AI ENHANCED SKILL MATCHER

Prof. Dikshendra Sarpate*, Prarthana Kolhe*, Srushti Kalbhor*, Sanchi Yehalegaonkar*

*1,2,3,4 Artificial Intelligent & Data Science, Zeal College Of Engineering and Research, India.

DOI: https://www.doi.org/10.56726/IRJMETS53582

ABSTRACT

In today's competitive job market, matching candidates to job descriptions efficiently and accurately is crucial for recruiters and HR professionals. This research introduces a web application, termed the Resume Match Predictor, designed to predict the percentage match between a candidate's resume and a given job description. Leveraging natural language processing (NLP) techniques and machine learning algorithms, the application aims to streamline the candidate screening process and enhance decision-making. This paper begins by outlining the challenges faced by recruiters in manually assessing resumes against job descriptions, including time constraints, subjectivity, and inconsistency. It then presents the architecture and key functionalities of the Resume Match Predictor, detailing the process of text preprocessing, feature extraction, and model training for predicting the match percentage. Furthermore, the paper discusses the NLP techniques employed in the application, such as word2Vec, and text classification algorithms. Special attention is given to the integration of pre-trained language models, Ethical considerations surrounding algorithmic decision-making in recruitment are addressed, including bias mitigation, fairness, and transparency. Resume Match Predictor across various industries and job roles. Results demonstrate its ability to significantly reduce the time and effort required for candidate screening while maintaining accuracy and fairness.

Conclusion, the Resume Match Predictor emerges as a valuable tool for modernizing the recruitment process, offering tangible benefits in terms of efficiency, objectivity, and candidate quality.

Keywords: Skill Matching, Word2vec, NLP, Streamlit, Skill Extraction, Tokenization.

I. INTRODUCTION

Recruitment in the current IT scenario has taken up the pace, as market is flooded with various cross cutting technologies. Software organizations are on the chase to enroll raw talents directly from the colleges through employment fairs. Usually the Human Resource (HR) department of the company works on the recruitment and hires the employees or candidates on the basis of their skills and educational qualifications. Manually, to recruit the candidates with the matching job profiles for the HR department and for the candidates to search the job according to their interest or skills is a very tough job. Analysis of Resume is one of the most costly and time taking task during the time spent choosing another expert. [2] The number of results of job searching is huge but not well-ranked, so the job seeker has to review every job description. Since no one has enough time to read all the jobs in the search result, the actual quality of the job-seeking service is low. This is a classic problem of information overflow. The reason for such a result is that current job searching websites use the same information retrieval technology like “Inverted index” as the common search engines, which just use keywords to map all the stored documents. Modern search engines all have some ranking algorithms to sort the search results, like page rank, so the top results always be the most related ones. However such algorithms are unavailable to the job search systems because the criteria of how to rank the job searching result is very personalized.[7] Formal job search and application typically involves matching one’s profile or curriculum vitae (CV) with the available job descriptions (JD), and then applying for those job opportunities whose JDs are the closest match to one’s CV, and also considering his/her needs, constraints, and aspirations. A new methodology that combines several natural language processing (NLP) techniques for robust skill extraction from naturally occurring texts in CVs and JDs is proposed. An approach for inferring implicit skills in a JD is introduced, and a method to extract these implicit skills from other similar JDs is presented. A bi-directional matching algorithm to match skills between CVs and JDs is suggested to obtain the most relevant job recommendation for each CV. Literature Survey. For Skill matching- Analyze job descriptions to identify required skills and qualifications for specific roles. Create a profile of desired skills and prioritize them based on their relevance and importance to the job. Semantic Similarity: Calculate semantic similarity scores between resume skills and job requirements.
using techniques like Word Embeddings (e.g., Word2Vec, GloVe) or pre-trained language models (e.g., BERT). Measure the similarity between skill vectors in the resume and job description embeddings to assess skill relevance.\[9\] In this project, we'll leverage the power of Python, NLP techniques, and the Streamlit library to build a web-based resume analyzer. This tool will allow users to upload their resumes, and it will provide insights and feedback on various aspects, such as: Key Skills Identification: Extracting and highlighting key skills mentioned in the resume, helping candidates understand if they align with the job requirements. Experience and Education Parsing: Parsing and summarizing the candidate's work experience and educational background, providing a concise overview for recruiters. Formatting and Readability Evaluation: Analyzing the overall formatting, readability, and structure of the resume to ensure it's well-presented and easy to read. Custom Feedback and Recommendations: Providing personalized feedback and recommendations based on the content of the resume, such as suggesting relevant courses or certifications to enhance skills. Technologies and Tools: Python: We'll use Python for the backend processing, including NLP tasks such as text extraction, parsing, and analysis. Streamlit is a popular Python library for building interactive web applications with minimal code. We'll use Streamlit to create the user interface (UI) for our resume analyzer, allowing users to upload their resumes and view the analysis results in real-time. Natural Language Processing (NLP) Libraries: We'll leverage NLP libraries such as spaCy or NLTK for text processing tasks like tokenization, entity recognition, and keyword extraction. Machine Learning Models (Optional): Depending on the complexity of the analysis, we may incorporate machine learning models to improve the accuracy of skills identification, sentiment analysis, or other tasks. The objective of the Resume Matching Predictor App is to provide job seekers with a fast and accurate evaluation of how well their resumes match a specific job description. By leveraging natural language processing (NLP) techniques, the app analyzes the textual content of both the resume and the job description to identify similarities and key points of alignment. The app then generates a percentage score indicating the degree of match, helping users understand their suitability for the position and guiding them in tailoring their resumes effectively for increased job application success.

