

**Columbian College of Arts & Sciences**

**MS in Data Science**

**Project Title:**

**RESUME MATCH**

**A Job Description and Resume Matching System**

**Instructor: Prof. Ning Rui**

**Team Members:**

**Pavani Chella – G41309434**

**Yasaswitha Gaddam – G31728688**

**Abstract**

This project presents **"Resume Match"**, an innovative system designed to transform the traditional recruitment process by automating the evaluation of resumes against job descriptions. Built on robust Natural Language Processing (NLP) and machine learning techniques, this tool provides a comprehensive analysis of candidate compatibility, offering metrics such as similarity scores, keyword matches, sentiment analysis, and experience calculations. By leveraging **TF-IDF Vectorization**, **Cosine Similarity**, and **TextBlob**, the system evaluates resumes for similarity to a job description and extracts valuable insights. The system is augmented with an intuitive user interface that enables recruiters to upload resumes, input job descriptions, and receive detailed visual insights, including bar charts, pie charts, and word clouds. By addressing inefficiencies in manual screening, the tool significantly reduces processing time, enhances accuracy, and minimizes biases. This report explores the conceptual framework, implementation details, results, and future potential of the **"Resume Match"** system, showcasing its role in modernizing and optimizing talent acquisition strategies.

## Introduction

Efficiently matching resumes with job descriptions is a significant challenge in recruitment. Manually evaluating resumes is time-intensive and prone to inconsistencies. This project aims to address these challenges by developing an NLP-based tool that automates the evaluation process.

The system processes resume in multiple formats (PDF, DOCX, TXT, CSV), calculates similarity scores using **TF-IDF** and **Cosine Similarity**, and provides additional insights like keyword matches, sentiment analysis, and experience extraction. The tool is designed to be robust, scalable, and user-friendly, enabling recruiters to prioritize resumes that best align with job descriptions. In today’s competitive job market, recruiters face significant challenges in efficiently matching candidates to job requirements. The traditional manual review process is labor-intensive, error-prone, and susceptible to bias. ‘Resume Match’ addresses these challenges by automating resume screening, leveraging NLP and AI techniques to provide actionable insights. This report outlines the development and functionality of the system.

**Table of Contents**

1. Abstract
2. Introduction
3. Problem Statement
4. Objectives
5. Methodology
6. System Architecture
7. Implementation Details

- Frontend

- Backend

- NLP Techniques

1. Results and Visualizations
2. Benefits
3. Limitations
4. Future Scope
5. Conclusion
6. References

**Problem Statement**

Recruiters spend approximately 23% of their time manually reviewing resumes, a process plagued by inefficiencies and human biases. The high volume of applications exacerbates these issues, often leading to overlooked talent. Key challenges include:

- Time-consuming resume screening.

- Errors and biases in manual review.

- Inefficient candidate-job matching.

**Objectives**

The primary goals of the "Resume Match" system are:

1. Efficiency: Accelerate the resume screening process.

2. Accuracy: Provide precise matches based on skills and experience.

3. Usability: Ensure an intuitive interface for recruiters and job seekers.

4. Insights: Offer detailed metrics like similarity scores, sentiment analysis, and keyword matches.

**Methodology**

The project utilizes a combination of NLP techniques, machine learning models, and visualization tools. The methodology involves:

1. Resume parsing and text preprocessing.

2. Similarity analysis using cosine similarity and TF-IDF vectorization.

3. Sentiment analysis to gauge tone and intent.

4. Experience extraction through date analysis.

5. Visualization of results to enhance interpretability.

**System Architecture**

The system comprises three main components:

1. Frontend: A Streamlit-based web application for user interaction.

2. Backend: Python-based processing using libraries like scikit-learn, TextBlob, and NLTK.

3. Visualization: Graphs, charts, and word clouds for result representation.

### **Tools and Libraries**

* **Python**: Programming language for development.
* **Streamlit**: Framework for building the user interface.
* **NLTK and TextBlob**: For NLP tasks like tokenization, stopword removal, lemmatization, and sentiment analysis.
* **PyPDF2 and python-docx**: For reading resumes in PDF and DOCX formats.
* **Matplotlib and WordCloud**: For generating visualizations.
* **scikit-learn**: For implementing TF-IDF vectorization and cosine similarity calculations.

