VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Belagavi-590018

Mini Project Report on

" Candidate Eligibility Analysis based on Resume Classification using LSTM "

Submitted in partial fulfillment as per VTU curriculum for **VI Semester**
For the award of degree of

Bachelor of Engineering
In
Artificial Intelligence and Data Science

Submitted by

ANUSHA G (1EP21AD007)
LAVANYA U (1EP21AD028)
R THANUJA REDDY (1EP21AD039)

Under the Guidance of
Dr. Sudhanshu Saurabh
Associate Professor
Dept. of AI&DS, EPCET

EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

Department of Artificial Intelligence & Data Science Jnana Prabha Campus, Virgo Nagar Post, Bidarahalli. Bengaluru – 560049 2023-2024

EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

Department of Artificial Intelligence and Data Science

Certificate

This is to certify that **ANUSHA G (1EP21AD007), LAVANYA U (1EP21AD028),** and **R THANUJA REDDY (1EP21AD039),** students of East Point College of Engineering and Technology have completed the Mini Project in partial fulfilment for the award of **Bachelor of Engineering** in **Artificial Intelligence & Data Science** of the Visvesvaraya Technological University, Belgaum during the year **2023-2024**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report. This Internship Report has been approved as it satisfies the academic requirements in respect of Mini Project work carried out for 6th semester of Bachelor of Engineering degree.

Dr. Sudhanshu Saurabh

Associate Professor
Department of AI & DS
EPCET, Bengaluru-49

Dr. Anand R

Professor & HOD
Department of AI & DS
EPCET, Bengaluru-49

EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise doesn't depend solely on the individual efforts but on the guidance, encouragement and cooperation of the intellectuals, elders and friends.
I would like to take this opportunity to thank them all.

First and foremost, I would like to thank Late Dr. S M Venkatapathi, chairman, East Point Group of Institution, Bengaluru, for providing necessary infrastructure and creating a good environment.

I extend my heart-full gratitude to our honourable CEOs, Shri S V Pramod Gowda and Shri S V Raijv Gowda for providing required support.

I express my gratitude to Dr. Mrityunjaya V Latte , Principal, EPCET who has always been a great source of inspiration.

I express our very sincere regards and thanks to Dr. Anand R , Head of the department, AI- DS, EPCET.

I extend my sincere thanks to all the faculty members of AI-DS Department, EPCET, who have encouraged us throughout the course. I also express my deep sense of obligation to my parents and God for their consistent blessings and encouragement.

EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

DECLARATION

I, **ANUSHA G [1EP21AD007]**, **LAVANYA U [1EP21AD028] ,** and **R THANUJA REDDY [1EP21AD039]** students of VI Semester BE in Artificial Intelligence and Data Science, East Point College of Engineering and Technology hereby declare that the Mini Project entitled "**Online Reviews Sentiment Analysis"** has been carried out by me and submitted in partial fulfillment of the requirements of the VI Semester for the award of degree of **Bachelor of Engineering in Artificial Intelligence and Data Science** of Visvesvaraya Technological University, Belagavi during academic year 2023-2024.

Date :
Place : Bengaluru

ANUSHA G (1EP21AD007)
LAVANYA U (1EP21AD028)
R THANUJA REDDY (1EP21AD039)

ABSTRACT

This project focuses on predicting the eligibility of job candidates based on their resumes using advanced deep learning techniques, specifically Long Short-Term Memory (LSTM) networks and Deep Neural Networks (DNN). By preprocessing resume data, extracting relevant features, and applying sophisticated machine learning models, we aim to develop a robust system capable of accurately classifying resumes into eligible and non-eligible categories. This automated approach seeks to streamline the resume screening process, thereby reducing the time and effort required from human recruiters.

The process begins with the standardization and splitting of the dataset into training and testing sets. We employ various feature extraction techniques, such as text analysis and keyword extraction, to enhance the dataset's predictive capabilities.

Multiple machine learning algorithms, including LSTM networks and DNN, are implemented and rigorously evaluated to determine the most effective classifier for eligibility prediction.

The primary objective of this project is to achieve high accuracy in classifying resumes, offering valuable insights into the performance of different algorithms. The findings demonstrate that advanced machine learning techniques, particularly deep learning models, substantially enhance classification accuracy.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Overview Of Resume Classification

The recruitment process is a critical function within any organization, serving as the gateway to acquiring top talent essential for achieving strategic objectives. Unfortunately, traditional resume screening methods have become a significant bottleneck, impeding the efficiency and effectiveness of the hiring process.

Manually reviewing a deluge of resumes is not only time-consuming but also error-prone. Recruiters often struggle to identify the most promising candidates amidst a sea of applications, leading to delays in filling critical positions. Moreover, relying solely on keyword matching to assess candidate qualifications is a simplistic approach that fails to capture the full spectrum of a candidate's skills, experience, and potential. This superficial evaluation often results in overlooking qualified individuals who may not possess the exact keywords but possess transferable skills and a strong aptitude for the role.

Compounding these challenges is the pervasive issue of unconscious bias, which can seep into the selection process and unfairly disadvantage certain candidates. Stereotypes and preconceived notions can inadvertently influence decision-making, leading to the exclusion of qualified individuals from diverse backgrounds. This not only undermines organizational diversity and inclusion efforts but also limits the talent pool from which to choose.

To address these limitations, organizations must explore innovative approaches to talent acquisition that leverage technology and data-driven insights to streamline the screening process, mitigate bias, and enhance the overall quality of hires.

1.2 Candidate Eligibility Classification Using LSTM Networks

Deep neural networks, particularly Long Short-Term Memory (LSTM) networks, have significantly transformed the process of candidate eligibility classification, enabling automated and efficient screening of job resumes. This process is crucial for businesses aiming to streamline their recruitment workflow, ensuring that the most suitable candidates are shortlisted

efficiently. By analyzing resume content, LSTM networks can determine whether a candidate meets the job requirements, thereby automating and enhancing the hiring process.

1.3 Introduction to LSTM

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture capable of learning long-term dependencies. Unlike traditional RNNs, LSTMs have a special cell structure that allows them to selectively remember or forget information over extended periods, making them highly effective for handling sequential data.

The Challenge with Traditional RNNs is that the traditional Recurrent Neural Networks (RNNs) are designed to process sequential data, but they encounter a significant limitation: the vanishing gradient problem. This issue arises as the network processes information from earlier points in the sequence. The gradients used to update the network's parameters become progressively smaller, hindering the network's ability to learn and capture long-term dependencies within the data.

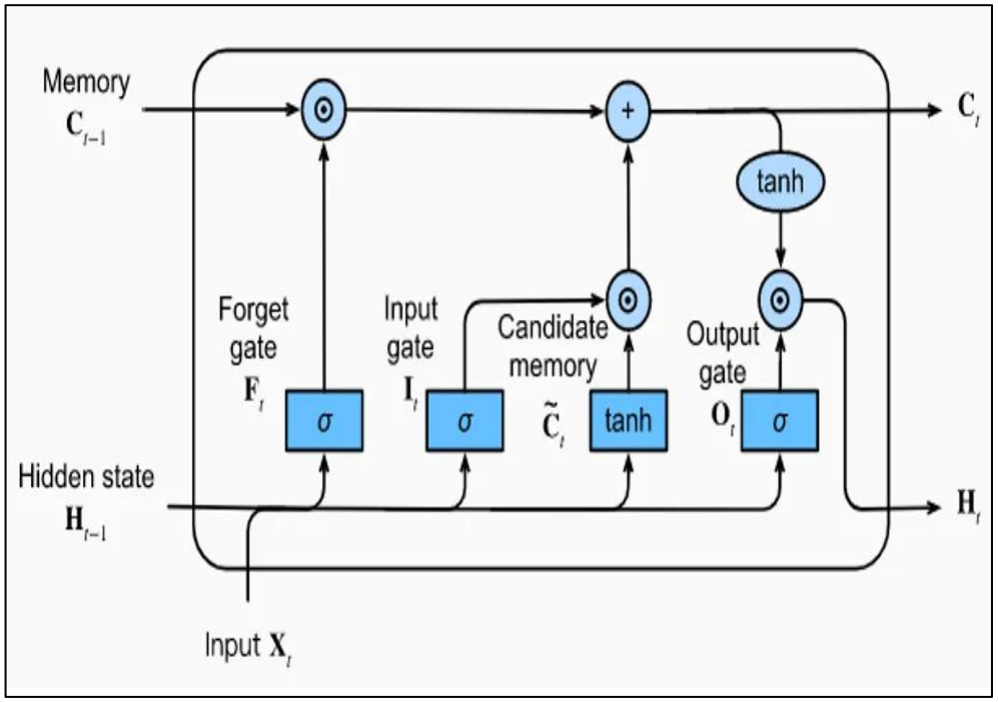

Fig 1.3: LSTM Unit Architecture

LSTMs address the limitations of traditional RNNs by introducing a sophisticated cell structure. The core components of an LSTM unit are:

- **Cell State:** This is the heart of the LSTM, running through the entire unit. It acts as a conveyor belt, carrying information along the sequence.
- **Input Gate:** Determines which information from the input will be stored in the cell state.
- **Forget Gate:** Decides which information from the cell state should be forgotten.
- **Output Gate:** Controls which information from the cell state is output.

1.4 Advantages of LSTM-Based Candidate Eligibility Assessment Over Traditional Methods

➢ **Enhanced Accuracy and Precision**

- **Comprehensive Profile Evaluation:** LSTMs process entire resumes, capturing the full spectrum of a candidate's skills, experiences, and career progression. This leads to a more accurate and nuanced assessment of their qualifications.
- **Skill Alignment:** By understanding the context of skills and experiences, our model effectively matches candidates to job requirements, reducing the risk of mismatches.
- **Predictive Analytics:** Leveraging historical data, LSTMs can predict candidate performance, enabling more informed hiring decisions.

➢ **Reduced Bias and Fairer Hiring**

- **Objective Evaluation:** LSTMs make decisions based on data and patterns, minimizing the impact of human biases that can influence traditional screening processes.
- **Diverse Candidate Pools:** Our model can identify qualified candidates from diverse backgrounds, promoting inclusivity and fairness.

➢ **Data-Driven Insights**

- **Performance Metrics:** The system provides valuable insights into candidate trends, skill gaps, and hiring effectiveness.
- **Continuous Improvement:** By analyzing hiring data, we can refine the model over time to enhance its accuracy and predictive capabilities.

1.5 Problem Statement

Manual screening of job resumes for skills and technologies is laborious and inefficient, particularly in high-volume hiring. Keyword-based methods often miss qualified candidates due to variations in skill descriptions, leading to biases and inefficiencies. We propose an LSTM-based machine learning solution to automate resume classification. Our approach aims to reduce dataset dimensions and optimize the use of LSTM models for efficient, accurate, and unbiased resume screening, thereby freeing recruiters' time for deeper candidate evaluation.

1.6 Objectives

The goal is to develop a machine learning model for real-time candidate eligibility classification by analyzing resumes. The model aims to achieve high accuracy by leveraging LSTM (Long Short-Term Memory) to capture the temporal dependencies in resume data and predict the best fit for job positions.

- **Business Objectives and Constraints**

  - **High Stakes in Misclassification**: Misclassifying candidates can result in hiring unsuitable candidates or missing out on highly qualified ones. Therefore, the model must prioritize high accuracy and minimize both false positives and false negatives to ensure the best hiring decisions.
  - **Scalability and Integration**: The solution should be scalable to handle large volumes of resumes, making it suitable for organizations of all sizes. Additionally, it must integrate seamlessly with existing Applicant Tracking Systems (ATS) for easy adoption and implementation.
  - **User-Friendly Interface**: The end product should feature a user-friendly interface, whether as an app or web interface, enabling HR professionals to utilize it with minimal training and effort.
  - **Data Privacy and Continuous Learning**: Given the sensitivity of resume data, the solution must adhere to strict data privacy and security standards. Additionally, the model should support continuous learning, allowing it to be updated with new data to improve accuracy over time.

CHAPTER 2
LITERATURE REVIEW

This section reviews the literature that has previously been done in resume classification, provides an overview of existing knowledge in this particular field of research.

Recent research has explored more sophisticated approaches for resume classification, such as Resume Classification Using Bidirectional LSTM and Attention Mechanism by Swayami Bera, Biraj Ghosh, D. Vanusha , this paper explores the use of Bidirectional Long Short-Term Memory (LSTM) combined with an attention mechanism to enhance resume classification accuracy. The attention mechanism allows the model to focus on the most relevant parts of the text, leading to improved feature extraction and classification performance. The authors demonstrate that this combined approach significantly boosts the model's ability to accurately classify resumes, making it a valuable tool for automating the recruitment process. [1]

Abstractive Text Summarization for Resumes with Cutting Edge NLP Transformers and LSTM by Oykü Berfin Mercan, Senem Tanberk , the authors propose a hybrid model that integrates Long Short-Term Memory (LSTM) networks and advanced Natural Language Processing (NLP) transformers for resume summarization. This model generates concise summaries of resumes, enabling recruiters to quickly assess candidates' qualifications. The hybrid approach leverages the sequential processing capabilities of LSTM and the contextual understanding of transformers, resulting in high-quality summaries that retain essential information. [2]

Resume Recommendation Using RNN Classification and Cosine Similarity by Lavanya P., Sasikala E , this paper presents a recommendation system that utilizes Recurrent Neural Networks (RNN) for resume classification and Cosine Similarity for matching resumes to job descriptions. The approach aims to enhance the accuracy of job recommendations by considering the contextual information within resumes. By combining RNN's ability to handle sequential data with the precise matching capabilities of Cosine Similarity, the proposed system improves the relevance of job recommendations. [3]

Automated Resume Classification Using Machine Learning by Pradeep Kumar Roy, Sunil Kumar Singh, Tapan Kumar Das, Asis Kumar Tripathy,t he authors propose a Deep Convolutional Neural

Network (CNN) model for automated resume classification. The model is designed to process large-scale resume datasets and improve the efficiency of the recruitment process by accurately categorizing resumes into predefined categories. The use of deep CNN enables the model to capture complex patterns in the data, leading to robust performance in classifying resumes and streamlining the hiring workflow. [4]

Authors	| Title	Year	Model Implemented	Detailed Summary
Swayami Bera, Biraj Ghosh, D. Vanusha	Resume Classification Using Bidirectional LSTM And Attention Mechanism	2022	Bidirectional LSTM And Attention Mechanism	This Paper Explores The Use Of Bidirectional LSTM Combined With An Attention Mechanism To Improve Resume Classification Accuracy. The Attention Mechanism Helps The Model Focus On Relevant Parts Of The Text, Leading To Better Feature Extraction And Classification Performance. |
| Oykü Berfin Mercan, Senem Tanberk	Abstractive Text Summarization For Resumes With Cutting Edge NLP Transformers And LSTM	2023	LSTM And NLP Transformers	The Authors Propose A Hybrid Model That Combines LSTM And Advanced NLP Transformers For Resume Summarization. The Model Generates Concise Summaries Of Resumes, Making It Easier For Recruiters To Quickly Assess Candidates' Qualifications.
|Lavanya P., Sasikala E.	Resume Recommendation Using RNN Classification And Cosine Similarity	2021	RNN And Cosine Similarity	This Paper Presents A Recommendation System That Uses Recurrent Neural Networks (RNN) For Classification And Cosine Similarity For Matching Resumes To Job Descriptions. The Approach Aims To Enhance The Accuracy Of Job Recommendations By Considering The Contextual Information In Resumes.
Pradeep Kumar Roy, Sunil Kumar Singh, Tapan Kumar Das, Asis Kumar Tripathy	Automated Resume Classification Using Machine Learning	2020	Deep CNN	The Authors Propose A Deep Convolutional Neural Network (CNN) Model For Automated Resume Classification. The Model Is Designed To Handle Large-Scale Resume Datasets And Improve The Efficiency Of The Recruitment Process By Accurately Classifying Resumes Into Predefined Categories. .