II. LITERATURE SURVEY

1) SHIQIANG GUO proposed In this we have made an effort to propose a personalized job-resume matching system, which could help job seekers to find appropriate jobs more easily. We devised a new statistical-based ontology similarity measure to compare the resume models and the job models. Job searching, which has been the focus of some commercial job-finding websites and research papers is not a new topic in information retrieval. Usually, scholars call them Job Recommender Systems (JRS), because most of them use technologies from recommender systems. Classified Recommender Systems into four categories: Collaborative Filtering, Content-based filtering, Knowledge-based and Hybrid approaches. In the system, job descriptions and resumes are parsed into job models and resume models by the information extraction module. When searching the jobs by a resume, similarity values between the resume model and job description models are calculated in the ontology matching module. The result is sorted by the ontology similarity scores, which are the sum of similarities of different fields multiplied by their weights. Future work on this system would place greater consideration on job seekers’ personal preferences like job location, career development plan, and company background.

2) Thimma Reddy Kalva proposed Skill Finder is a tool which ranks the student skills from the resumes to the job requirements from Employers, Department and Faculty looking for student interns, full-time employees and also Research, Graduate and Teaching Assistants. The Skill Finder tool works efficiently in matching the student resumes to the jobs posted. The process of used is called Named Entity Recognition (NER) . One approach for finding named entities is to use a combination of lists and regular expressions. It allows to maintain a complete history of job requirements from external employers and department. It hosts student personal information, academic history and student resumes. It successfully sends email alerts to students on a job posting. This system is scalable and flexible to extend further for adding.

3) Niti Khamker1, Yuti Khamker1, Mani Butwall2 proposed Human Resource Management is seemingly supported by and given additional opportunities by the event of Job Characteristics Model (JCM) that successively is predicated on the conception of contemporary job style. luckily, the event in the fashionable data system, digital technologies, the universal access of electronic technology and, the internet led to the inclination of the worldwide Human Resource Management development and create the system additional
applicable. Following the trend, the planned system tries to style a system for e-recruitment which can
sanctuary new model of economical operation on Human Resource Management within the web Age. This
projected system, enforces a company-adjusted accomplishment system that will assist the human resource
department in briefly listing the proper candidate for a particular job profile.