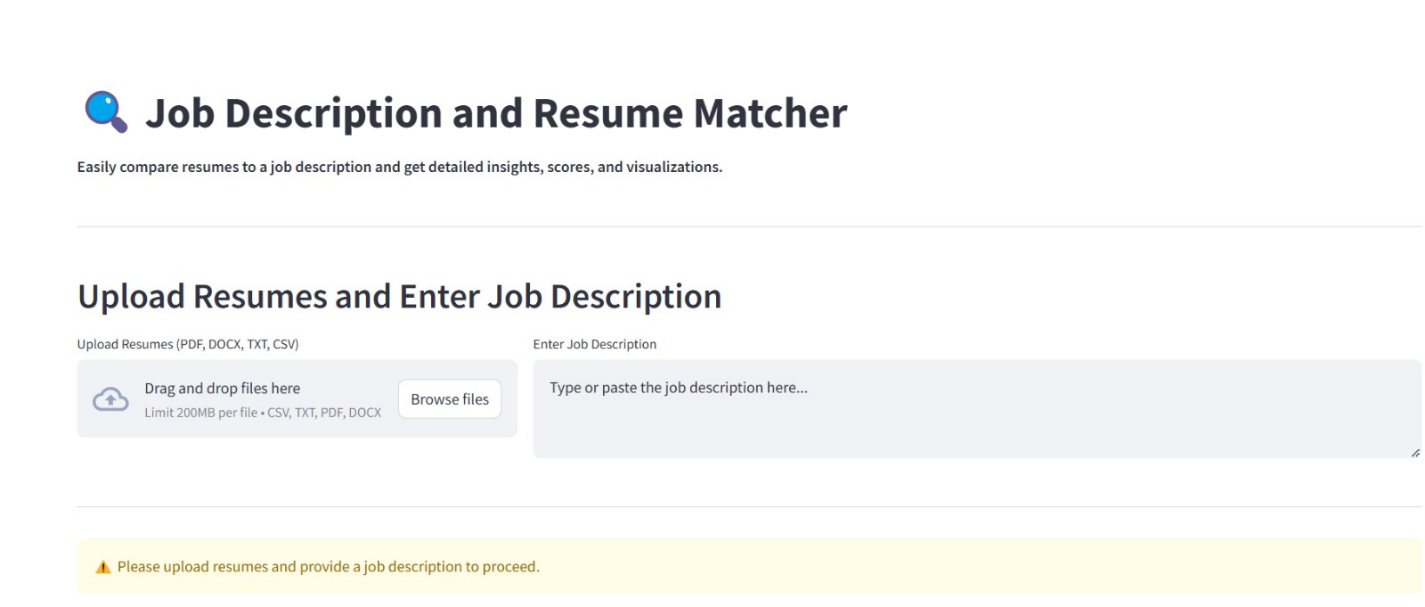
**Implementation Details**

**Frontend**

- Framework: Streamlit.

- Features: Resume upload (PDF, DOCX, TXT), job description input, and real-time results.

- Interactive Elements: Drag-and-drop functionality, expanders for detailed insights, and clickable processing buttons.



**Backend**

- Resume Parsing: Handles multiple formats (PDF, DOCX, TXT).

- Text Preprocessing: Tokenization, stopword removal, and text normalization.

- NLP Models: TF-IDF for vectorization and cosine similarity for scoring.

**NLP Techniques**

- Keyword Extraction: Identifies overlapping skills and terms between resumes and job descriptions.

- Sentiment Analysis: Uses TextBlob to detect the tone of resumes.

- Experience Calculation: Extracts and calculates work experience from chronological data in resumes.

### **Features**

1. **Resume Similarity Scoring**:
   * Compares resumes and job descriptions using **TF-IDF Vectorization** and **Cosine Similarity**.
2. **Keyword Matching**:
   * Identifies overlapping keywords between resumes and the job description.
3. **Sentiment Analysis**:
   * Assesses the tone of resumes (positive, neutral, or negative) using **TextBlob**.
4. **Experience Extraction**:
   * Extracts and calculates years of experience from resumes using regular expressions.
5. **Visualizations**:
   * **Bar Chart**: Displays similarity scores.
   * **Pie Chart**: Shows sentiment analysis results.
   * **Word Cloud**: Highlights matched keywords visually.

## Data

### **Input Formats**

The system supports resumes in:

* PDF
* DOCX
* TXT
* CSV

### **Job Description**

The job description is entered as free text by the user in the provided input field.