Table 2.1: Literature Review

CHAPTER 3
METHODOLOGY

This section outlines the systematic approach and procedures used to conduct research. It encompasses the strategies, techniques, and tools employed for data collection, analysis, and interpretation.

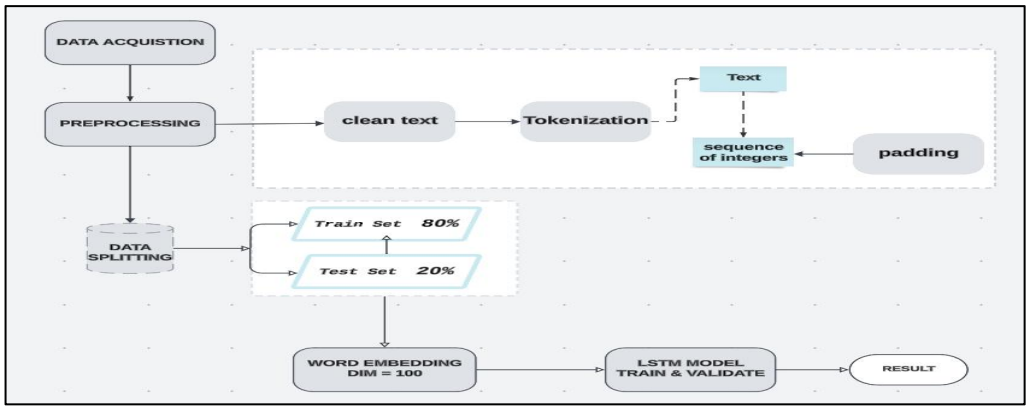

Fig 3.1 : Flow Chart Of Proposed Model

3.1 Data Collection

In this data collection effort focused on candidate eligibility based on resume classification using LSTM, a dataset comprising various resumes was gathered. Each resume included details such as degree, experience, technologies known, organizations worked for, and eligibility status. The classification criterion for resumes was based on their alignment with job requirements: those meeting the criteria were categorized as "Eligible," while those not meeting the criteria were deemed "Not Eligible." Upon analysis, it was found that the dataset contained a balanced mix of resumes with diverse backgrounds and skills. This dataset serves as a valuable resource for developing and testing resume classification models, offering insights into candidate qualifications and suitability for different roles within the tech industry.

A sample of the dataset is shown below using the head() function:"

	Degree	Experience	Technology	Organization	Eligible	Not Eligible
0	Bachelor of Computer Science in Artificial Int...	5	Python,R,Deep Learning,Machine Learning	Microsoft,Google	1	0
1	Bachelor of Computer Science in Artificial Int...	1	Python,R,Deep Learning,Machine Learning	Microsoft,Google	1	0
2	Bachelor of Computer Science in Data Science a...	1	Python,R,Data Analysis,Probability Theory	Google,IBM	1	0
3	Bachelor of Computer Science in Data Science a...	1	Python,R,Data Analysis,Probability Theory	Google,IBM	1	0
4	Bachelor of Computer Science in Data Science a...	1	Python,R,Data Analysis,Probability Theory	Google,IBM	1	0

Fig 3.2 : Sample Datatset

Visualization of eligible vs. not eligible

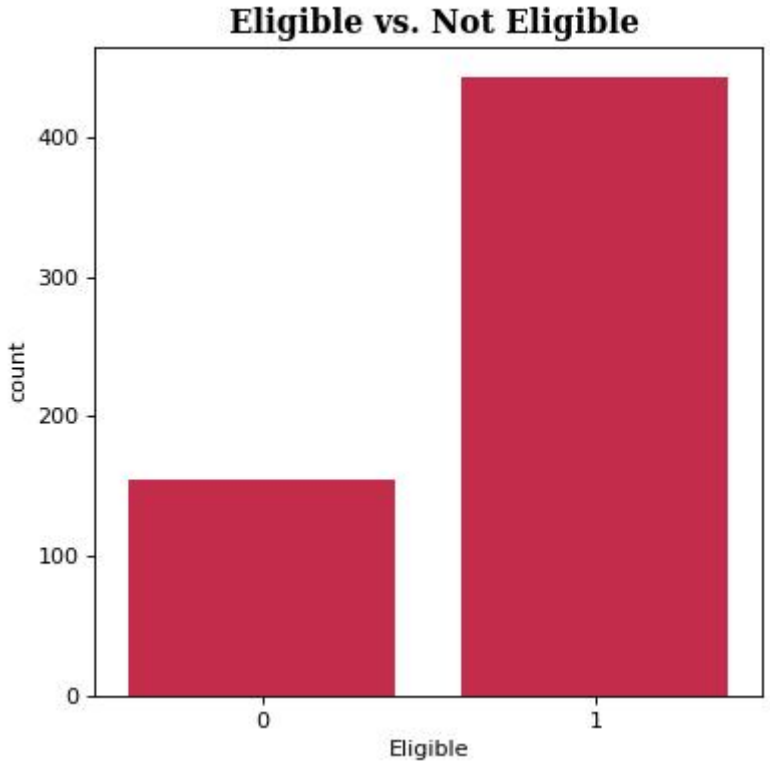

Fig 3.2.1 : Eligible Vs. Not Eligible

3.2 Data Cleaning

Data cleaning is a crucial step in the data analysis process, especially when dealing with large datasets. It involves identifying and rectifying errors, inconsistencies, and outliers in the data to improve its quality and reliability for subsequent analysis. Data cleaning likely included tasks such as:

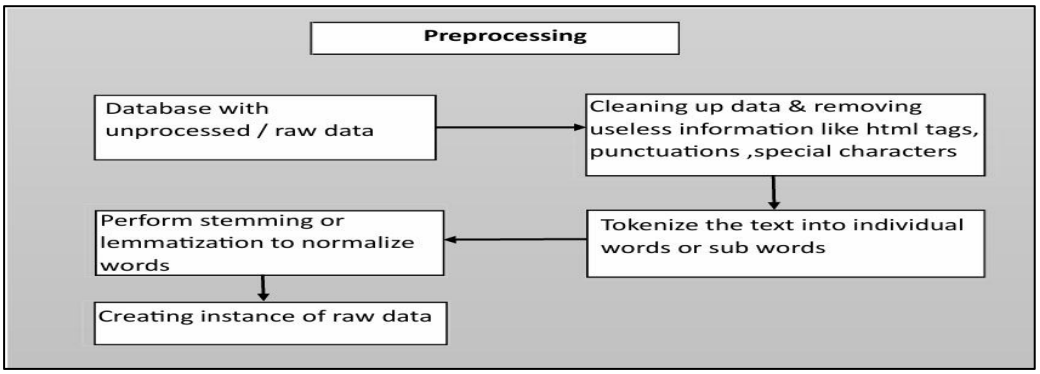

Fig 3.3 Data Cleaning and Preprocessing

3.2.1 Dealing With Duplicates

Identifying and removing duplicate reviews or instances of the same review to ensure each data point is unique and representative.

3.2.2 Handling Missing Data

Addressing any missing values in the reviews or scores, either by imputation or deletion based on the extent of missingness. The present dataset does not contain any missing values in reviews or score.

3.2.3 Cleaning Text

The ‘cleanResume’ function removes special characters, converts text to lowercase, and optionally removes stopwords (common words like "the", "and", etc.) using the nltk library

```
#pre-processing of data to remove special characters, hashtags, urls etc
import re
def cleanResume(resumeText):
    resumeText = re.sub('http\S+\s*', ' ', resumeText)  # remove URLs
    resumeText = re.sub('RT|cc', ' ', resumeText)  # remove RT and cc
    resumeText = re.sub('#\S+', '', resumeText)  # remove hashtags
    resumeText = re.sub('@\S+', '  ', resumeText)  # remove mentions
    resumeText = re.sub('[%s]' % re.escape("""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""), ' ', resumeText)  # remove punctuations
    resumeText = re.sub(r'[^\x00-\x7f]',r' ', resumeText)
    resumeText = re.sub('\s+', ' ', resumeText)  # remove extra whitespace
    return resumeText

data['cleaned_resume'] = data.Technology.apply(lambda x: cleanResume(x))
data
```

The `cleanResume` function is designed to clean resume text by removing unwanted elements such as URLs, retweet (RT) and cc tags, hashtags, mentions, punctuation, non-ASCII characters, and extra whitespace. This function is applied to the `Technology` column in the `data` DataFrame using a lambda function and the `apply` method. Each entry in the `Technology` column is processed by the `cleanResume` function to clean the text. The cleaned text is then stored in a new column called `cleaned_resume` in the `data` DataFrame. This systematic cleaning prepares the resume data for further analysis or model training.