4) Sakshi Takkar proposed IT industries are currently facing a major challenge of mapping candidate’s skill.
So Resume Analysis with automated systems is a very potential area of research. Automated Resume Analysis
Systems helps the companies to find the candidates for the job post on the basis of his/her skill or
talent. These systems are made with the help of machine learning techniques or algorithms. Different
algorithms have different features which are compared on the basis of various metrics like Root Mean
Square Error (RMSE), accuracy, recall, relative absolute error and precision. In the future scope, one can aim
for the creation of an end to end continuous expert system to handle the complete recruitment process of an
organization.

5) Akshay Gugnani, Hemant Misra Proposed System: A job recommender system to match resumes to job
descriptions (JDs), which are unstructured text. Uses natural language processing (NLP) techniques for skill
extraction from resumes and JDs.

<table>
<thead>
<tr>
<th>Sr no</th>
<th>Title</th>
<th>Author</th>
<th>Year</th>
<th>Methodology</th>
<th>Limitation/ cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Job Role Description and Skill Matching in a Rapidly Changing Labor</td>
<td>Author: George Stalidis IAEME</td>
<td>2023</td>
<td>In the contemporary dynamic, globalized, and digitalized labor market,</td>
<td>The implemented methods for analyzing job postings showed limitations and called for tuning and some manual intervention</td>
</tr>
<tr>
<td></td>
<td>Market Using Knowledge Engineering</td>
<td>ISSN: 0000-0003</td>
<td></td>
<td>important challenges are related to skill-matching</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Resume Match System</td>
<td>Author: Niti Khamker IAEME</td>
<td>2023</td>
<td>Text Analysis could be a crucial field for machine learning algorithms.</td>
<td>This projected system enforces a company-adjusted accomplishment system that will assist the human resource department in briefly listing the proper candidate for a particular job profile.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ISSN: 0232-9877</td>
<td></td>
<td>However, the information (raw data), a series of symbols cannot be</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>gobbled</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Comparative Analysis of Tools for Matching Work-Related Skill Profiles</td>
<td>Author: Florian Büttiker IAEME</td>
<td>2021</td>
<td>When thinking about the possibilities of CV matching improvement one can think about infinite solutions.</td>
<td>Especially for the job description of Google the available CVs obviously included suitable candidates</td>
</tr>
<tr>
<td>Page</td>
<td>Title</td>
<td>Author</td>
<td>Year</td>
<td>Abstract</td>
<td>Keywords</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>---------------------------</td>
<td>------</td>
<td>--------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4</td>
<td>Implicit Skills Extraction Using Document Embedding and Its Use in Job Recommendation</td>
<td>Akshay Gugnani</td>
<td>2020</td>
<td>The premise of our proposed approach is to identify implicit or latent skills and use them to improve job recommendation for candidates.</td>
<td>The paper proposed a generalizable ensemble method for skill extraction from unstructured text of resumes as well as JDs</td>
</tr>
<tr>
<td>5</td>
<td>A Deep Insight of Automatic Resume Classifiers For Skill Mapping By Recruiters</td>
<td>Sakshi Takkar</td>
<td>2019</td>
<td>Mainly focus on analyzing CVs of researchers of different disciplines</td>
<td>Resume Analysis with automated systems is a very potential area of research</td>
</tr>
<tr>
<td>6</td>
<td>Jobseeker-industry matching system using automated keyword selection and visualization approach</td>
<td>Norhaslinda Kamaruddin</td>
<td>2019</td>
<td>These websites provide a platform for the jobseeker to find a suitable job and potential employers to advertise job or position that they need to hire.</td>
<td>The extraction of relevant information for job selection is not a trivial task</td>
</tr>
<tr>
<td>7</td>
<td>How to Match Jobs and Candidates - A Recruitment Support System Based on Feature Engineering and Advanced Analytics</td>
<td>Andrzej Janusz</td>
<td>2018</td>
<td>The content-based methods predict preferences regarding a set of items based on their similarity to items that a user has preferred in the past</td>
<td>In the paper, we described a recruitment support system aiming to help recruiters in finding candidates who are likely to be interested in a given job</td>
</tr>
<tr>
<td>8</td>
<td>A Job Post and Resume Classification System for Online Recruitment</td>
<td>Abeer Zaroor</td>
<td>2017</td>
<td>First, the job post is pre-processed and filtered through removing noisy information such as city names, and state and country acronyms that appear in the job title or job details.</td>
<td>In this paper, we have proposed a job post and resume classification system (JRC) based on coupling an integrated skills knowledge base and an automatic matching procedure between</td>
</tr>
<tr>
<td>9</td>
<td>RESEARCHER: A PERSONALIZED RESUME-JOB MATCHING SYSTEM</td>
<td>Shiqiang Guo</td>
<td>2015</td>
<td>The basic problem in this thesis is how to find appropriate job descriptions by user's resume, which means we need calculate the similarity between the user's resume and the job.</td>
<td>We have introduced a novel method for the accurate and automatic matching of job offers</td>
</tr>
</tbody>
</table>
III. METHODOLOGY

