongoDB as the database to create a robust and user-friendly application.

#### **1.2 Problem Statement**

Human Resource (HR) departments often face the daunting task of evaluating vast numbers of resumes to shortlist candidates. Many resumes contain excessive and irrelevant information, making manual evaluation time-consuming and error-prone. Existing automated tools may lack precision or fail to extract relevant details efficiently. This project aims to address these challenges by building a transformer-based NER model that can accurately extract key information from resumes, such as names, addresses, qualifications, and work experience, thereby streamlining the candidate shortlisting process.

#### **1.3 Objectives**

The primary objectives of this project are:

1. To develop a transformer-based NER model capable of accurately identifying key entities in resumes.
2. To design a user-friendly web interface using Django for uploading and processing resumes.
3. To store and manage extracted information effectively using MongoDB.
4. To simplify and accelerate the candidate shortlisting process for HR departments.
5. To ensure scalability and adaptability of the system for future enhancements.

#### **1.4 Scope of the Project**

This project focuses on creating a pipeline for automatic resume processing using NER. Key functionalities include:

* Uploading resumes in various formats (e.g., PDF, DOCX).
* Extracting essential information such as name, contact details, skills, qualifications, and work experience.
* Displaying extracted data on a dashboard for easy evaluation.
* Storing and retrieving data efficiently using MongoDB.
* Providing APIs for integration with other HR tools or platforms.

While the current scope is limited to resumes in English, future extensions may include support for multilingual resumes and additional entity types. The system is designed to be modular, ensuring ease of maintenance and scalability.

### 

### 

### 

### 

### 

### 

### 

### 

### **2. Background and Literature Review**

#### **2.1 What is Named Entity Recognition (NER)?**

Named Entity Recognition (NER) is a subtask of natural language processing (NLP) that focuses on identifying and categorizing entities within text into predefined categories such as names of persons, organizations, locations, dates, percentages, and more. NER plays a vital role in extracting structured information from unstructured text, enabling better analysis and decision-making across various domains.

#### **2.2 Importance of NER in Resume Evaluation**

In the context of resume evaluation, NER can:

* Extract relevant details such as candidate names, contact information, educational qualifications, skills, and work experience.
* Reduce the time and effort required for manual resume screening.
* Ensure consistency and objectivity in the evaluation process.
* Enable efficient storage and retrieval of candidate information for future reference. By automating the extraction of key details, NER significantly enhances the efficiency and accuracy of the recruitment process.

#### **2.3 Evolution of NER Models**

The development of NER models has evolved over time:

1. Rule-based Systems: Early NER systems relied on handcrafted rules and pattern-matching techniques. While effective for specific use cases, these systems lacked scalability and adaptability.
2. Machine Learning Models: With the advent of machine learning, algorithms like Conditional Random Fields (CRFs) and Hidden Markov Models (HMMs) were used to train NER models on labeled datasets, improving accuracy and flexibility.
3. Deep Learning Models: Neural networks, particularly recurrent neural networks (RNNs) and long short-term memory networks (LSTMs), brought significant advancements by capturing contextual information in text.
4. Transformer-based Models: Recent innovations like BERT (Bidirectional Encoder Representations from Transformers) and RoBERTa (Robustly Optimized BERT Pretraining Approach) have revolutionized NER by leveraging self-attention mechanisms to process and understand text more effectively.

#### **2.4 Overview of Transformer Models (e.g., BERT, RoBERTa)**

Transformers are a class of deep learning models designed to handle sequential data efficiently. Key features include:

* Self-Attention Mechanism: Enables the model to focus on relevant parts of the input text, capturing contextual relationships between words.
* Bidirectional Context Understanding: Models like BERT process text in both forward and backward directions, providing a deeper understanding of context.
* Pretraining and Fine-tuning: Transformer models are pretrained on large corpora and can be fine-tuned for specific tasks, including NER.
* State-of-the-Art Performance: Transformers have achieved superior performance across various NLP tasks, including NER, due to their ability to handle complex linguistic patterns.

#### **2.5 Existing Solutions and Their Limitations**

Several tools and frameworks exist for NER, including spaCy, Stanford NER, and Hugging Face Transformers. While these tools provide robust solutions, they have limitations:

* Generic Models: Pretrained models may not perform optimally on domain-specific tasks like resume processing without fine-tuning.
* Scalability Issues: Some solutions struggle to handle large-scale data efficiently.
* Lack of Customization: Limited flexibility to adapt to specific requirements or integrate with existing systems.
* Language Constraints: Many models are designed primarily for English, limiting their applicability for multilingual resumes.

This project addresses these limitations by fine-tuning transformer models for domain-specific NER tasks and integrating them into a scalable and user-friendly system.