	Degree	Experience	Technology	Organization	Eligible	Not Eligible	cleaned_resume
0	Bachelor of Computer Science in Artificial Int...	5	Python,R,Deep Learning,Machine Learning	Microsoft,Google	1	0	Python R Deep Learning Machine Learning
1	Bachelor of Computer Science in Artificial Int...	1	Python,R,Deep Learning,Machine Learning	Microsoft,Google	1	0	Python R Deep Learning Machine Learning
2	Bachelor of Computer Science in Data Science a...	1	Python,R,Data Analysis,Probability Theory	Google,IBM	1	0	Python R Data Analysis Probability Theory
3	Bachelor of Computer Science in Data Science a...	1	Python,R,Data Analysis,Probability Theory	Google,IBM	1	0	Python R Data Analysis Probability Theory
4	Bachelor of Computer Science in Data Science a...	1	Python,R,Data Analysis,Probability Theory	Google,IBM	1	0	Python R Data Analysis Probability Theory

Fig 3.4 : Cleaned Dataset

3.2.4 Tokenization

In this process, the `preprocess_text` function first tokenizes the text in the 'Technology' column of the DataFrame using NLTK's `word_tokenize`, converting the text to lowercase and removing common English stopwords. This cleans and normalizes the text by filtering out insignificant words. The preprocessed text is then applied to the 'Technology' column in the DataFrame. Subsequently, all preprocessed text from the 'Technology' column is combined into a single string. This combined text is used to generate a word cloud using the `WordCloud` class from the `wordcloud` library, with default stopwords to exclude common words from the visualization.

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords
import string
from wordcloud import WordCloud

oneSetOfStopWords = set(stopwords.words('english')+['``',"''"])
totalWords =[]
Sentences = data['cleaned_resume'].values
cleanedSentences = ""
for i in range(len(data)):
    cleanedText = cleanResume(Sentences[i])
    cleanedSentences += cleanedText
    requiredWords = nltk.word_tokenize(cleanedText)
    for word in requiredWords:
        if word not in oneSetOfStopWords and word not in string.punctuation:
            totalWords.append(word)