Data Requirements:
Identify the specific information you want to extract from resumes, such as personal details, work experience, education, skills, certifications, etc. Determine the structure of your data annotation schema, including the labels for each section and the format for representing extracted information. Collect a diverse set of resumes from various sources, including online job portals, career websites, professional networking sites (e.g., LinkedIn), academic databases, and personal contacts. Ensure the resumes cover a wide range of industries, job roles, experience levels, and formats (e.g., chronological, functional, combination).

Data Annotation:
Develop guidelines and standards for annotating resumes, including rules for labeling each section and extracting relevant information. Establish a team of annotators trained to follow the annotation guidelines consistently. Annotate each resume to label sections (e.g., personal information, work experience) and extract relevant data using manual or automated methods.

Data Cleaning and Preprocessing:
Remove any sensitive or personally identifiable information (PII) from the resumes to ensure compliance with data protection regulations. Clean the text data by removing formatting inconsistencies, special characters, and irrelevant information. Preprocess the text data by tokenizing, stemming, lemmatizing, removing stop words, and performing other text normalization techniques.

Augmentation:
Generate synthetic resumes by introducing variations in the existing dataset, such as modifying job titles, adding or removing experiences, or changing educational backgrounds. Use data augmentation techniques such as text paraphrasing, synonym replacement, or random insertion/deletion of words to increase the diversity of the dataset.

Data Splitting:
Divide the annotated dataset into training, validation, and test sets using a stratified sampling approach to ensure balanced representation across different categories (e.g., industries, experience levels). Allocate a larger portion of the data to the training set (e.g., 70-80%) and smaller portions to the validation and test sets (e.g., 10-15% each).

Model Training and Evaluation:
Choose an appropriate machine learning or deep learning model architecture (e.g., natural language processing (NLP) models like BERT, LSTM, or SVM) for the resume analysis task. Train the model using the annotated training data and fine-tune hyperparameters using the validation set to optimize performance. Evaluate the trained model’s performance on the test set using metrics such as accuracy, precision, recall, F1-score, and confusion matrix.
Iterative Improvement: Analyze the model’s performance and identify areas for improvement based on evaluation results and user feedback. Iterate on the model architecture, feature extraction techniques, data preprocessing methods, and annotation guidelines to enhance performance and address any limitations.

Deployment and Maintenance: Deploy the trained model as part of your smart resume analyzer application or service, ensuring scalability, reliability, and usability. Monitor the performance of the deployed model in real-world scenarios and collect user feedback to identify potential issues and opportunities for further enhancement. Regularly update the dataset and retrain the model to keep it up-to-date with evolving resume formats, industry trends, and user requirements.