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **3. System Requirements**

#### **3.1 Hardware Requirements**

To ensure smooth development and deployment of the NER system, the following hardware specifications are recommended:

* Processor: Intel i5 or higher / AMD Ryzen 5 or higher
* RAM: 8 GB (16 GB recommended for training models)
* Storage: 100 GB free disk space
* GPU: NVIDIA GPU with CUDA support (e.g., NVIDIA GTX 1660 or higher) for faster model training

#### **3.2 Software Requirements**

The system requires the following software components:

* Operating System: Windows 10, macOS, or Linux
* Python: Version 3.8 or higher
* Django: Version 4.0 or higher
* MongoDB: Version 5.0 or higher
* CUDA Toolkit: Required for GPU acceleration
* Libraries and Frameworks:
  + Hugging Face Transformers
  + spaCy
  + NumPy
  + pandas
  + scikit-learn
  + PyTorch or TensorFlow

#### **3.3 Tools and Libraries Used**

The project utilizes the following tools and libraries:

* Frontend Development:
  + Django Templates
  + HTML, CSS, JavaScript
* Backend Development:
  + Django REST Framework (DRF) for API development
  + Python for integrating the NER model
* Database:
  + MongoDB for storing and retrieving extracted resume data
* NER Model Development:
  + Hugging Face Transformers for building and fine-tuning the model
  + PyTorch for training the model
* Testing and Deployment:
  + Postman for API testing
  + Docker for containerization
  + Cloud platforms (e.g., AWS, Heroku) for deployment

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **4. System Design**

#### **4.1 Architecture Diagram**

The system architecture consists of the following components:

1. Frontend: Built using Django to provide a user-friendly interface for uploading resumes and viewing extracted information.
2. Backend: Implements the NER model using Python and provides APIs for data processing.
3. Database: MongoDB stores extracted information for efficient retrieval and management.
4. NER Model: A transformer-based model fine-tuned for resume data extraction.

#### **4.2 Data Flow Diagram**

The data flow includes the following steps:

1. Users upload resumes through the frontend interface.
2. The backend processes the resumes and sends them to the NER model.
3. The model extracts relevant entities and stores them in MongoDB.
4. Extracted data is displayed on the frontend for user review.

#### **4.3 Module Overview**

##### **4.3.1 Frontend (Django)**

* User-friendly interface for uploading resumes.
* Dashboard for viewing extracted data.

##### **4.3.2 Backend (Python and NER Model)**

* API endpoints for resume upload, data processing, and retrieval.
* Integration of the NER model for entity extraction.

##### **4.3.3 Database (MongoDB)**

* Stores extracted entities such as names, contact details, and qualifications.
* Enables efficient querying and retrieval of data.

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **5. Dataset**

#### **5.1 Dataset Description**

The dataset consists of resumes in various formats (PDF, DOCX, etc.) containing information such as:

* Personal details (name, contact information).
* Educational qualifications.
* Work experience.
* Skills and certifications.

#### **5.2 Data Preprocessing**

Preprocessing steps include:

* Converting resumes to text format.
* Removing irrelevant content (e.g., formatting artifacts).
* Normalizing text (e.g., lowercasing, removing special characters).

#### **5.3 Annotation and Labeling**

The dataset is annotated with entity labels such as:

* PERSON for names.
* ORG for organizations.
* DATE for dates.
* SKILL for skills and competencies.

#### 5.4 Challenges in Dataset Preparation

* Handling diverse resume formats.
* Ensuring consistent annotation quality.
* Balancing entity representation in the dataset.

### 

### 

### 

### **6. NER Model**

#### **6.1 Introduction to Transformer Models**

Transformer models like BERT and RoBERTa are state-of-the-art for NLP tasks due to their ability to understand contextual relationships in text using self-attention mechanisms.

#### **6.2 Model Selection**

For this project, we use a pre-trained transformer model (e.g., BERT) fine-tuned on the annotated resume dataset for NER.

#### **6.3 Training the Model**

##### **6.3.1 Data Splitting**

* Training Set: 70% of the dataset.
* Validation Set: 15% of the dataset.
* Test Set: 15% of the dataset.

##### **6.3.2 Fine-tuning Pre-trained Models**

* The pre-trained model is fine-tuned using the annotated dataset.
* Hyperparameters such as learning rate and batch size are optimized for best performance.

#### **6.4 Evaluation Metrics**

The model is evaluated using:

* Precision: Accuracy of predicted entities.
* Recall: Completeness of predicted entities.
* F1-Score: Harmonic mean of precision and recall.

#### **6.5 Model Performance**

* Achieves high precision and recall for key entities.
* Demonstrates robustness across diverse resume formats.

