JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

RESUME SCREENING CLASSIFICATION

Dr.J.B. Jona, Arjun R, Ramkumar O N

Associate Professor, Student, Student
Department of Computer Applications,
Coimbatore Institute of Technology, Coimbatore, India

Abstract: This paper introduces a novel approach to streamline the recruitment process through the integration of Natural Language Processing (NLP) techniques with machine learning algorithms for resume screening. The objective is to automate the initial phase of candidate evaluation, ensuring a more efficient and unbiased selection process. By leveraging NLP, the system extracts meaningful features from textual data, enabling the development of robust machine learning models.

***Keywords-* Natural Language processing, Job Titles, Company Name, Resume, Certification, Language, Education.**

I. Introduction

1. Recruitment is a pivotal aspect of organizational success, and manual resume screening can be time-consuming and prone to biases. This paper addresses this challenge by proposing a system that harnesses the power of NLP to analyze and classify resumes. The integration of machine learning models aims to enhance the accuracy and penCV: Open Source Computer Vision
2. Keras: Keras is an open-source deep learning API written in Python, which runs on top of other popular deep learning frameworks, such as TensorFlow and Theano.
3. CNN: Convolutional Neural Network
   MNIST: Modified National Institute speed of candidate assessment, ultimately leading to more informed hiring decision.

II. Methodology

The methodology section outlines the steps involved in the resume screening process. Textual data preprocessing, including tokenization and stopwords removal, is employed. Feature extraction is performed using TF-IDF vectorization, and the dataset is divided into training and testing sets. Various machine learning models, including but not limited to logistic regression, decision trees, and support vector machines, are trained and evaluated.

Abbreviations and Acronyms

RS: Resume Screening
RC: Resume Classification
RSML: Resume Screening using Machine Learning
RSC: Resume Screening Classifier
NLP-RS: Natural Language Processing for Resume Screening
CVS: Curriculum Vitae Screening
CSA: Candidate Screening Application
RSCM: Resume Screening with Classification Models
RSIA: Resume Screening and Information Analysis
HRSC: Human Resources Screening Classifier

III. Implementation

1. Load and Inspect Data:

- Load the resume dataset.
- Explore the structure and content of the dataset.

2. Data Preprocessing:

- Perform any necessary data cleaning.
- Tokenize and clean the text data.
- Handle missing values if any.

3. Text Vectorization:

- Use techniques like TF-IDF (Term Frequency-Inverse Document Frequency) to convert text data into numerical vectors.
- Split the data into training and testing sets.

4. Label Encoding:

- Convert categorical labels (categories) into numerical format using Label Encoding.

5. Model Training:

- Choose a machine learning model for text classification (e.g., Multinomial Naive Bayes, Support Vector Machine, Neural Network).
- Train the model using the training set.

6. Model Evaluation:

- Evaluate the model's performance on the testing set using metrics like accuracy, precision, recall, and F1 score.

7. Hyperparameter Tuning (Optional):

- Fine-tune model hyperparameters to improve performance.

8. Prediction and Application:

- Use the trained model to predict the category of unseen resumes.
- Integrate the model into your application or workflow for automated resume screening.

IV. Training the data

Training the data model involves the following architecture

1. Data Loading and Exploration:

- Load the resume dataset.
- Explore the dataset to understand its structure and contents.

2. Data Preprocessing:

- Handle any missing data.
- Clean the text data by removing irrelevant information, special characters, and formatting issues.
- Tokenize the text into words or phrases.
- Label encode the target variable (categories).

3. Text Vectorization:

- Convert the text data into numerical vectors using techniques like TF-IDF or word embeddings.

4. Train-Test Split:

- Split the dataset into training and testing sets to evaluate the model's performance.

5. Choose a Model:

- Select a machine learning model suitable for text classification. Common choices include Naive Bayes, Support Vector Machines, or neural networks.

6. Model Training:

- Train the selected model using the training dataset.

7. Model Evaluation:

- Evaluate the model's performance on the testing set using metrics such as accuracy, precision, recall, and F1 score.