Natural Language Processing using spacy

Examines job and user descriptive, unstructured text data using natural language processing techniques. Focuses mostly on explicit qualities. Due to its correctness, speed, and incorporation of word vectors such as en_vectors_web_lg, spaCy is used instead of NLTK. Analyses text data by separating named entities and vectors; job and user vectors are then compared using similarity methods. Depends on spaCy’s capability for semantic analysis includes vector representations made using Word2Vec word embedding. Recommendations are based on similarities found in the user profile and the job listing language. Trains a named entity recognition (NER) model to recognize talents using the OpenNLP framework. To obtain enough samples, training data is created by scraping more than 3000 job descriptions and 80 resumes. Contextual modeling retraining capabilities, and easier flexibility are advantages over rule-based systems. The requirement for substantial training data with human annotations is a drawback. For a good model, at least 15,000 sentences are needed. The end product is a trained model that can recognize and extract skills from text as named entities.

Neural Network Architecture for Job Skill Detection

Implements LSTM and BiLSTM models for sequence labeling to detect job skills in text. Takes in word embeddings, POS tags, and capitalization as input features. Outputs skill/not-skill label for each word. LSTM uses previous words as context, and BiLSTM uses words before and after for context. Helps deal with ambiguities. Multiple embedding dimensions and features were experimented with to optimize performance.

Clustering algorithms

Groups resumes and positions using clustering methods such as hierarchical clustering, k-means, and k-medoids. Various strategies, such as data point partitioning and density distribution modelling, are used in different clustering methodologies. Assists in identifying job and CV clusters that are more similar to one another than they are to one another. Offers groupings so that candidates and positions can be appropriately matched through analysis.

Word2Vec

Represents text data using Word2Vec word embeddings, allowing similarity calculations. Word vectors draw from a vast corpus to extract semantic meaning and usage context. CVs and job texts’ similarity can be detected by applying similarity functions to the word vectors. Compared to keyword/frequency comparisons alone, word vector space similarity yields better results. Word2Vec and Doc2Vec are the popular embedding algorithms for text. The difference between both of them is that in the case of word2vec, embedding is derived for a word, whereas, in the case of doc2vec, embedding is derived for the document.

The Gale-Shapley Algorithm, also known as the Stable Marriage Problem algorithm, is a fundamental tool in solving matching problems where two sets of elements need to be paired based on their preferences. Initially proposed to solve the problem of stable matching between an equal number of men and women in the context of marriage, its applications have extended to various fields beyond matrimonial matchmaking. Moreover, the algorithm finds utility in assigning students to schools, matching kidney donors with recipients, and pairing job seekers with employers, among other scenarios. At its core, the Gale-Shapley Algorithm provides a systematic method for creating stable matches that satisfy the preferences of both sides, ensuring that there are no pairs of individuals who would prefer each other over their assigned partners, thereby promoting fairness and efficiency in various allocation processes.
Fig. 1 Flowchart

Fig. 2 Count vectorization
IV. IMPLEMENTATION

**Data Collection:** Define Requirements: Clearly define the types of information your smart resume analyzer needs to extract. This could include personal information, work experience, education, skills, certifications, etc. Gather a diverse set of resumes from various sources such as job boards, career websites, LinkedIn profiles, academic databases, etc. Ensure that the resumes represent different industries, job roles, experience levels, and formats.

**Data Annotation:** Annotate the resumes to label each section (e.g., personal information, work experience) and extract relevant information. This can be done manually or using automated tools. For manual annotation, you may need a team of annotators to label each section consistently.

**Pre-processing:** Various stop words that are common to the CVs were removed as they do not provide relevant information to the algorithm to decide closeness to the job. The stop words removed were common words like education, skills, name of the student's educational institution, and hobbies. Part-of-speech tagging was done to check the sentence structure. POS tagging can be used to understand the structure of grammar and sifting for grammatical errors. The documents are also checked for spelling errors. Essentially, a sentence must consist of a noun phrase, verb and a subject or an auxiliary verb, subject, and a main verb. The sentences where this specified construct is not present are manually fixed. Lemmatization is also performed on the text obtained post-parsing the PDFs.

