K.R. MANGALAM UNIVERSITY

School of Engineering and Technology

Project Report On

RESUME SORTER USING NLP

Group Code ---- Y1-2024-25-G284

Academic Year: 2024–2025

Under the Guidance of :
Dr. Tanu Gupta
Assistant Professor
(SOET)

Submitted By:
Kajal Garg : 2401560029
Rekha Kumayan : 2401560029
Class : MCA

CERTIFICATE

This is to certify that the project titled **"Resume Sorter using NLP"** submitted by *Rekha Kumayan(2401560008), Kajal Garg(2401560029)* in partial fulfillment of the requirements for the award of the degree of
Master of Computer Applications from **K.R. Mangalam University** is a bona fide record of work carried out by them during the academic year **2024–2025** under my guidance.

Dr. Tanu Gupta
Guide

Head of Department

Sign_____

DECLARATION

We hereby declare that the project titled **"Resume Sorter Using NLP"** submitted to **K.R. Mangalam University, Gurugram**, in partial fulfillment of the requirements for the award of the degree of **Master of Computer Applications** is our original work.

This project has been carried out by us under the guidance of **Dr. Tanu Gupta**, School of Engineering and Technology.

We further declare that this work has not been submitted previously, in whole or in part, for the award of any degree, diploma, or any other similar title at this or any other university/institution.

All the information provided in this report is true to the best of our knowledge and belief. Proper references and acknowledgments have been given wherever necessary.

Submitted by:

Name	**Roll Number**
Rekha	2401560008
Kajal Garg	2401560029

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to **Dr. Tanu Gupta**, Faculty, School of Engineering and Technology, **K.R. Mangalam University**, for her invaluable guidance, encouragement, and constant support throughout the course of this project.
Her insightful suggestions, timely feedback, and immense knowledge greatly contributed to the successful completion of our work.

We are also deeply thankful to **K.R. Mangalam University** for providing the necessary facilities, resources, and an environment conducive to learning and research, which made this project possible.

We sincerely appreciate the cooperation and support of all the faculty members and staff who directly or indirectly helped us during this project.

Finally, we are immensely grateful to our families and friends for their continuous encouragement, understanding, and moral support throughout the duration of this project.

This project would not have been possible without the contributions of all these wonderful individuals.

TABLE OF CONTENTS

ABSTRACT

The **Resume Sorter using Natural Language Processing** is a smart recruitment tool designed to automate the screening and ranking of resumes based on job requirements. This system employs NLP techniques to extract key information such as skills, education, and experience from unstructured resume data. A central feature of the project is the use of *Latent Dirichlet Allocation (LDA)* for topic modeling, which helps in identifying the underlying themes and skillsets present in resumes. These extracted topics are then compared with job description topics to compute similarity scores, enabling accurate and unbiased ranking of candidates. By integrating LDA with other NLP techniques like tokenization, keyword extraction, and vector similarity, the system improves the efficiency and fairness of the recruitmen.process.

INTRODUCTION

1.1 Overview

The **Resume Sorter** is an automated system developed to streamline the recruitment process by intelligently filtering and ranking resumes based on their relevance to a given job description. Manual resume screening can be labor-intensive, especially when dealing with a large applicant pool. This project addresses that problem by using **Natural Language Processing** and **Latent Dirichlet Allocation (LDA)** to enhance decision-making during candidate selection.

The workflow begins with preprocessing resumes and job descriptions through NLP techniques such as tokenization, stop word removal, and lemmatization. The core of the system uses **LDA** for topic modeling to uncover hidden themes or topics within each resume and job description. These topics are used to represent each document in a structured format.

The similarity between the job description and each resume is then calculated using metrics like cosine similarity, allowing the system to score and rank candidates based on how closely their skills and experiences align with the job requirements.

This solution improves the accuracy, speed, and objectivity of resume screening. It can be particularly useful for HR teams, staffing agencies, and companies that handle high-volume recruitment.