8. Hyperparameter Tuning (Optional):

- Fine-tune the model's hyperparameters to improve performance.
- Save the Model (Optional)
- Save the trained model for future use or deployment.

```
def build_model():
    # Define a CNN model for digit classification
    model = keras.Sequential([
        keras.Input(shape=(28, 28, 1)),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(9, activation="softmax")]
    )

    model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

    return model

def main(args):
    data_choice = args['data']
    batch_size = args['batch_size']
    epochs = args['epochs']
    model_save_fpath = args['model_save_fpath']
    exclude_fonts = args['exclude_fonts']

    # Load data depending on user choice
    x_train, x_val, x_test, y_train, y_val, y_test = prep_data.get_data(data_choice=data_choice,
                                                                        exclude=exclude_fonts)


    # Get a model instance
    model = build_model()
    # Train the model
    print("Starting training...")
    model.fit(x_train, y_train,
              validation_data=(x_val, y_val),
              batch_size=batch_size,
              epochs=epochs)
    print("Training complete")
```

V. Coding

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
import pandas as pd
import sns as sns
from matplotlib import pyplot as plt
import seaborn as sns
import re
import nltk
import pandas as pd
from matplotlib import pyplot as plt
from nltk.corpus import stopwords
import string
from wordcloud import WordCloud
warnings.filterwarnings('ignore')
from matplotlib.gridspec import GridSpec
from sklearn.naive_bayes import MultinomialNB
from sklearn.multiclass import OneVsRestClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score
from pandas.plotting import scatter_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
resumeDataSet = pd.read_csv('UpdatedResumeDataSet.csv' ,encoding='utf-8')
resumeDataSet['cleaned_resume'] = ''
resumeDataSet.head()
print ("Displaying the distinct categories of resume -")
print (resumeDataSet['Category'].unique())
resumeDataSet = pd.read_csv('UpdatedResumeDataSet.csv' ,encoding='utf-8')
resumeDataSet['cleaned_resume'] = ''
resumeDataSet.head()
plt.figure(figsize=(15,15))
plt.xticks(rotation=90)
sns.countplot(y="Category", data=resumeDataSet)
nltk.download('punkt')

# Assume that 'cleanResume' is a function defined in the 'cleanedresume' module
from cleanedresume import cleanResume

resumeDataSet = pd.read_csv('UpdatedResumeDataSet.csv', encoding='utf-8')
resumeDataSet['cleaned_resume'] = ''
resumeDataSet.head()

oneSetOfStopWords = set(stopwords.words('english') + ['``', "''"])
totalWords = []
Sentences = resumeDataSet['Resume'].values
cleanedSentences = ""
for i in range(0, 160):
    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)
wc = WordCloud().generate(cleanedSentences)
plt.figure(figsize=(15, 15))
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

1. Import Libraries:

Import necessary libraries, including pandas for data manipulation, TfidfVectorizer for text vectorization, train_test_split for splitting the data, MultinomialNB for the Naive Bayes classifier, and metrics for model evaluation.

2. Load Dataset:

Load the resume dataset from a CSV file using pandas.

3. Data Preprocessing and Text Vectorization:

Perform data preprocessing steps such as cleaning, tokenization, and label encoding as needed.
Use TfidfVectorizer to convert the text data into numerical vectors.

4. Train-Test Split:
Split the dataset into training and testing sets to assess the model's performance.

5. Model Training:
Create a Multinomial Naive Bayes classifier.
Train the classifier using the training data (X_train for features and y_train for labels).

6. Model Prediction:
Use the trained classifier to make predictions on the testing set (X_test).

7. Model Evaluation:
Evaluate the model's performance using metrics such as accuracy, precision, recall, and F1 score on the testing set.