wordfreqdist = nltk.FreqDist(totalWords)
mostcommon = wordfreqdist.most_common(50)
print(mostcommon)
```

Finally, the word cloud is displayed using Matplotlib, providing a visual representation of the most frequently occurring words in the dataset.

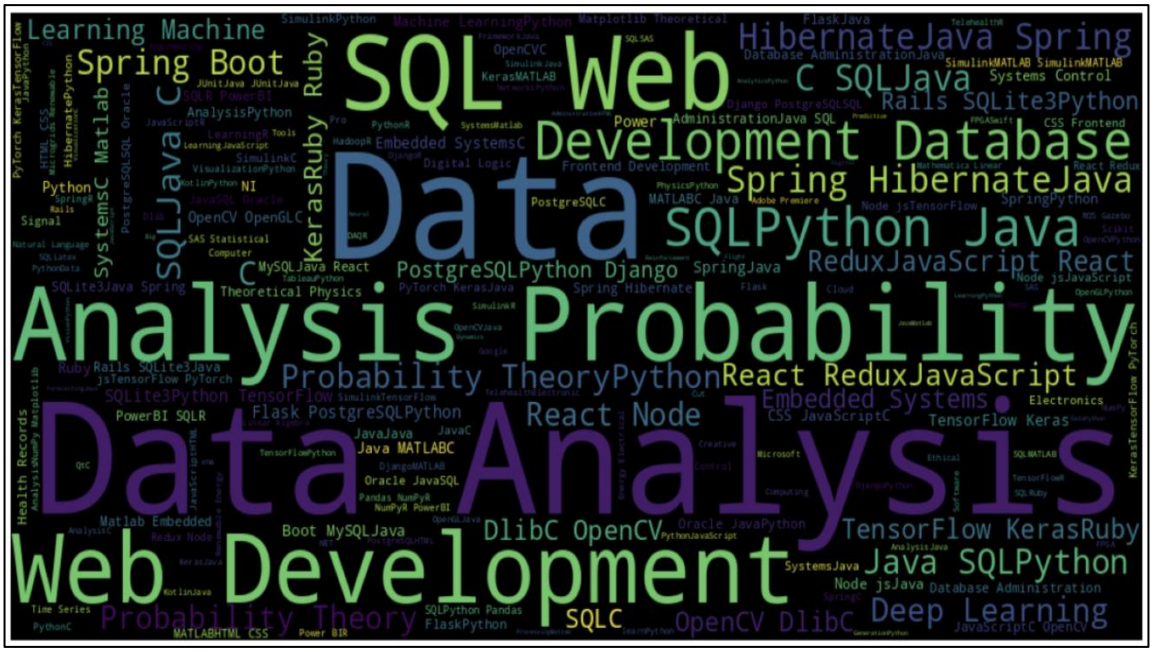

Fig 3.5 : Word Cloud After Tokenization

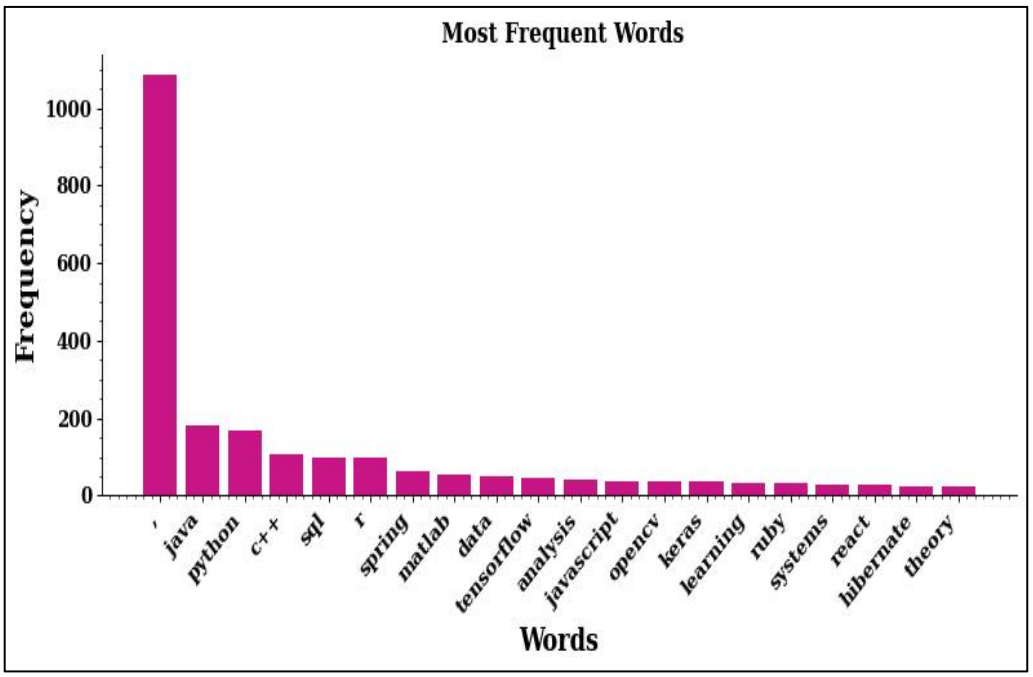

Fig 3.5.1 : Represenation Of Word Frequency

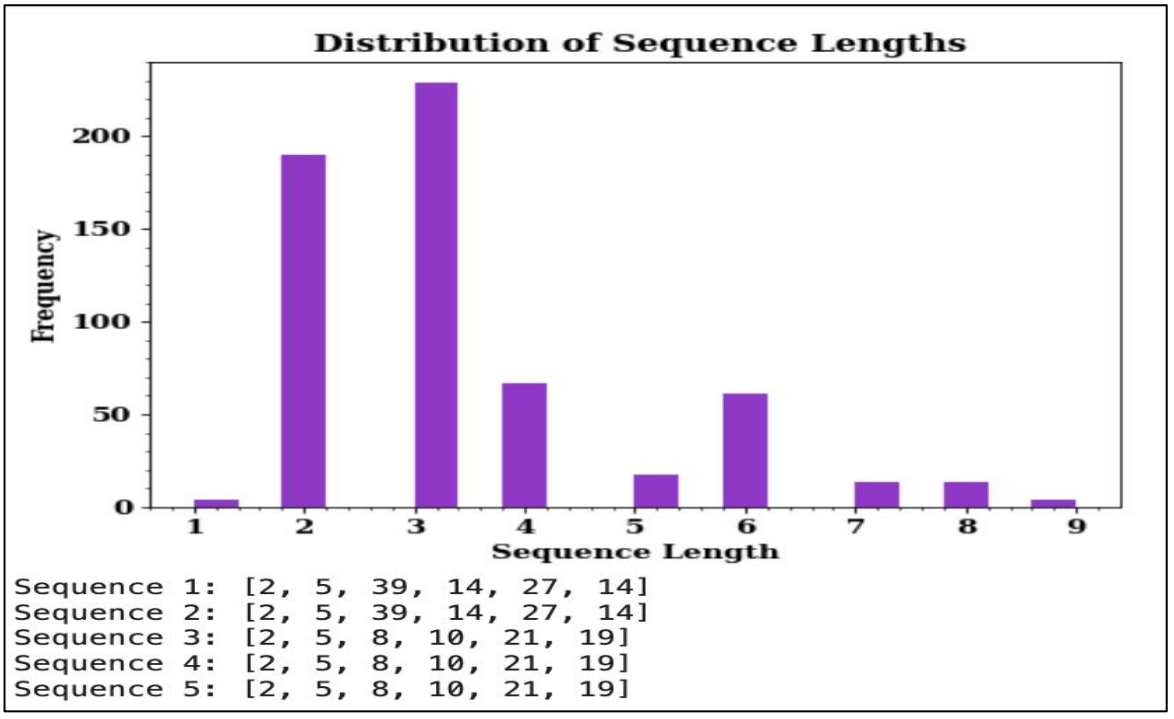

Fig 3.5.2 : Sequence Length Distribution

A bar chart to visualize the count of each unique value in the 'Technology' column of your DataFrame called data.

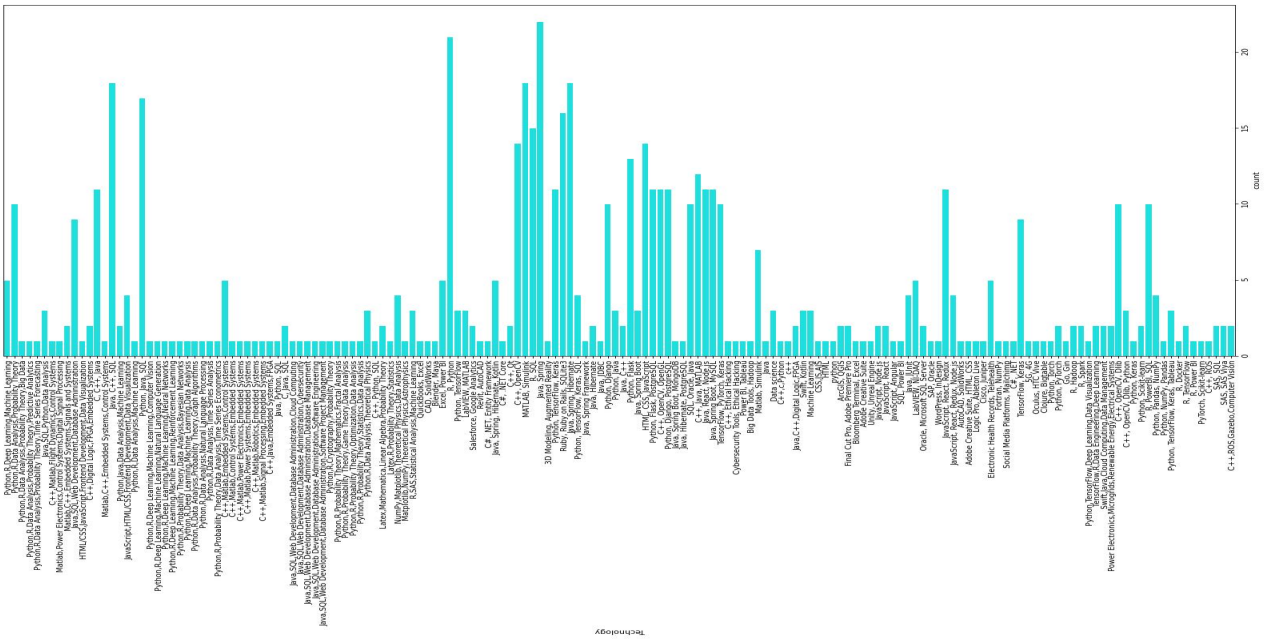

Fig 3.5.3 : Data Visualization of Technology Attribute

A data plot to visualize relationship between ‘Experience the likelihood of being ‘Eligible’ or ‘Not - Eligible’.

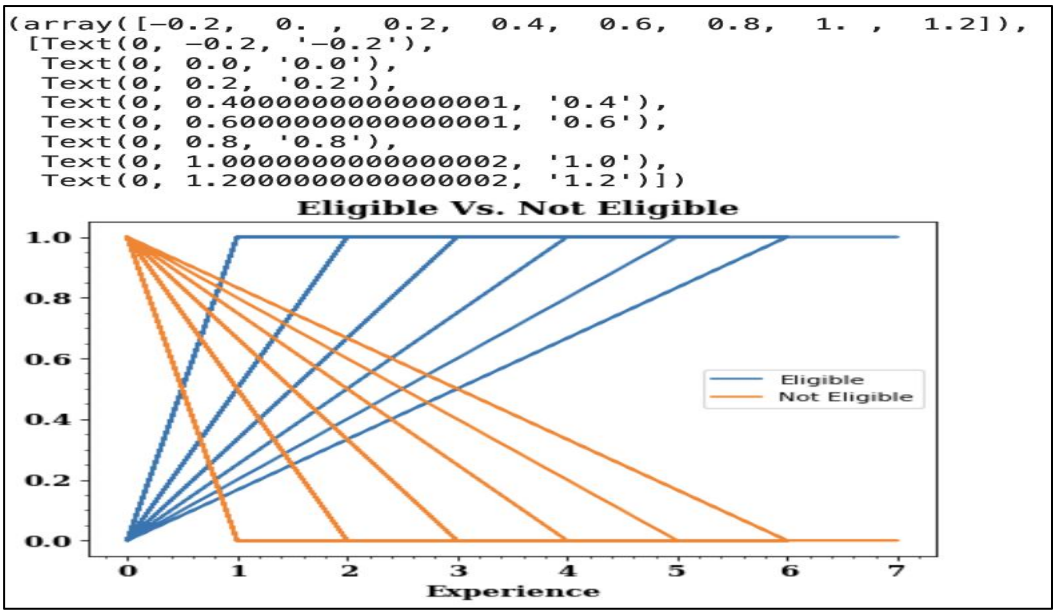

Fig 3.5.4 : Experience Vs. Eligible Vs. Not Eligible

3.2.5 Padding

The padding process involves adjusting the length of all sequences to a uniform size by adding zeros. In this case, the `pad_sequences` function pads each sequence with zeros up to a maximum length of 50 tokens, ensuring that all sequences in the dataset are of equal length for consistent input to the model.

3.3 Data Splitting

3.3.1 Shuffling

