Chandigarh Engineering College Jhanjeri
Mohali-140307
Department of Artificial Intelligence (AI) and Data Sciences

Smart Resume Analyzer

Project-I

BACHELOR OF TECHNOLOGY
(Artificial Intelligence and Data Science)

SUBMITTED BY:

Ankit Singh Rawat,Ankush Kumar,
Anup Kumar Singh, Labikesh

Roll No :-
2330691,2330692,
2330693,2330740

Jan 2025

Under the Guidance of
Dr. Shanky Goyel
(Asst. Professor)

Department of Artificial Intelligence and Data Science
Chandigarh Engineering College Jhanjeri Mohali - 1040307

Table of Contents

S.No.	Contents	Page No
1.	Introduction	01 - 02
2.	Brief Literature survey	03 - 06
3.	Problem formulation	07
4.	Objectives	08
5.	Methodology/ Planning of work	09 - 10
6.	Facilities required for proposed work	11
7.	References	12

Introduction

• Recruitment plays a vital role in shaping an organization's growth by hiring suitable candidates.
In today's competitive job market, a single job posting can attract hundreds or even thousands of applications.
Manually screening such a large volume of resumes is:
- Time-consuming and tedious
- Prone to human error and inconsistency
- Susceptible to unconscious bias

• Traditional Applicant Tracking Systems (ATS) often rely on keyword matching, which fails to capture the contextual meaning of skills, experience, and education.
For example, a candidate proficient in "Data Analysis" might be overlooked if the job description mentions "Data Analytic."

• To overcome these limitations, this project proposes a Resume Analyzer using Machine Learning (ML) and Natural Language Processing (NLP).
The system will:
- Extract structured information (skills, education, experience, projects) from unstructured resumes.
- Compare candidate profiles with job descriptions using ML models.
- Rank candidates based on their suitability score.
- Provide recruiters with a faster, fairer, and smarter decision-making tool.

Chandigarh Engineering College Jhanjeri
Mohali-140307
Department of Artificial Intelligence (AI) and Data Sciences

Brief Literature Survey

2.1 Traditional Resume Screening Methods

• Keyword-based Screening:

> Works by matching specific keywords in resumes with those in the job description.

> Example: If the job requires "Python," resumes with the word "Python" are shortlisted.

>Limitations:

- Ignores context (e.g., "Python basics" vs. "Python expert").

- Fails when synonyms are used (e.g., "Data Analysis" vs. "Data Analytics").

- High false positives and negatives.

• Rule-based Systems:

> Recruiters or system administrators define strict rules such as:

- "If candidate has B.Tech in Computer Science AND 2+ years of Python → shortlist."

• Limitations:

>Hard to scale and maintain.

2.2 Machine Learning based Resume Screening

• Supervised Learning Approaches:

> Algorithms like Logistic Regression, Random Forest, Support Vector Machine (SVM) used for resume classification.

> Trained on labeled datasets (suitable vs unsuitable resumes).

> Can handle larger feature sets compared to keyword systems.

> Limitations: Require well-labeled training data.

• Unsupervised Learning Approaches:

> Clustering techniques (K-means, Hierarchical Clustering) group resumes by similarity.

> Useful when labeled data is not available.

> Limitations: Less accurate for ranking candidates individually.

2.3 Natural Language Processing for Resume Parsing

• Text Preprocessing:

> Tokenization, stop-word removal, stemming/lemmatization.

> Standardizes resume data into structured format.

• Named Entity Recognition (NER):

> Extracts entities like skills, degrees, organizations, job titles, years of experience.

• Vectorization Methods:

> Bag of Words (BoW): Simple representation based on word frequency.

> TF-IDF (Term Frequency–Inverse Document Frequency): Measures importance of words.

> Word Embeddings (Word2Vec, GloVe): Capture contextual similarity.

2.4 Deep Learning and Transformer based Models

• Recurrent Neural Networks (RNNs) and LSTMs:

> Capture sequential patterns in resumes.