Resume Dataset
|
Job Description
|
Text Preprocessing
-Tokenization
-Stop Word Removal
-Lemmatization
|
Topic Modelling Using LDA
-Extract Hidden Topics
-Represent docs as Topics
|
Resume Ranking & Sorting
- Based on match score
|
Final Output
-Sorted Resumes

1.2 Objective

The primary objective of this project is to develop an automated resume screening system that utilizes **Natural Language Processing** and **Latent Dirichlet Allocation (LDA)** to efficiently process and analyze resumes. This system aims to extract key information such as skills, experience, and qualifications from candidate resumes and match them against the requirements outlined in job descriptions. The specific objectives of this project are:

1. **Automate Resume Screening**: To reduce manual intervention and automate the process of sorting and ranking resumes, making the recruitment process faster and more scalable.

2. **Leverage NLP Techniques**: To apply NLP techniques such as tokenization, lemmatization, and stopword removal for efficient text preprocessing, making raw resume data machine-readable and analyzable.

3. **Topic Modeling with LDA**: To implement **Latent Dirichlet Allocation (LDA)**, a topic modeling algorithm, to identify key topics or themes present in resumes and job descriptions, ensuring that the system captures hidden patterns in the data.

4. **Improve Recruitment Efficiency**: To enhance the efficiency and accuracy of the recruitment process by reducing the time spent on manually sorting resumes and minimizing the potential for human bias in candidate selection.

5. **Offer Scalable and Reliable Solutions**: To provide a scalable and reliable system that can handle large volumes of resumes and job descriptions, making it adaptable for use in organizations of various sizes.

1.3 Scope

This project focuses on developing an automated resume sorting system leveraging **Natural Language Processing (NLP)** and **Latent Dirichlet Allocation (LDA)** to enhance the resume screening process. The scope of the project includes:

1. **Resume Preprocessing and Data Extraction**:
    a. The system will process resumes in various formats (e.g., DOCX, PDF, TXT) and extract structured data such as candidate information, skills, work experience, and educational background using NLP techniques.
    b. Preprocessing steps include tokenization, stopword removal, and lemmatization to prepare text data for further analysis.
2. **Topic Modeling Using LDA**:
    a. The project will employ **Latent Dirichlet Allocation (LDA)** to extract topics from resumes and job descriptions. LDA will help identify hidden themes or skills mentioned in resumes that are relevant to the job description.
    b. This topic modeling will create a richer representation of resume data, allowing the system to go beyond keyword matching.
3. **Resume Ranking Based on Job Description Matching**:
    a. The system will compare the extracted topics from resumes against the job description's topics to compute a similarity score.
    b. A ranking algorithm will be developed based on similarity scores, allowing the system to sort resumes from the most relevant to the least relevant for a given job.
4. **Improvement in Recruitment Efficiency**:
    a. The project will automate the initial phase of resume screening, reducing the time spent on manual review.
    b. It will help recruiters quickly identify top candidates, thereby speeding up the hiring process and enabling a more scalable solution to handle large volumes of applicants.

1.4 Motivation

The recruitment process often involves manually reviewing hundreds of resumes, which is time-consuming and prone to human bias. This inefficiency can delay hiring and result in suboptimal candidate selection. With the rise of large applicant pools, there is a pressing need for automated systems that can quickly and accurately process resumes.

This project leverages *Natural Language Processing* and *Latent Dirichlet Allocation (LDA)* to

automate resume screening, improving efficiency, reducing bias, and enhancing candidate-job matching. By using topic modeling, the system not only matches keywords but also identifies underlying skills and qualifications, ensuring a more objective and effective recruitment process.

The motivation behind this project is to streamline the hiring process, support scalability, and provide HR teams with better insights for decision-making.

LITERATURE REVIEW

2.1 Existing Resume Screening Methods

Traditional resume screening methods, though widely used, face several challenges in terms of efficiency, bias, and accuracy:

1. **Manual Screening**: Involves recruiters manually reviewing resumes, which is time-consuming and prone to bias and inconsistencies, resulting in subjective decision-making.

2. **Keyword-Based Matching**: Uses specific keywords from the job description to filter resumes. However, this approach lacks flexibility, missing relevant candidates who don't use exact keywords, and fails to understand context or synonyms.

3. **Rule-Based Systems**: Apply predefined rules (e.g., matching job titles or qualifications). These systems are rigid, unable to adapt to new terminology or assess qualitative aspects of resumes, leading to limited results.