```
from sklearn.utils import shuffle

# Get features and labels from data and shuffle
features = data['cleaned_resume'].values
original_labels = data['Technology'].values
labels = original_labels[:]

for i in range(len(data)):
  labels[i] = str(labels[i].lower())  # convert to lowercase
  labels[i] = labels[i].replace(" ", "")  # use hyphens to convert multi-token labels into single tokens

features, labels = shuffle(features, labels)

# Print example feature and label
print(features[0])
print(labels[0])
```

```
LabVIEW MATLAB
labview,matlab
```

Fig 3.6 Shuffling of the dataset

Shuffling involves randomly reordering the dataset to ensure that the distribution of features and labels is mixed, which helps prevent bias in model training. In this process, the shuffle function from sklearn.utils is used to randomly shuffle the features and labels arrays in unison. This ensures that the relationship between features and their corresponding labels remains consistent while introducing randomness to the order of the data.

3.3.2 Train-Validation Split

Train-validation split divides the dataset into two subsets: one for training the model and one for evaluating its performance. Typically, a portion (e.g., 20%) of the data is reserved for validation to assess how well the model generalizes to unseen data. This split helps in tuning hyperparameters and preventing overfitting.

```
num_validation_samples = int(VALIDATION_SPLIT*data.shape[0])
x_train = data[:-num_validation_samples]
y_train = labels[:-num_validation_samples]
x_val = data[-num_validation_samples: ]
y_val = labels[-num_validation_samples: ]
```

3.3.3 Splitting Data to Training, and Testing Data

Splitting data into training and testing sets involves dividing the dataset into two subsets: one for

training the model and one for evaluating its performance. Typically, a large portion of the data (e.g., 80-90%) is used for training, while the remaining portion (e.g., 10-20%) is reserved for testing. This ensures that the model is trained on one set of data and tested on a separate, unseen set to gauge its generalization ability.

```
import pandas as pd
train = pd.read_csv('Train.csv')
train
```

	id	Technology	Experience	Eligible	Not_Eligible
0	1	Python,R,Deep Learning,Machine Learning	5	1	0
1	2	Python,R,Deep Learning,Machine Learning	1	1	0
2	3	Python,R,Data Analysis,Probability Theory	1	1	0
3	4	Python,R,Data Analysis,Probability Theory	1	1	0
4	5	Python,R,Data Analysis,Probability Theory	1	1	0

Fig 3.7 : After Splitting Dataset Into Training Data

```
import pandas as pd
test = pd.read_csv('Test.csv')
test
```

	id	Technology	Experience	Eligible	Not_Eligible
0	480	Python,TensorFlow,Deep Learning,Data Visualiza...	1	1	0
1	481	TensorFlow,R,Data Engineering,Deep Learning	5	1	0
2	482	TensorFlow,R,Data Engineering,Deep Learning	5	1	0
3	483	Java,C++,Digital Logic,FPGA	1	1	0
4	484	Swift,Java,Cloud Computing,Data Management	5	1	0

Fig 3.8 : After Splitting Dataset Into Testing Data

3.3.4 Tokenizing And Padding The Testing And Training Data

Tokenizing and padding the training and testing data involves converting text into numerical sequences and ensuring uniform length. First, the `Tokenizer` is fit on the training data to create a vocabulary, and then it transforms both training and testing text into sequences of integers. Next,

sequences are padded to a consistent length using `pad_sequences`, ensuring that all sequences are of the same size. This process standardizes the data, making it ready for model training and evaluation.

```
tokenizer = Tokenizer(num_words=MAX_NB_WORDS)
tokenizer.fit_on_texts(texts)

sequences = tokenizer.texts_to_sequences(texts)
word_index = tokenizer.word_index
print('Vocabulary size:', len(word_index))

Vocabulary size: 227

data = pad_sequences(sequences, padding = 'post', maxlen = MAX_SEQUENCE_LENGTH)
print('Shape of data tensor:', data.shape)
print('Shape of label tensor:', y.shape)

Shape of data tensor: (479, 200)
Shape of label tensor: (479, 3)
```

3.4 Word Embedding

3.4.1 Loading GloVe Embeddings

Loading GloVe embeddings involves reading pre-trained word vectors from a file to use as word representations in your model. The embeddings, with a dimensionality of 100 (EMBEDDING_DIM = 100), are loaded into a dictionary where each word maps to its corresponding vector.

```
EMBEDDING_DIM = 100
GLOVE_DIR = "/content/glove.6B." + str(EMBEDDING_DIM) + "d.txt"

embeddings_index = {}
f = open(GLOVE_DIR,'rb')
print('Loading GloVe from:', GLOVE_DIR,'...', end='')
```

This provides a rich, pre-trained representation of words, capturing semantic relationships, and is used to initialize the embedding layer in the neural network model.

3.4.2 Embedding Matrix:

An embedding matrix is created, mapping each word in the vocabulary to its corresponding GloVe embedding.

```
for line in f:
  values = line.split()
  word = values[0]
  embeddings_index[word] = np.asarray(values[1:], dtype='float32')
f.close()
print("Done.\n Proceeding with Embedding Matrix...", end="")
embedding_matrix = np.random.random((len(word_index) + 1, EMBEDDING_DIM))
for word, i in word_index.items():
  embedding_vector = embeddings_index.get(word)
  if embedding_vector is not None:
    embedding_matrix[i] = embedding_vector
print(" Completed!")
```

3.5 Model Architecture

- **Input Layer**

  This layer takes sequences of integers, each representing a tokenized word, with a fixed length of MAX_SEQUENCE_LENGTH. It provides the raw input data to the model, ensuring that all input sequences have the same size.

- **Embedding Layer**

  This layer uses pre-trained GloVe embeddings to convert word indices into dense vector representations. It is initialized with the embedding matrix and set as non-trainable to preserve the pre-trained weights, providing rich, semantic word representations.

- **LSTM Layer**

  The Long Short-Term Memory (LSTM) layer with 60 units processes the embedded sequences, capturing temporal dependencies in the data. By setting return_sequences=True, it outputs the entire sequence of hidden states for further processing.

- **Global Max Pooling Layer**

  This layer reduces the output from the LSTM layer by extracting the maximum value along each feature dimension. It summarizes the sequence information into a fixed-size vector, making it easier to manage and process in subsequent layers.

- **Dropout Layers**

  Two Dropout layers with a rate of 0.1 are added to prevent overfitting. Dropout randomly sets a fraction of input units to zero during training, helping the model generalize better by reducing dependency on specific neurons.

- **Dense Layers**

  A Dense layer with 50 units and ReLU activation functions as a fully connected layer, learning non-linear combinations of features. This layer contributes to the final classification or prediction by processing the output from previous layers.

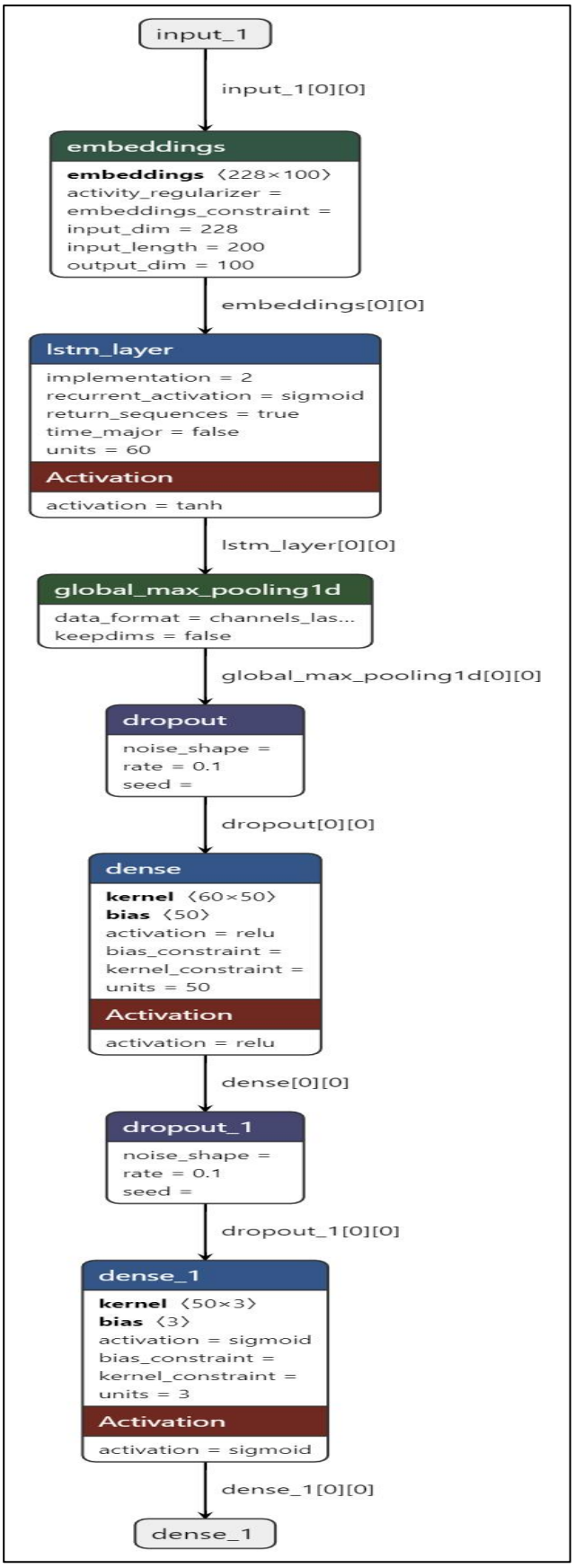

Fig 3.9 : Model Architecture Flow Chart

3.6 LSTM Model Fine-Tuning

In the fine-tuning process for LSTM models, the pre-trained weights are slightly adjusted to enhance performance on the new task. Fine-tuning typically necessitates a smaller learning rate and fewer epochs compared to training an LSTM model from scratch, as the model has already internalized significant sequential patterns and dependencies from its initial training on a large corpus. This approach capitalizes on the power of transfer learning, enabling LSTM models to achieve high performance on diverse tasks with relatively little task-specific data.

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 200)]             0

embeddings (Embedding)       (None, 200, 100)          22800

lstm_layer (LSTM)            (None, 200, 60)           38640

global_max_pooling1d (Glob   (None, 60)                0
alMaxPooling1D)

dropout (Dropout)            (None, 60)                0

dense (Dense)                (None, 50)                3050

dropout_1 (Dropout)          (None, 50)                0

dense_1 (Dense)              (None, 3)                 153

=================================================================
Total params: 64643 (252.51 KB)
Trainable params: 41843 (163.45 KB)
Non-trainable params: 22800 (89.06 KB)
_________________________________________________________________
```