## Preprocessing

### **Steps**

1. **Text Cleaning**:
   * Converts text to lowercase.
   * Removes special characters, punctuation, and digits.

**Example:**

* + Input: "Experienced in Python (since 2015)!"
  + Output: "experienced in python since"

1. **Tokenization**:
   * Splits text into individual words.

**Example:**

* + Input: "experienced in python"
  + Output: ["experienced", "in", "python"]

1. **Stopword Removal**:
   * Removes commonly used words that don’t add value to the meaning (e.g., "the," "is").

**Example:**

* + Input: ["experienced", "in", "python"]
  + Output: ["experienced", "python"]

1. **Lemmatization**:
   * Reduces words to their base forms (e.g., "running" to "run").

**Example:**

* + Input: ["experienced", "python"]
  + Output: ["experience", "python"]

## Feature Extraction

1. **TF-IDF Vectorization**:
   * Converts text into numerical representations based on word importance.
   * Assigns higher weights to words that appear frequently in one document but rarely across others.

**Formula:**

where TF is Term Frequency and IDF is Inverse Document Frequency.

1. **Cosine Similarity**:
   * Measures the similarity between vectors to compute how closely a resume aligns with the job description.

**Formula:**

1. **Keyword Matching**:
   * Uses set operations to identify overlapping keywords between job descriptions and resumes.
2. **Sentiment Analysis**:
   * Determines the overall polarity (positive, neutral, negative) of resumes using TextBlob.
3. **Experience Extraction**:
   * Identifies years mentioned in resumes to calculate total experience using regular expressions.

**Results and Visualizations**

### **Outputs**

1. **Similarity Scores**:
   * Ranks resumes in descending order based on their similarity to the job description.

**Example:**

|  |  |
| --- | --- |
| **Resume File** | **Similarity Score** |
| Data\_Scientist.pdf | 0.85 |
| Software\_Engineer.docx | 0.72 |
| Project\_Manager.pdf | 0.60 |

1. **Keyword Matches**:
   * Lists matched keywords for each resume.

**Example:**

|  |  |
| --- | --- |
| Resume File | Matched Keywords |
| Data\_Scientist.pdf | Python, Machine Learning, Data |
| Software\_Engineer.docx | Python, Software Development |

1. **Sentiment Analysis**:
   * Displays sentiment scores, indicating whether the tone is positive, neutral, or negative.

**Example:**

|  |  |
| --- | --- |
| Resume File | Sentiment Score |
| Data\_Scientist.pdf | Positive (0.32) |
| Software\_Engineer.docx | Neutral (0.10) |

1. **Experience Details**:
   * Provides the total years of experience extracted from each resume.

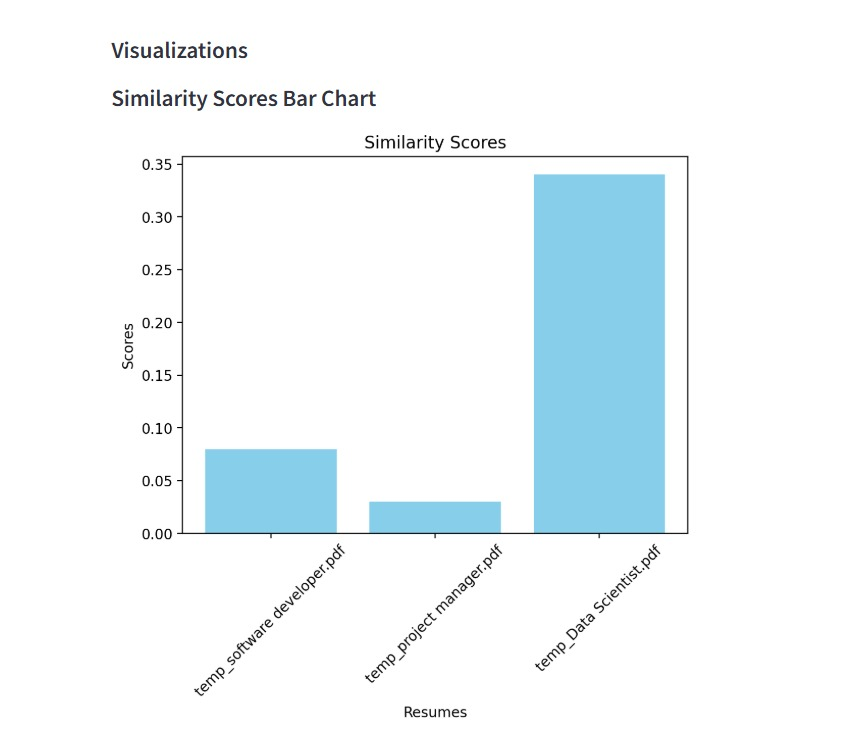
**Example:**

|  |  |
| --- | --- |
| Resume File | Experience |
| Data\_Scientist.pdf | 5 years |
| Software\_Engineer.docx | 3 years |

### **Visualizations**

#### **Bar Chart**

* **Description:** Highlights similarity scores for easy comparison.
* **Purpose:** Identify the best-matching resumes quickly.