### **7. Implementation**

#### **7.1 Development Environment**

The development environment for the project includes:

* IDE: Visual Studio Code or PyCharm
* Python Environment: Anaconda or virtualenv for dependency management
* Version Control: Git and GitHub for source code management
* Operating System: Compatible with Windows, macOS, and Linux

#### **7.2 Step-by-Step Implementation**

1. Frontend Development:
   * Set up Django project and create templates for uploading resumes.
   * Design a dashboard to display extracted data.
2. Backend Development:
   * Implement API endpoints using Django REST Framework (DRF).
   * Integrate the NER model for processing uploaded resumes.
3. NER Model Integration:
   * Load the pre-trained transformer model.
   * Fine-tune the model using the annotated dataset.
   * Deploy the model as a service accessible via APIs.
4. Database Integration:
   * Configure MongoDB to store extracted entities.
   * Implement efficient querying mechanisms for data retrieval.
5. Testing and Debugging:
   * Test the system using various resume formats.
   * Debug and optimize the code for performance and accuracy.

### 

### 

### 

### 

### **8. Testing and Evaluation**

#### **8.1 Testing Methodology**

* Unit Testing: Test individual components such as the NER model, APIs, and database queries.
* Integration Testing: Ensure seamless interaction between frontend, backend, and database.
* System Testing: Validate the entire pipeline from resume upload to data extraction and display.
* User Acceptance Testing (UAT): Collect feedback from HR professionals to refine the system.

#### **8.2 Test Cases**

Sample test cases include:

* Uploading resumes in different formats (PDF, DOCX).
* Extracting entities such as name, contact details, and skills.
* Handling invalid or corrupted resume files.
* Querying and displaying stored data correctly.

#### **8.3 Evaluation Metrics**

* Accuracy: Measure the correctness of extracted entities.
* Precision and Recall: Evaluate the performance of the NER model.
* System Latency: Assess the time taken to process resumes.
* User Feedback: Gather qualitative feedback to identify areas for improvement.

### 

### 

### 

### 

### 

### **9. Deployment**

#### **9.1 Deployment Environment**

* Server: Cloud-based platforms such as AWS or Heroku
* Containerization: Use Docker to package the application for deployment
* Database Hosting: MongoDB Atlas for cloud-based database management

#### **9.2 Deployment Steps**

1. Containerization:
   * Create a Dockerfile for the application.
   * Build and test Docker containers locally.
2. Cloud Deployment:
   * Deploy the Docker container to a cloud platform.
   * Configure server settings and environment variables.
3. Database Configuration:
   * Set up MongoDB Atlas for remote database access.
   * Ensure secure connections using authentication and encryption.
4. Domain and SSL:
   * Configure a custom domain for the application.
   * Enable SSL certificates for secure communication.

#### **9.3 Post-Deployment Monitoring**

* Use monitoring tools such as Prometheus or AWS CloudWatch to track system performance.
* Regularly review logs for errors and optimize the system as needed.
* Gather user feedback to plan future updates and enhancements.

### 

### 

### 

### 

### **10. Challenges and Solutions**

#### **10.1 Challenges Faced**

1. Handling Diverse Resume Formats:
   * Resumes are submitted in various formats (PDF, DOCX, plain text), requiring robust preprocessing.
2. Entity Ambiguity:
   * Names, organizations, and other entities may overlap, leading to extraction errors.
3. Model Generalization:
   * Ensuring the model performs well on unseen resumes from different industries.
4. Data Annotation:
   * Time-consuming and labor-intensive process to label training data accurately.

#### **10.2 Solutions Implemented**

1. Preprocessing Pipelines:
   * Developed a pipeline to convert resumes into a consistent text format.
2. Advanced NER Models:
   * Used transformer-based models capable of understanding context and resolving ambiguities.
3. Cross-Validation:
   * Employed cross-validation techniques to improve model robustness.
4. Annotation Tools:
   * Leveraged annotation tools like Label Studio to streamline the labeling process.

### 

### 

### 

### 

### **11. Future Scope**

#### **11.1 Enhancements to the System**

1. Support for Additional Languages:
   * Extend the system to process resumes in languages other than English.
2. Integration with ATS:
   * Integrate the system with Applicant Tracking Systems (ATS) for end-to-end recruitment solutions.
3. Improved Visualization:
   * Add graphical dashboards for visualizing candidate data and analytics.

#### **11.2 Advanced NER Techniques**

1. Domain-Specific Models:
   * Develop models tailored to specific industries, such as healthcare or IT.
2. Active Learning:
   * Use active learning to iteratively improve the model with minimal human intervention.

#### **11.3 Scalability and Performance**

1. Distributed Computing:
   * Use distributed computing frameworks to handle large-scale resume processing.
2. Real-Time Processing:
   * Enable real-time resume evaluation for faster candidate shortlisting.

### 

### 

### 

### 

### **12. Conclusion**

The Named Entity Recognition (NER) system for resume evaluation demonstrates the potential of transformer-based models in automating a critical aspect of the recruitment process. By leveraging advanced NLP techniques, the system effectively extracts relevant information from resumes, reducing manual effort and improving efficiency. The project highlights the importance of preprocessing, model fine-tuning, and robust deployment practices. Future enhancements will further expand the system's capabilities, making it an indispensable tool for modern HR departments.