> Useful for understanding sentence-level context.

• Transformer Models (BERT, RoBERTa, XLNet):

> Pre-trained on massive text corpora.

> Provide state-of-the-art results in semantic similarity.

> Example: Matching “developed ML models” in resume with “machine learning expertise required” in job description.

• Advantages over ML:

> Understand semantics and context, not just keywords.

> More robust against different wording styles in resumes.

2.5 Existing Tools and Systems

• LinkedIn Talent Insights:

> Provides insights and filters for candidate search.

> Commercial, not customizable for research.

• Jobscan:

> Matches resume keywords with job descriptions.

> Focused on keyword optimization, lacks semantic understanding.

• HireVue & AI Recruitment Tools:

> Use ML/AI for shortlisting candidates and even video interview analysis.

> Often criticized for lack of transparency and fairness

2.6 Summery of Literature Findings

• Keyword and rule-based systems are outdated and insufficient.

• ML algorithms improve automation but need structured datasets.

• NLP helps extract structured information from unstructured resumes.

• Deep Learning and Transformers (BERT) outperform older approaches by capturing semantic meaning.

• Existing commercial tools are powerful but closed-source, costly, and not customizable for academics.

• Therefore, there is a research gap for a customizable, open-source Resume Analyzer using ML + NLP for fair, efficient candidate ranking.