1. **Word cloud generation** is a visualization technique used to represent text data, where the size of each word indicates its frequency or importance within the text. Here's how you can generate a word cloud in Python using the word cloud library:

2. **Keyword extraction** Text from CV and Job Requirement PDFs: Open and read the PDF files containing the CV and job requirements. Use a pdf plumber to extract text from each page of the PDFs. Join the extracted text into a single string and replace newline characters for preprocessing.

   - **Text Preprocessing:** Preprocess the extracted text to remove unnecessary characters and format it into a cleaner string.
   - **Clean the Job Description Data:** Remove Non-Alphanumeric Characters: Remove any characters that are not letters or numbers. This helps to eliminate special characters, punctuation, and symbols that may not contribute to the understanding of the text.
   - **Convert to Lowercase:** Convert all text to lowercase. This ensures consistency and prevents the algorithm from treating words with different cases as different entities.
   - **Tokenization:** Tokenize the text into individual words or tokens. This step breaks down the text into its constituent parts, making it easier to process.
   - **Remove Stopwords:** Remove common stopwords from the text. Stopwords are words that occur frequently in a language (e.g., "the", "and", "is") but often do not carry significant meaning and can be safely ignored.
   - **Stemming or Lemmatization (Optional):** Apply stemming or lemmatization to reduce words to their base or root form. This step can help reduce the dimensionality of the data and improve the accuracy of the analysis by treating different forms of the same word as identical.
   - **Compute Cosine Similarity:** The function compute cosine similarity computes the cosine similarity between two text documents. It uses CountVectorizer from scikit-learn to convert the text into a matrix of token counts. The vectorizer is fit on the combined text of both documents. The resulting vectors are used to compute the cosine similarity using the cosine similarity function from scikit-learn.

3. **Output Match Percentage:** Calculate and print the match percentage between the CV and the job requirement based on cosine similarity.

4. **Application Building** - Streamlit is a popular Python library for building interactive web applications with simple Python scripts. It allows you to create data-driven web apps quickly and easily.

V. RESULT

**Data Collection:**
Identify Data Sources: Explore various sources and collect the dataset of resumes and job descriptions. Consider using APIs or web scraping techniques to collect resumes from online sources in bulk.
Data Quality Assurance:
Conduct quality checks to ensure the integrity and accuracy of the collected resumes. Remove duplicates and irrelevant documents to maintain a clean dataset.

Data Preprocessing:
Text Extraction: Extract text content from resume documents in various formats such as PDF, Word, or plain text. Utilize Optical Character Recognition (OCR) tools for scanned documents to convert images into machine-readable text.

Tokenization and Normalization: Tokenize the text into individual words or tokens. Normalize the text by removing stop words, punctuation, and non-alphanumeric characters. Apply stemming or lemmatization to reduce words to their base or root forms for better feature extraction.

Feature Extraction: Extract relevant features from the preprocessed text, such as word frequencies, TF-IDF scores, n-grams, or embeddings (e.g., Word2Vec). Consider incorporating domain-specific features or metadata (e.g., job titles, and years of experience) to enhance the model’s performance.

Data Splitting:
Divide the preprocessed dataset into training, validation, and test sets using stratified sampling to ensure representative distribution across different categories. Maintain consistent proportions between the sets (e.g., 70-80% for training, 10-15% for validation, and 10-15% for testing).

Classification algorithms:
For a resume analyzer based on AI, a classification algorithm is needed to categorize or extract information from resumes. Here are some classification algorithms commonly used in natural language processing (NLP) tasks like this:

Naive Bayes: Naive Bayes classifiers are simple and efficient for text classification tasks. They are based on Bayes’ theorem and assume that features are independent, which might not hold true in practice but often works well in practice for text data. Naive Bayes classifiers can be trained quickly and are suitable for large datasets.