4. **Applicant Tracking Systems (ATS)**: Automate basic resume sorting, primarily relying on keyword matching and formatting. ATS tools often misinterpret non-standard resume formats and fail to grasp the context of skills and qualifications.

5. **Hybrid Systems**: Combine ATS with basic machine learning for better ranking. While they offer improvements, they still require human validation and are limited by the quality of training data and manual rules.

2.2 Role of NLP in Recruitment

- **Resume Screening**: NLP automates resume analysis, identifying key qualifications, skills, and experience relevant to the job.

- **Candidate Matching**: NLP matches candidates to job descriptions by analyzing language patterns, ensuring a better fit than simple keyword matching.

- **Chatbots & Engagement**: NLP-powered chatbots engage candidates, answer questions, collect information, and conduct initial interviews.

- **Sentiment Analysis**: NLP evaluates candidate responses to gauge enthusiasm, motivation, and attitude for cultural fit.

- **Interview Analysis**: NLP analyzes interview responses, assessing communication, confidence, and articulation of ideas.

- **Job Description Optimization**: NLP improves job postings by identifying language that attracts the right candidates and ensuring inclusivity.

- **Bias Reduction**: NLP reduces unconscious bias by focusing on objective qualifications and experience, promoting diversity.

- **Skill Gap Analysis**: NLP identifies skill shortages, helping companies adjust recruitment strategies to meet workforce needs.

2.3 Topic Modeling in Text Analysis

Topic Modeling in Text Analysis is a technique used to identify hidden themes or topics within large collections of text. It is widely used in natural language processing (NLP) and machine learning to extract useful information from unstructured text data, such as articles, reviews, social media posts, and more. Here's an overview of key points related to topic modeling:

1. Purpose

Topic modeling helps uncover the underlying structure of text data by grouping words that frequently appear together. It aims to automatically discover topics that best represent the content in a collection of documents.

2. Popular Algorithms

- **Latent Dirichlet Allocation (LDA)**: The most commonly used method for topic modeling, LDA assumes each document is a mixture of topics and each topic is a mixture of words. It helps identify these hidden topics based on word distribution patterns.

- **Non-Negative Matrix Factorization (NMF)**: NMF factorizes a document-term matrix into two smaller matrices, focusing on word co-occurrence to find topics.

3. Applications

- **Content Summarization**: Automatically generating summaries based on key topics found within large text datasets.

- **Document Classification**: Categorizing documents based on the identified topics, enhancing search and retrieval.

- **Trend Analysis**: Identifying emerging topics or trends over time, useful in social media analysis or market research.

- **Recommendation Systems**: Suggesting content based on common topics between a user's preferences and available documents.

2.4 Gaps in Current Systems

Current systems, whether in AI, data analytics, or other domains, often exhibit certain gaps that hinder their effectiveness and limit their potential. Here are some of the key gaps in current systems:

1. Data Quality and Availability

- **Gap**: Many systems rely on incomplete, outdated, or noisy data, which impacts the accuracy and reliability of insights generated.
- **Impact**: Inconsistent or biased data can lead to incorrect predictions, decisions, and outcomes.

2. Scalability and Performance

- **Gap**: Current systems may struggle to scale efficiently as data volume, complexity, or user demands grow.
- **Impact**: Slow processing times or system crashes can hinder productivity and affect user experience.

3. Interoperability

- **Gap**: Many systems are not fully compatible with each other, especially when integrating new technologies or tools.
- **Impact**: Difficulty in data sharing, integration, or collaboration across different platforms or software can lead to inefficiencies and duplicated efforts.

4. User Experience (UX)

- **Gap**: Existing systems may have complex interfaces that are not user-friendly or intuitive, limiting their accessibility for non-expert users.
- **Impact**: This can lead to lower adoption rates and decreased effectiveness in decision-making.

5. Bias and Fairness

- **Gap**: Many systems, particularly in AI and machine learning, can unintentionally perpetuate biases based on the data they are trained on.
- **Impact**: This results in unfair outcomes, such as discrimination in recruitment, lending, or criminal justice.

6. Explainability and Transparency

- **Gap**: Many AI and machine learning models, especially deep learning models, are often seen as "black boxes," making it difficult to understand how they arrive at decisions.

- **Impact**: Lack of transparency limits trust in these systems, especially in critical applications like healthcare, finance, and legal domains.