2.7 Diagrams

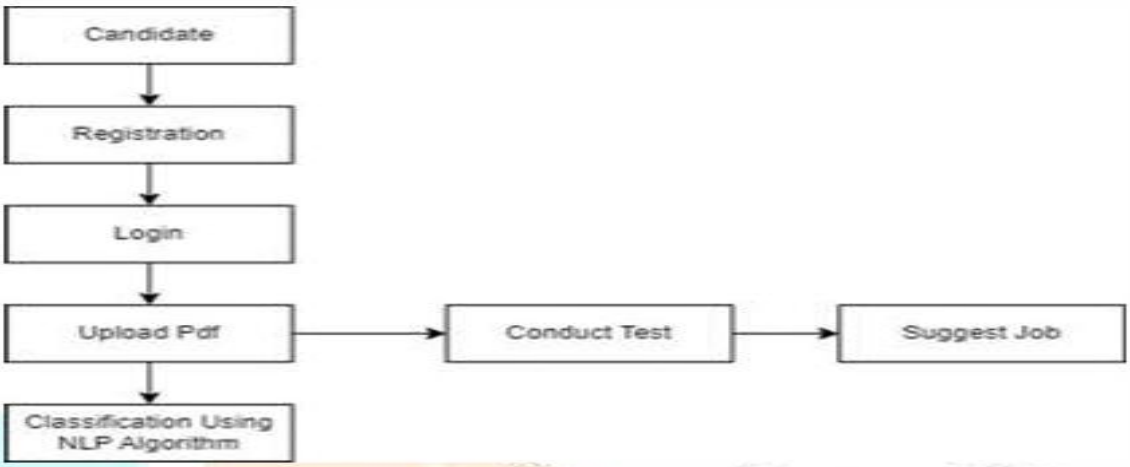

Figure : System Architecture

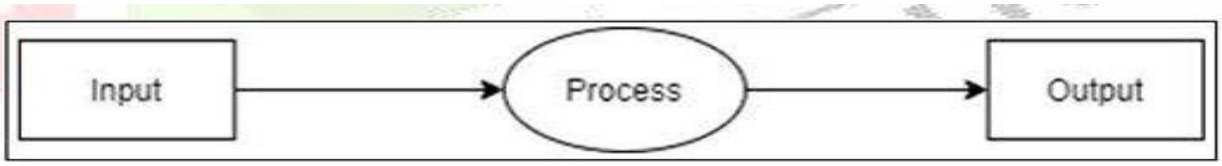

Figure : Data Flow diagram

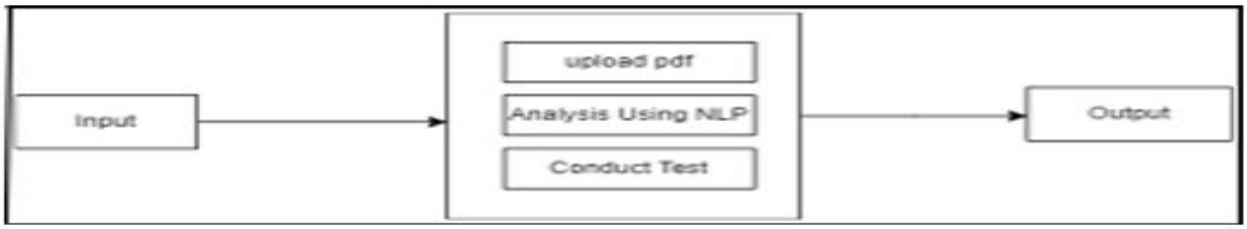

Figure : Data Flow diagram

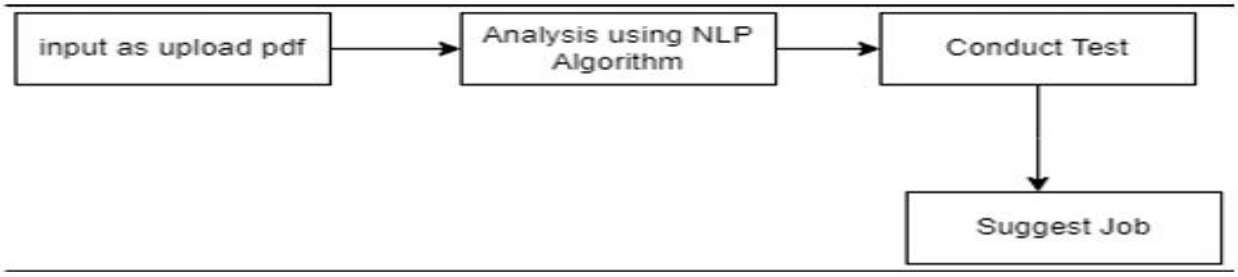

Figure : Data Flow diagram

Problem Formulation

3.1 Info

The recruitment process has become increasingly complex due to the rising number of applicants for each job posting. Manual resume screening is time-consuming, inconsistent, and prone to human bias. Existing Applicant Tracking Systems (ATS) mainly rely on keyword matching, which often fails to capture the context or semantic meaning of resumes.

The main research problem is:

"How to develop an intelligent Resume Analyzer using Machine Learning and NLP that can extract structured information from resumes, compare them with job requirements contextually, and rank candidates fairly and efficiently?"

3.2 Challenges Identified

- Resumes exist in multiple formats (PDF, DOC, TXT) with unstructured content.
- Difficulty in extracting and standardizing features such as skills, education, and experience.
- Ensuring contextual understanding (e.g., Data Analysis ≈ Data Analytics).
- Reducing human bias while maintaining accuracy.
- Handling large volumes of resumes in limited time.

3.3 Significant of Proposed Work

- Automate resume screening, saving recruiter time and effort.
- Improve accuracy and fairness in candidate selection.
- Provide a scalable solution that can process thousands of resumes efficiently
- Deliver ranked candidate lists, enabling better dec-making in hiring.

Objectives

The primary aim of this project is to design and implement an intelligent Resume Analyzer using Machine Learning and NLP that automates the process of candidate evaluation and ranking.

=> Specific Objectives:

• Automated Resume Screening

Develop a system that can automatically process resumes of different formats (PDF, DOC, TXT) and extract meaningful information.

• Feature Extraction using NLP

Identify and structure key features such as skills, education, work experience, and projects from unstructured resume text.

• Resume–Job Description Matching

Implement ML/NLP models to measure the similarity between resumes and job requirements at both keyword and semantic levels.