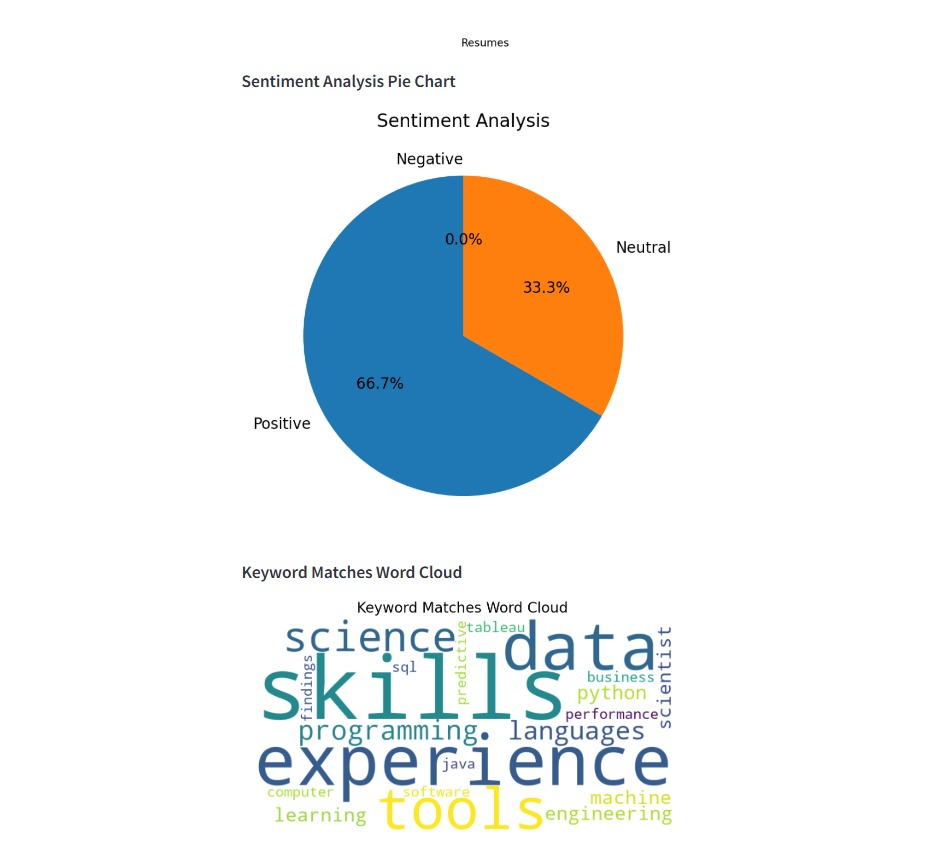


#### **Pie Chart**

* **Description:** Summarizes sentiment distribution across resumes.
* **Purpose:** Understand the tone of resumes at a glance.

#### **Word Cloud**

* **Description:** Emphasizes frequently matched keywords visually.
* **Purpose:** Highlight relevant terms from resumes.



**Benefits**

- Efficiency: Processes resumes in bulk, saving significant time.

- Accuracy: Minimizes errors through automated analysis.

- Objectivity: Reduces bias in candidate evaluation.

- Enhanced User Experience: Provides clear, visual results.

**Limitations**

1. Dependence on Job Description Quality: Results may vary based on input accuracy.

2. Processing Speed: Slower for large datasets.

3. Text-only Resumes: Limited ability to process multimedia content.

**Future Scope**

1. Multilingual Support: Enable analysis in multiple languages.

2. Custom Weighting: Allow recruiters to prioritize certain skills.

3. Enhanced Visualizations: Add dynamic and interactive graphical elements.

4. Improved Scalability: Optimize performance for larger datasets.

5. Integration of **transformer models** like BERT for deeper semantic analysis.

6. Support for additional file formats like images and HTML.

7. Real-time interactive visualizations for enhanced user experience.

**Conclusion**

"Resume Match" effectively demonstrates the potential of NLP and AI to revolutionize recruitment. By automating and enhancing resume screening, it empowers recruiters and job seekers alike, fostering better alignment and reducing inefficiencies. Future developments will focus on expanding capabilities, ensuring inclusivity, and enhancing user experience.