7. Security and Privacy

- **Gap**: Systems often face vulnerabilities in protecting sensitive data from breaches or misuse.

- **Impact**: Data breaches and privacy violations can damage reputations and violate regulations, particularly with the growing concerns around data protection laws.

8. Adaptability to Change

- **Gap**: Many systems are rigid and struggle to adapt to changing business needs, technology advancements, or external conditions.

- **Impact**: This limits the long-term usefulness of the system as it may become outdated or irrelevant over time.

SYSTEM ANALYSIS

3.1 Problem Definition

In today's recruitment landscape, Human Resource (HR) departments face the daunting task of processing hundreds or even thousands of resumes for a single job opening. Manual screening of resumes is time-consuming, prone to human error, and often inconsistent due to subjective judgments. Additionally, valuable candidates may be overlooked due to inefficient keyword-based search or unconscious bias.

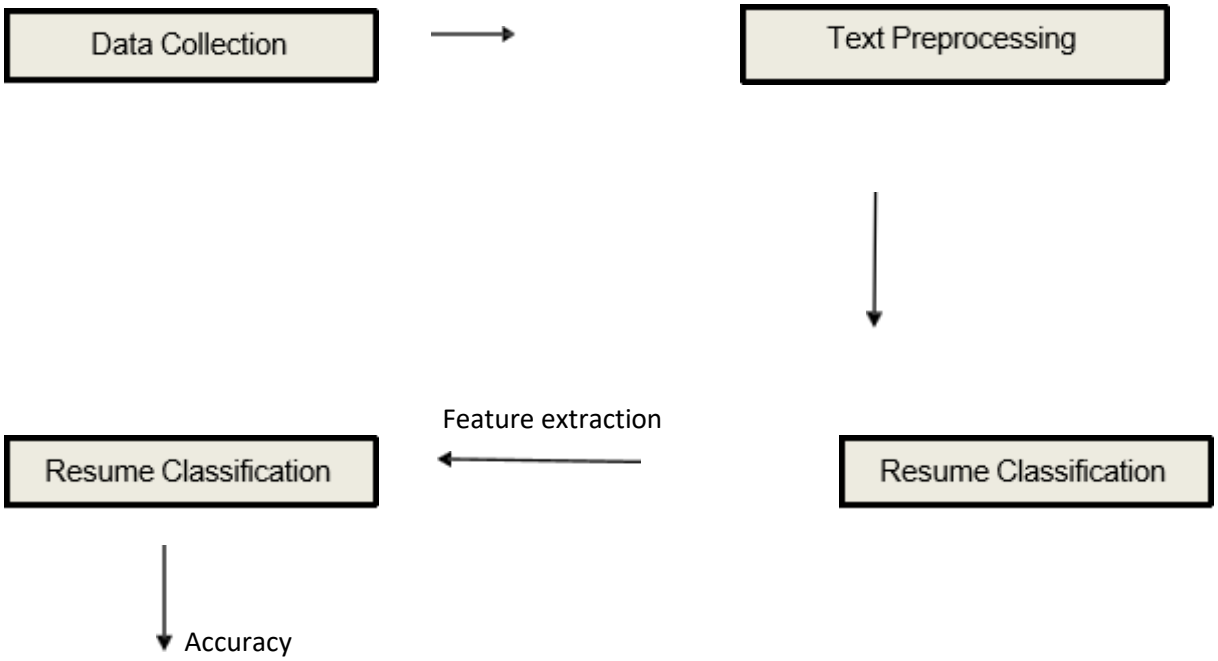
Traditional Applicant Tracking Systems (ATS) rely heavily on rigid keyword matching and lack contextual understanding, resulting in inaccurate filtering and irrelevant candidate shortlisting. This leads to suboptimal hiring decisions, wasted time, and increased costs for organizations.

3.2 Feasibility Study

The proposed NLP-based Resume Sorter is feasible across multiple dimensions:

- **Technical Feasibility**: Uses mature technologies like Python, NLP libraries (SpaCy, NLTK), and ML models. Cloud support ensures scalability.
- **Operational Feasibility**: Easily integrates with HR workflows, improving efficiency and reducing manual workload.
- **Economic Feasibility**: Low development cost due to open-source tools; saves time and resources in the long term.
- **Legal & Ethical Feasibility**: Designed to comply with data protection laws and minimize bias for fair hiring.
- **Schedule Feasibility**: Can be developed and deployed within a reasonable time frame using agile methods.