• Candidate Ranking Mechanism

Design a scoring system that ranks candidates based on their suitability (skills match, experience level, education relevance).

• Bias Reduction and Fairness

Minimize human subjectivity by using standardized, data-driven evaluation techniques.

• Deployment for Practical Use

Provide a web-based interface where recruiters can upload resumes and job descriptions, and receive ranked results.

Methodology / Planning of Work

5.1 Data Collection

• Collect a diverse dataset of resumes in multiple formats (PDF, DOC, TXT) from Kaggle and other open sources.
• Gather job descriptions from online job portals such as LinkedIn, Naukri, and Indeed.

5.2 Preprocessing of Resumes

• Convert all resumes into plain text.
• Perform standard NLP preprocessing:
> Tokenization, stop-word removal, lemmatization.
> Named Entity Recognition (NER) for extracting skills, degrees, organizations, and job titles.

5.3 Feature Extraction

• Apply different techniques to convert resume and job description text into numerical features:
> TF-IDF (Term Frequency–Inverse Document Frequency) for keyword importance.
> Word Embeddings (Word2Vec, GloVe) for contextual similarity.
> Transformer-based Embeddings (BERT) for semantic understanding.

5.4 Model Development & Evaluation

• Train and test multiple ML models: Logistic Regression, Random Forest, Support Vector Machine (SVM).
• Experiment with Deep Learning models (BERT) for improved accuracy.
• Evaluate performance using Precision, Recall, Accuracy, and F1-score.

5.5 Resume Ranking

• Implement a weighted scoring system:

> Skills – 40%

> Experience – 30%

> Education – 20%

> Projects – 10%

• Generate a final ranking list of candidates based on suitability scores.

5.6 Deployment

• Develop a web-based application (Flask/Django).

• Functionality: Recruiter uploads job description + resumes → System outputs ranked candidates with suitability scores.

Facilities Required For Proposed Work

6.1 Hardware Requirements

- Processor: Intel i5 or higher (i7/i9 recommended for faster processing).
- RAM: Minimum 8 GB (16 GB preferred for deep learning models).
- Storage: At least 500 GB HDD/SSD to store datasets and processed resumes.
- GPU (optional): NVIDIA GPU (e.g., GTX/RTX series) for accelerating deep learning tasks.

6.2 Software Requirements

- Programming Language: Python (version 3.9 or above)
- Libraries & Frameworks:
  > NLP: NLTK, SpaCy
  > ML: Scikit-learn, Pandas, NumPy
  > Deep Learning: TensorFlow, PyTorch, HuggingFace Transformers
- Development Tools: Jupyter Notebook, PyCharm/VS Code
- Deployment Frameworks: Flask / Django for building the web application.
- Version Control: GitHub or Git for collaboration and backup.

6.3 Dataset Requirements

- Resume Dataset: Publicly available datasets (e.g., Kaggle Resume Dataset..
- Job Descriptions: Collected from LinkedIn, Naukri, Indeed, and other job portals.
- Custom Dataset: Additional resumes and job postings manually gathered for testing.

REFERENCES

1. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT, pp. 4171–4186.

2. Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing. 3rd Edition, Pearson.

3. Brownlee, J. (2016). Machine Learning Mastery with Python: Understand Your Data, Create Accurate Models, and Work Projects End-to-End. Machine Learning Mastery.

4. Kaggle Resume Dataset. (2021). Available at: https://www.kaggle.com

5. Jobscan. (2023). Resume Optimization and Screening Tool. Available at: https://www.jobscan.co

6. LinkedIn Talent Insights. (2023). Talent Analytics for Recruiters. Available at: https://business.linkedin.com/talent-solutions

7. Choudhary, S., & Singh, A. (2021). Automated Resume Screening using Machine Learning and NLP. International Journal of Computer Applications, Vol. 183(29), pp. 12–18