**References**

1. scikit-learn Documentation: <https://scikit-learn.org/>
2. NLTK Documentation: <https://www.nltk.org/>
3. TextBlob Documentation: <https://textblob.readthedocs.io/>

**Contribution**

### **Contribution Report : Pavani Chella**

**Contributions**:

#### **Project Design and Architecture**

* Led the design of the project’s backend workflow, focusing on modularity and efficiency.
* Defined the core features, including **TF-IDF Vectorization**, **Cosine Similarity**, and **Sentiment Analysis**, as part of the NLP pipeline.
* Collaborated with the second team member to outline the integration of backend features with the frontend for seamless functionality.

#### **Backend Development**

* **Preprocessing Pipelines**:
  + Implemented preprocessing steps such as text cleaning, tokenization, lemmatization, and stopword removal using **NLTK**.
  + Ensured all resume data was normalized for consistent vectorization.
* **Similarity Scoring**:
  + Developed the process\_resumes function to calculate similarity scores between resumes and job descriptions using **TF-IDF** and **Cosine Similarity**.
  + Optimized vectorization for handling large datasets efficiently.

#### **NLP Features**

* **Keyword Matching**:
  + Built a system to extract and compare keywords between resumes and job descriptions.
  + Utilized set operations for fast keyword intersection and identification of critical overlaps.
* **Experience Extraction**:
  + Created robust regex-based logic to extract years of experience from resumes, addressing variations in date formats.

#### **Validation and Testing**

* Conducted extensive testing on backend functionalities to ensure accurate similarity computation and feature extraction.
* Debugged preprocessing issues caused by inconsistent resume structures (e.g., PDFs with non-standard encoding).

#### **Documentation and Reporting**

* Authored technical documentation for backend modules, detailing algorithms and workflows.
* Wrote detailed sections of the project report on **NLP techniques**, backend architecture, and preprocessing examples.

### **Contribution : Yasaswitha Gaddam**

**Contributions**:

#### **Collaborative NLP Development**

* Partnered with Pavani Chella to refine NLP workflows and implement efficient algorithms for **TF-IDF Vectorization** and **Cosine Similarity**.
* Enhanced the preprocessing pipeline by experimenting with additional cleaning techniques for noisy resume data.

#### **Frontend Development**

* **Streamlit Interface**:
  + Designed and implemented a user-friendly frontend using **Streamlit**.
  + Enabled users to upload multiple resumes and view similarity scores, keyword matches, and sentiment breakdowns interactively.
* **Error Handling**:
  + Added intuitive error messages and warnings for unsupported file formats and missing inputs.

#### **Visualization Development**

* **Bar Charts**:
  + Developed dynamic bar charts to compare similarity scores across resumes, enabling recruiters to easily identify top candidates.
* **Pie Charts**:
  + Created pie charts to visualize sentiment distribution (positive, neutral, negative) across resumes.
* **Word Clouds**:
  + Generated word clouds to highlight frequently matched keywords, improving readability and insight.
* Scaled all visualizations to adapt dynamically to varying dataset sizes for clarity and professional presentation.

#### **Feature Integration**

* Connected frontend components to backend workflows, ensuring the smooth display of similarity scores, matched keywords, sentiment analysis, and experience details.
* Ensured real-time updates for visualizations based on user inputs.

#### **Testing and Optimization**

* Conducted end-to-end testing of the system with a variety of resumes and job descriptions to ensure robustness and accuracy.
* Improved processing speed by optimizing file handling and visualization generation.

#### **Presentation and Reporting**

* Designed and delivered a comprehensive presentation, demonstrating the system’s NLP capabilities and user interface.
* Authored sections of the project report on **visualizations**, **frontend architecture**, and **system usability**.

### **Collaborative Contributions**

Both team members collaborated closely on:

* Brainstorming and defining the project scope.
* Fine-tuning the NLP workflows, particularly TF-IDF and cosine similarity computations.
* Testing the system with diverse datasets to ensure accuracy and scalability.
* Co-authoring the final presentation and report.