Fig 3.10 LSTM Model Fine-Tuning

CHAPTER 4
RESULTS & DISCUSSION

4.1 Performance Measure

Performance measurement involves assessing how well the model generalizes to new, previously unseen examples and how effectively it accomplishes the task it was trained for. Evaluation provides insights into the model's strengths, weaknesses, and overall effectiveness, guiding further improvements or decisions. Various performance metrics such as accuracy, precision, recall, ROC curves, and F1 score are measured.

The ROC curve (Receiver Operating Characteristic curve) is a crucial evaluation tool, especially for binary classification tasks. It plots the true positive rate (recall) against the false positive rate at various threshold settings, providing a visual representation of the trade-offs between sensitivity and specificity. The area under the ROC curve (AUC) serves as an aggregate measure of the model's performance across all classification thresholds, with values closer to 1 indicating better performance. By analyzing the ROC curve, one can gain a deeper understanding of the model's discriminative ability and make more informed decisions about threshold selection and overall model effectiveness.

4.1.1 Train And Validation Loss

Plotting train and validation loss, this visual representation helps in understanding how well the model is learning and generalizing to new data.

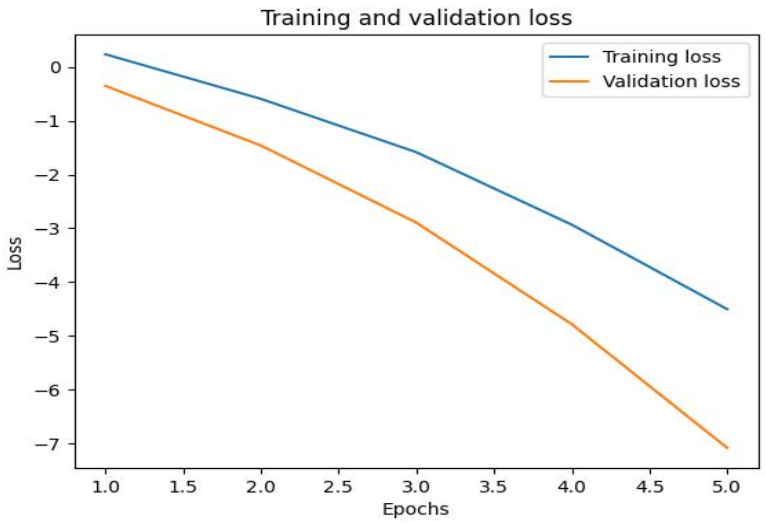

Fig 4.1.1 Train vs Validation Loss

4.1.2 Train And Validation Accuracy

Plotting train and validation accuracy, this visual representation helps in understanding how well the model is learning and generalizing to new data.

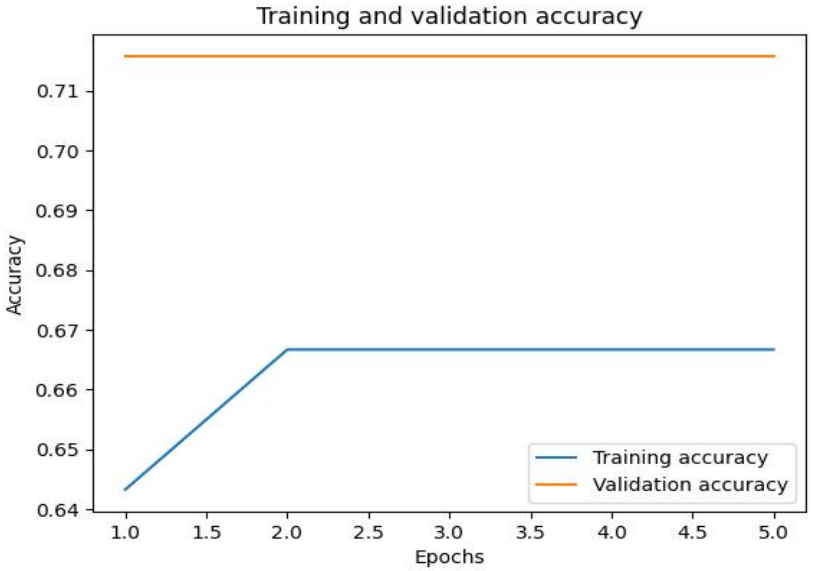

Fig 4.1.2 Train vs Validation Accuracy

4.1.3 Confusion Matrix

A confusion matrix provides a detailed breakdown of the model's predictions compared to the actual outcomes, allowing for a deeper understanding of its accuracy, precision, recall, and other performance metrics.

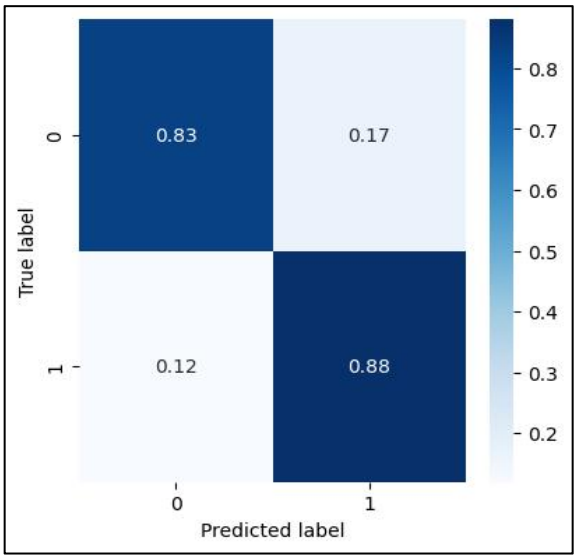

Fig 4.1.3 Confusion Matrix

4.1.4 ROC Curve

The ROC curve plots the true positive rate against the false positive rate to evaluate a binary classification model's performance. The area under the curve (AUC) measures the model's ability to distinguish between classes, with 1 being perfect and 0.5 indicating no discrimination.