3.3 Workflow Diagram



METHODOLOGY

4.1 Data Collection

Data collection involves gathering resumes and job descriptions to train and test the system. The sources include:

- **Public Datasets**: Open-source resume datasets from Research papers.
- **Job Portals**: Scraped or manually collected job descriptions from sites like LinkedIn, Indeed, etc.
- **Synthetic Data**: Artificial resumes and job postings created to simulate various job roles.
- **User Inputs**: Optionally, anonymized resumes submitted by users for testing.

4.2 Preprocessing Techniques

Preprocessing is crucial to ensure that text data is clean, standardized, and ready for analysis. The following techniques are applied to resume and job description data:

1. **Lowercasing**:
   Converts all text to lowercase for consistency, as "Python" and "python" should be treated as the same word.

2. **Tokenization**:
   Breaks text into smaller units (tokens), typically words or phrases, which makes it easier to analyze the content of resumes.

3. **Stopword Removal**:
   Eliminates common, uninformative words (e.g., “and,” “the,” “in”), which do not contribute to the meaningful analysis of resumes or job descriptions.

4. **Lemmatization**:
   Reduces words to their root form. For example, “running” is converted to “run,” ensuring uniformity and reducing dimensionality in the dataset.

5. **Punctuation and Special Character Removal**:
   Removes unnecessary characters, like punctuation marks, numbers, or special symbols that may not add value to the text analysis.

6. **Vectorization (TF-IDF)**:
   Converts the text data into a numerical format using **TF-IDF** (Term Frequency-Inverse Document Frequency). This helps highlight the important terms in a document while reducing the weight of common, less informative words.

7. **Topic Modeling (LDA)**:
   Latent Dirichlet Allocation (LDA) is used to discover latent topics within resumes and job descriptions. This is done by analyzing the co-occurrence patterns of words

across multiple documents (resumes/job descriptions) and grouping them into topics. Each resume is treated as a mixture of these topics. The extracted topics allow the system to understand the key themes and match resumes with job descriptions based on topic similarity.

4.3 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is a **topic modeling** technique that identifies hidden topics within a collection of documents (resumes or job descriptions). LDA assumes each document is a mixture of several topics, and each topic is a distribution over words. The goal is to extract these underlying topics to better understand the context and content of the documents.

Steps in LDA:

1. **Topic Extraction**:
   LDA identifies a set of topics based on word co-occurrence patterns in the text. For example, a resume about software engineering might have topics like "programming," "databases," and "teamwork."

2. **Document Representation**:
   Once topics are identified, LDA assigns a topic distribution to each document (resume/job description), indicating the proportion of each topic in that document.

3. **Topic-Document Matching**:
   By analyzing the topic distributions of resumes and job descriptions, the system can match resumes to jobs with similar topic distributions, improving the accuracy of candidate-job fit.

4.5 Resume Ranking Algorithm

The **Resume Ranking Algorithm** is designed to rank resumes based on their relevance to a specific job description. The algorithm uses various features, including keyword matching, semantic similarity, and topic modeling (via LDA), to assign a score to each resume. Here's how it works:

Steps in the Resume Ranking Process:

1. **Preprocessing**:
   Resumes and job descriptions undergo preprocessing (tokenization, lemmatization, stopword removal) to prepare the text for analysis.

2. **TF-IDF_Vectorization**:
   Both resumes and job descriptions are converted into numerical vectors using **TF-IDF**. This step emphasizes important, unique words while downplaying common terms.

3. **Topic_Modeling_(LDA)**:
   LDA is used to identify underlying topics in both resumes and job descriptions. Each document is represented as a distribution of topics, capturing the semantic meaning beyond specific keywords.

IMPLEMENTATION

5.1 Tools and Libraries Used

The following tools and libraries were employed to develop the Resume Sorter:

1. **Python**:
   The primary programming language used due to its vast ecosystem of libraries for NLP and machine learning.

2. **Natural Language Toolkit (NLTK)**:
   A comprehensive library used for basic text preprocessing tasks such as tokenization, stopword removal, and lemmatization.