Random Forest: Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It’s effective for text classification tasks due to its ability to handle high-dimensional data and nonlinear relationships. Random forests can provide insights into feature importance, which can be valuable for interpreting the model’s decisions.

Skill extraction:
Keyword Matching:
Use keyword-matching techniques to identify skill-related terms from the resume text. Maintain a dictionary or list of skill keywords and their synonyms to capture variations in skill terminology. Employ regular expressions or rule-based matching to detect skill mentions in the resume.

Named Entity Recognition (NER): Utilize NER models to identify and extract skill entities from resume text. Train or fine-tune NER models on annotated data to recognize skill-related entities accurately. Incorporate pre-trained NER models (e.g., spaCy, NLTK) and customize them to recognize specific skill categories.

Machine Learning Models:
Train machine learning models (e.g., SVM, Random Forest, or deep learning models) to classify resume sections or sentences containing skill information.

Skill matching: Job Skill Requirements Analysis:
Analyze job descriptions to identify required skills and qualifications for specific roles. Create a profile of desired skills and prioritize them based on their relevance and importance to the job.

Semantic Similarity: Calculate semantic similarity scores between resume skills and job requirements using techniques like Word Embeddings (e.g., Word2Vec, GloVe) or pre-trained language models (e.g., BERT). The Gale-Shapley Algorithm, also known as the Stable Marriage Problem algorithm, is a fundamental tool in solving matching problems where two sets of elements need to be paired based on their preferences. Initially proposed to solve the problem of stable matching. Moreover, the algorithm finds utility in assigning students to schools,
matching kidney donors with recipients, and pairing job seekers with employers, among other scenarios. At its core, the Gale-Shapley Algorithm provides a systematic method for creating stable matches. Measure the similarity between skill vectors in the resume and job description embeddings to assess skill relevance.

**Threshold-based Matching:**

Define a threshold or similarity score cutoff to filter and prioritize skill matches between the resume and job requirements. Consider additional factors such as the importance of the skill, domain expertise, and specific job context in the matching process.

**Percentage prediction**

**Feature Extraction:**

Count Vectorization: Convert text into numerical representations by counting the frequency of words. TF-IDF (Term Frequency-Inverse Document Frequency): Assign weights to words based on their frequency in the document and across documents.

**Modeling:**

Cosine Similarity: Calculate the cosine of the angle between the vectors representing the job description and resume. This measures the similarity between the two documents.

Machine Learning Models: Train a machine learning model (e.g., Support Vector Machines, Random Forest, Neural Networks) using the extracted features to predict the match percentage.

**Application building – Programming Languages:**

Python: It's widely used for natural language processing (NLP), machine learning, and web development. Streamlit is a Python library that makes it easy to create web applications for data science and machine learning tasks, allowing you to build interactive and responsive interfaces with minimal code.

**VI. CONCLUSION**

We focused on graph-based matching as an indicator for identifying comparable profiles, and our trials and evaluations proved the efficacy of our method. We do, however, recognize that skill extraction might be difficult due to erroneous matches, which highlights the necessity for ongoing development. Larger datasets from sites like LinkedIn will be incorporated in future studies, the Siamese Network will be improved on domain-specific skill descriptions, and key phrase extraction will be investigated to improve skill-matching memory. In this paper, we have proposed a job post and resume classification system (JRC) based on coupling an integrated skills knowledge base and an automatic matching procedure between candidate resumes and their corresponding job postings. By building a resume analyzer with Streamlit, we aim to empower job seekers with valuable insights into their resumes while providing recruiters with efficient tools for candidate screening. This project combines elements of web development, NLP, and data analysis, making it an exciting and impactful endeavor for aspiring developers and data scientists alike.

**VII. REFERENCES**