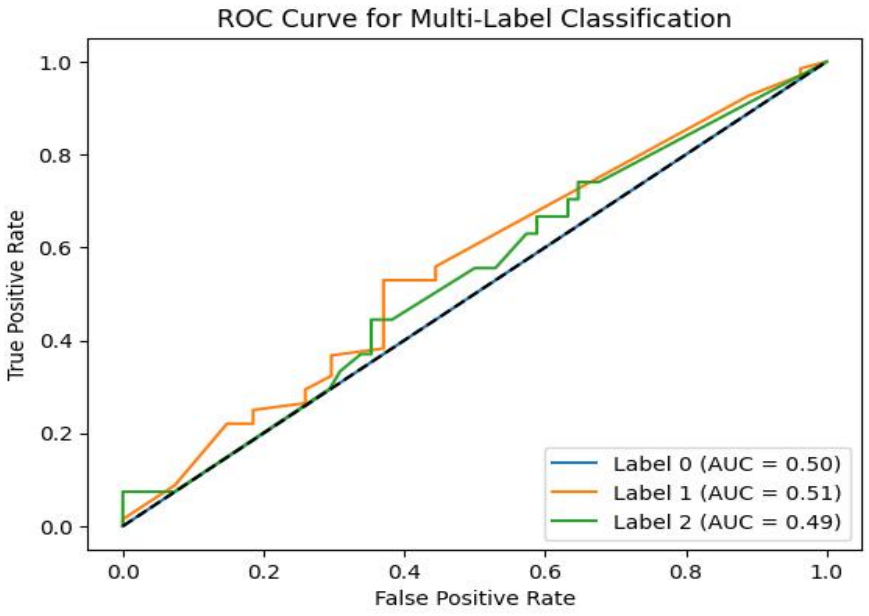

Fig 4.1.4 ROC Curves

CHAPTER 5
CONCLUSION & FUTURE SCOPE

5.1 Conclusion

This project aimed to develop a machine learning model capable of classifying candidates based on their technology skills, predicting their experience level and eligibility for a given role. By leveraging a Long Short-Term Memory (LSTM) network, the model learned to capture contextual information from technology descriptions, achieving promising results in predicting multiple labels simultaneously. The use of pre-trained GloVe embeddings enhanced the model's ability to understand semantic relationships between words, contributing to its overall performance. Evaluation metrics such as accuracy, ROC curves, and confusion matrices demonstrated the model's effectiveness in distinguishing between different classes. The model's ability to handle multi-label classification makes it a valuable tool for automating candidate screening and improving the efficiency of the recruitment process.

5.2 Future Scope

While the current model provides a solid foundation, there are several avenues for future exploration and improvement:

- **Hyperparameter Tuning**: Fine-tuning the model's hyperparameters, such as the number of LSTM units, dropout rates, and learning rate, could further enhance its performance.
- **Different Architectures**: Experimenting with alternative neural network architectures, such as bidirectional LSTMs or attention mechanisms, might lead to improved accuracy and better capture of long-range dependencies.
- **Enriching Input Features**: Incorporating additional features, such as candidate experience details or job requirements, could provide the model with more context and improve its predictive capabilities.
- **Explainability**: Investigating techniques for interpreting the model's predictions, such as attention visualization or feature importance analysis, would increase transparency and trust in the system.
- **Deployment and Integration**: Deploying the model as a web service or integrating it into existing recruitment platforms would enable real-world application and impact.

REFERENCES

[1]. Bera, S., Ghosh, B., Vanusha, D. (2022). Resume Classification Using Bidirectional LSTM and Attention Mechanism. In: Hu, YC., Tiwari, S., Trivedi, M.C., Mishra, K.K. (eds) Ambient Communications and Computer Systems. Lecture Notes in Networks and Systems, vol 356. Springer, Singapore. doi.org/10.1007/978-981-16-7952-0_22

[2]. Abstractive Text Summarization for Resumes With Cutting Edge NLP Transformers and LSTM, by Öykü Berfin Mercan, Sena Nur Cavsak, Aysu Deliahmetoglu (Intern), Senem Tanberk doi.org/10.48550/arXiv.2306.13315

[3]. Huseyinov, I., Diallo, I., Raed, M.W. (2023). Resume Recommendation using RNN Classification and Cosine Similarity. In: Kovalev, S., Kotenko, I., Sukhanov, A. (eds) Proceedings of the Seventh International Scientific Conference "Intelligent Information Technologies for Industry" (IITI'23). IITI 2023. Lecture Notes in Networks and Systems, vol 776. Springer, Cham. doi.org/10.1007/978-3-031-43789-2_9

[4]. Roy, P.K., Singh, S.K., Das, T.K., Tripathy, A.K. (2022). Automated Resume Classification Using Machine Learning. In: Rout, R.R., Ghosh, S.K., Jana, P.K., Tripathy, A.K., Sahoo, J.P., Li, KC. (eds) Advances in Distributed Computing and Machine Learning. Lecture Notes in Networks and Systems, vol 427. Springer, Singapore. doi.org/10.1007/978-981-19-1018-0_26

[5]. Jalili, Amirreza & Tabrizchi, Hamed & Razmara, Jafar & Mosavi, Amir. (2024). BiLSTM for Resume Classification. 000519-000524. 10.1109/SAMI60510.2024.10432836.

[6]. Competence-Level Prediction and Resume & Job Description Matching Using Context-Aware Transformer Models Changmao Li, Elaine Fisher, Rebecca Thomas, Steve Pittard, Vicki Hertzberg, Jinho D. Choi doi.org/10.48550/arXiv.2011.02998

[7]. S. Ramraj, V. Sivakumar and K. Ramnath G., "Real-Time Resume Classification System Using LinkedIn Profile Descriptions," 2020 International Conference on Computational Intelligence for Smart Power System and Sustainable Energy (CISPSSE), Keonjhar, India, 2020, pp. 1-4, doi: 10.1109/CISPSSE49931.2020.9212209.
keywords: {Domain Adaptation;CNN;SVM},

[8]. Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.

[9]. [PDF] Long short-term memory recurrent neural network architectures for large scale acoustic modeling , H Sak, AW Senior, F Beaufays - 2014 - isca-archive.org

[10]. [PDF] RESUME SCREENING USING LSTM , D Mule, S Doke, S Navale, SK Said - 2022 - academia.edu

[11]. Wang, J.H., Liu, T.W., Luo, X. and Wang, L., 2018, October. An LSTM approach to short text sentiment classification with word embeddings. In Proceedings of the 30th conference on computational linguistics and speech processing (ROCLING 2018) (pp. 214-223).

[12]. C. Li, G. Zhan and Z. Li, "News Text Classification Based on Improved Bi-LSTM-CNN," 2018

9th International Conference on Information Technology in Medicine and Education (ITME), Hangzhou, China, 2018, pp. 890-893, doi: 10.1109/ITME.2018.00199.
keywords: {Text categorization;Semantics;Training;Context modeling;Mathematical model;Support vector machines;text classification;Bi-LSTM-CNN;word order;text semantics},

[13]. X. Li, H. Shu, Y. Zhai and Z. Lin, "A Method for Resume Information Extraction Using BERT-BiLSTM-CRF," 2021 IEEE 21st International Conference on Communication Technology (ICCT), Tianjin, China, 2021, pp. 1437-1442, doi: 10.1109/ICCT52962.2021.9657937.
keywords: {Training;Resumes;Bit error rate;Neural networks;Manuals;Information retrieval;Feature extraction;resume parsing;information extraction;named entity recognition;BERT},