3. **SpaCy**:
   Used for advanced NLP tasks, including tokenization, named entity recognition, and lemmatization. SpaCy is fast and efficient for large-scale text processing.

4. **Gensim**:
   Utilized for **Latent Dirichlet Allocation (LDA)**, which is a topic modeling technique to discover hidden topics within resumes and job descriptions. Gensim enables topic extraction by analyzing word distributions across documents.

5. **Scikit-learn**:
   A robust machine learning library used for **TF-IDF vectorization** and cosine similarity computation. It provides simple tools for transforming textual data into numerical vectors and comparing document similarities.

6. **Pandas**:
   Used for handling and manipulating structured data (resumes, job descriptions) in a tabular format, making it easier to process and analyze the data.

7. **NumPy**:
   Supports efficient numerical operations on arrays and matrices, which are crucial for performing computations in NLP tasks and machine learning algorithms.

5.2 Code Snippets .

```python
#READ DATASET
import pandas as pd
#df=pd.read_csv(r'C:\Users\91813\Downloads\Dataset.csv')
df = pd.read_csv(r'C:\\Users\\DELL\\Downloads\\gpt_dataset_new.csv')



#CHECK FOR MISSING VALUES OR NULL VALUES
print(df['Resume'].isna().sum())
df['Resume'] = df['Resume'].fillna(' ')



#TOKENIZATION
import nltk
from nltk.tokenize import word_tokenize
nltk.download('punkt')
nltk.download('punkt_tab')
def safe_tokenize(df):
    if isinstance(df, str):
        return word_tokenize(df)
    else:
        return []   #Return an empty list if the text is not a string
df['tokens'] = df['Resume'].apply(safe_tokenize)
print(df[['Resume', 'tokens']])
```

```python
#STOPWORD REMOVAL
from nltk.corpus import stopwords
nltk.download('stopwords')
def remove_stopwords(text):
    tokens = word_tokenize(str(text))
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [word for word in tokens if word.lower() not in stop_wor
    return filtered_tokens
df['filtered_tokens'] = df['Resume'].apply(remove_stopwords)
print(df[['Resume', 'filtered_tokens']])



#LEMMATIZATION
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()
def preprocess_text(text):
    tokens = word_tokenize(str(text))
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [lemmatizer.lemmatize(word.lower()) for word in tokens
                       if word.lower() not in stop_words]
    return filtered_tokens
df['processed_tokens'] = df['Resume'].apply(preprocess_text)
print(df[['Resume', 'processed_tokens']])
```

```
#IMPORT GENSIM
from gensim import corpora
from gensim.models import LdaModel
import re



# FUNCTION TO CLEAN TEXT
def clean_text(text):
    return re.sub(r'[^a-zA-Z0-9\s]', '', text)
df['cleaned_resume'] = df['Resume'].apply(clean_text)



# CREATE DICTIONARY AND CORPUS FOR LDA
dictionary = corpora.Dictionary(df['processed_tokens'])
corpus = [dictionary.doc2bow(text) for text in df['processed_tokens']]



# PERFORM LDA
num_topics = 5
lda_model = LdaModel(corpus, num_topics=num_topics, id2word=dictionary, passes
topics = lda_model.print_topics(num_words=10)
for topic in topics:
    print(topic)
```

```
#ASSIGN TOPICS TO EACH DOCUMENT
def get_document_topics(bow):
    return sorted(lda_model.get_document_topics(bow), key=lambda x: -x[1])[:1]
df['dominant_topic'] = df['processed_tokens'].apply(lambda x: get_document_top



#DISPLAY SAMPLE OF RESULTS
print(df[['Resume', 'dominant_topic']].head())
df['dominant_topic'] = df['dominant_topic'].apply(lambda x: x[0][0] if x else



# TRAIN A CLASSIFIER USING THE DOMINANT TOPIC AS THE TARGET VARIABLE
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
X_train, X_test, y_train, y_test = train_test_split(df['processed_tokens'], df



# CONVERT PROCESSED TOKENS TO FEATURED VECTORS
def tokens_to_features(tokens):
    return dictionary.doc2bow(tokens)
X_train = [tokens_to_features(tokens) for tokens in X_train]
X_test = [tokens_to_features(tokens) for tokens in X_test]
```

```python
# CONVERT BOW TO DENSE VECTORS
from gensim.matutils import corpus2dense
X_train_dense = corpus2dense(X_train, num_terms=len(dictionary)).T
X_test_dense = corpus2dense(X_test, num_terms=len(dictionary)).T



# TRAIN USING  RANDOM FOREST CLASSIFIER
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train_dense, y_train)
y_pred = clf.predict(X_test_dense)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))



# TRAIN USING SVM CLASSIFIER
from sklearn.svm import SVC
svm_clf = SVC(kernel='linear', random_state=42)
svm_clf.fit(X_train_dense, y_train)
y_pred_svm = svm_clf.predict(X_test_dense)
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print("SVM Classification Report:\n", classification_report(y_test, y_pred_svm
```

```python
# TRAIN USING NAIVE BAYES CLASSIFIER
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
nb_clf = MultinomialNB()
nb_clf.fit(X_train_dense, y_train)
y_pred_nb = nb_clf.predict(X_test_dense)
print("Naive Bayes Accuracy:", accuracy_score(y_test, y_pred_nb))
print("Naive Bayes Classification Report:\n", classification_report(y_test, y_
```

```python
#TRAIN USING KNN CLASSIFIER
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
# Train KNN Classifier
knn_clf = KNeighborsClassifier(n_neighbors=5)
knn_clf.fit(X_train_dense, y_train)
y_pred_knn = knn_clf.predict(X_test_dense)
print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn))
print("KNN Classification Report:\n", classification_report(y_test, y_pred_knn
```

```python
#TRAIN USING LOGISTIC REGRESSION
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(max_iter=1000, random_state=42)
log_reg.fit(X_train_dense, y_train)
y_pred_lr = log_reg.predict(X_test_dense)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))
print("Logistic Regression Classification Report:\n", classification_report(y_
```

5.3 Output Screenshots

```
In [1]: runfile('C:/Users/DELL/Downloads/LDA.py', wdir='C:/Users/DELL/Downloads')
0
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
                                               Resume                                             tokens
0     As a seasoned Frontend Developer, I have a pro...  [As, a, seasoned, Frontend, Developer, ,, I, h...
1     With a solid background in Backend Development...  [With, a, solid, background, in, Backend, Deve...
2     As a Python Developer, I leverage my expertise...  [As, a, Python, Developer, ,, I, leverage, my,...
3     With a background in Data Science, I possess a...  [With, a, background, in, Data, Science, ,, I,...
4     Experienced Frontend Developer with a passion ...  [Experienced, Frontend, Developer, with, a, pa...
...                                                 ...                                                ...
1357  Computer Skills: â¢ Proficient in MS office (...  [Computer, Skills, :, â¢, Proficient, in, MS,...
1358  â Willingness to accept the challenges. â ...  [â, Willingness, to, accept, the, challenges...
1359  PERSONAL SKILLS â¢ Quick learner, â¢ Eagerne...  [PERSONAL, SKILLS, â¢, Quick, learner, ,, â¢...
1360  COMPUTER SKILLS & SOFTWARE KNOWLEDGE MS-Power ...  [COMPUTER, SKILLS, &, SOFTWARE, KNOWLEDGE, MS-...
1361  Skill Set OS Windows XP/7/8/8.1/10 Database MY...  [Skill, Set, OS, Windows, XP/7/8/8.1/10, Datab...

[1362 rows x 2 columns]
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
                                               Resume                                    filtered_tokens
0     As a seasoned Frontend Developer, I have a pro...  [seasoned, Frontend, Developer, ,, proven, tra...
1     With a solid background in Backend Development...  [solid, background, Backend, Development, ,, b...
2     As a Python Developer, I leverage my expertise...  [Python, Developer, ,, leverage, expertise, Py...
3     With a background in Data Science, I possess a...  [background, Data, Science, ,, possess, unique...
4     Experienced Frontend Developer with a passion ...  [Experienced, Frontend, Developer, passion, cr...
...                                                 ...                                                ...
1357  Computer Skills: â¢ Proficient in MS office (...  [Computer, Skills, :, â¢, Proficient, MS, off...
1358  â Willingness to accept the challenges. â ...  [â, Willingness, accept, challenges, ., â,...
1359  PERSONAL SKILLS â¢ Quick learner, â¢ Eagerne...  [PERSONAL, SKILLS, â¢, Quick, learner, ,, â¢...
1360  COMPUTER SKILLS & SOFTWARE KNOWLEDGE MS-Power ...  [COMPUTER, SKILLS, &, SOFTWARE, KNOWLEDGE, MS-...
1361  Skill Set OS Windows XP/7/8/8.1/10 Database MY...  [Skill, Set, OS, Windows, XP/7/8/8.1/10, Datab...

[1362 rows x 2 columns]
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
                                               Resume     dominant_topic
0  As a seasoned Frontend Developer, I have a pro...  [(2, 0.93920493)]
1  With a solid background in Backend Development...  [(2, 0.99179244)]
2  As a Python Developer, I leverage my expertise...   [(2, 0.9924329)]
3  With a background in Data Science, I possess a...   [(2, 0.9932436)]
4  Experienced Frontend Developer with a passion ...   [(2, 0.9927867)]
Accuracy: 0.9890109890109891
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        27
           1       0.97      1.00      0.99        68
           2       1.00      1.00      1.00       100
           3       1.00      1.00      1.00        17
           4       0.98      1.00      0.99        61

    accuracy                           0.99       273
   macro avg       0.99      0.98      0.98       273
weighted avg       0.99      0.99      0.99       273
```

CONCLUSION

The **Resume Sorter Using NLP and LDA** project aims to address the inefficiencies and biases inherent in traditional resume screening methods. By leveraging **Natural Language Processing (NLP)** for text preprocessing and **Latent Dirichlet Allocation (LDA)** for topic modeling, this system automates the extraction of relevant skills, experiences, and qualifications from resumes, ensuring a more accurate and objective matching process with job descriptions.

The system not only speeds up the recruitment process but also reduces human bias and enhances the quality of candidate selection by analyzing the underlying themes and context in resumes, rather than relying solely on keywords. This approach provides a scalable, data-driven solution for recruiters, ultimately leading to more informed, efficient, and fair hiring decisions.

Future work may involve expanding the system's capabilities to handle multiple languages, integrate with other HR tools, and further refine machine learning models to improve accuracy. Overall, this project demonstrates the potential of NLP and machine learning in revolutionizing the recruitment process.

REFERENCES

1. *Saatci, M., Kaya, R., & Ünlü, R. (2024). Resume Screening With Natural Language Processing (NLP). Alphanumeric Journal, 12(2), 121-140*
2. *Gan, C., Zhang, Q., & Mori, T. (2024). Application of LLM Agents in Recruitment: A Novel Framework for Automated Resume Screening. Journal of Information Processing, 32(0), 881–893.*
3. *Naveed, Z., Nisar, B., Saifullah, D. M., & Iqbal Baig, J. (2024). Resume Ranking Using Natural Language Processing. Journal of Computers and Intelligent Systems, 2(1), 61– 66.*
4. *Lalitha, B., Kadiyam, S., Kalidindi, R. V., Vemparala, S. M., Yarlagadda, K., & Chekuri, S. V. (2023). Applicant Screening System Using NLP. 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA), 379–383.*
5. *Li, S., Li, K., & Lu, H. (2023). National Origin Discrimination in Deep/learning/powered Automated Resume Screening. 1–15.*
6. *Mohanty, S., Behera, A., Mishra, S., Alkhayyat, A., Gupta, D., & Sharma, V. (2023). Resumate: A Prototype to Enhance Recruitment Process with NLP based Resume Parsing. 2023 4th International Conference on Intelligent Resume Screening with Natural Language Processing (NLP) | Saatçı et al., 2024 neering and Management (ICIEM), 1–6*
7. *Ali, I., Mughal, N., Khan, Z. H., Ahmed, J., & Mujtaba, G. (2022). Resume Classification System using Natural Language Processing and Machine Learning Techniques. Mehran University Research Journal of Engineering and Technology, 41(1), 65–79*
8. *Anand, A., & Dubey, M. S. (2022). CV Analysis Using Machine Learning. International Journal for Research in Applied Science and Engineering Technology, 10(5),1316*