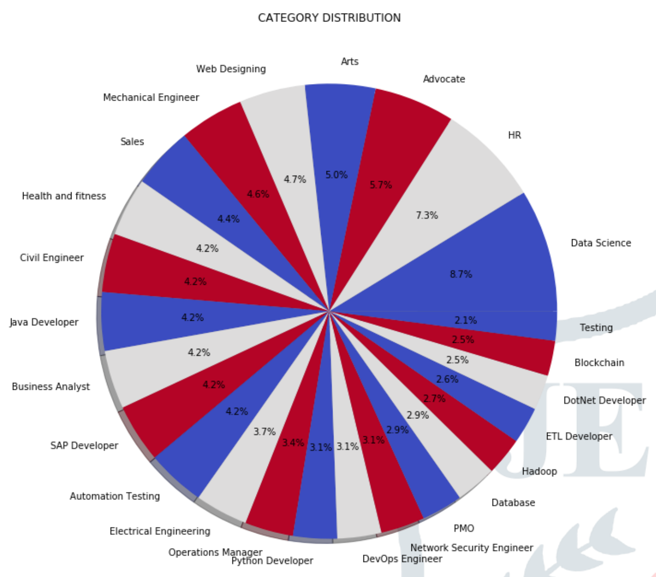
8. Save Model (Optional):
Optionally, save the trained model for later use or deployment.

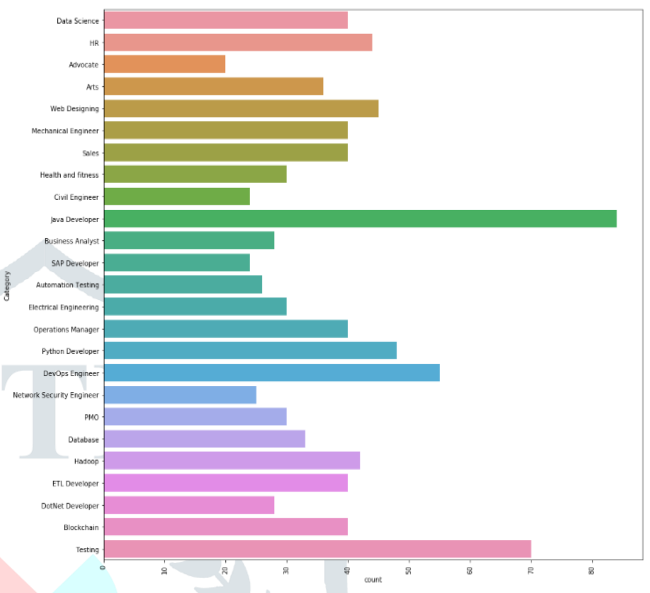
VI. Results

```
Displaying the distinct categories of resume -
['Data Science' 'HR' 'Advocate' 'Arts' 'Web Designing'
 'Mechanical Engineer' 'Sales' 'Health and fitness' 'Civil Engineer'
 'Java Developer' 'Business Analyst' 'SAP Developer' 'Automation Testing'
 'Electrical Engineering' 'Operations Manager' 'Python Developer'
 'DevOps Engineer' 'Network Security Engineer' 'PMO' 'Database' 'Hadoop'
 'ETL Developer' 'DotNet Developer' 'Blockchain' 'Testing']
Displaying the distinct categories of resume and the number of records belonging to each category -
Category
Java Developer               84
Testing                      70
DevOps Engineer              55
Python Developer             48
Web Designing                45
HR                           44
Hadoop                       42
Blockchain                   40
ETL Developer                40
Operations Manager           40
Data Science                 40
Sales                        40
Mechanical Engineer          40
Arts                         36
Database                     33
Electrical Engineering       30
Health and fitness           30
PMO                          30
Business Analyst             28
DotNet Developer             28
Automation Testing           26
Network Security Engineer    25
SAP Developer                24
Civil Engineer               24
Advocate                     20
Name: count, dtype: int64

Process finished with exit code 0
```

Education Details May 2013 to May 2017 B E UIT RGPV Data Scientist Data Scientist Matelabs Skill Details Python Exprience Less than 1 year months Statsmodels Exprience 12 months AWS Exprience Less than 1 year months Machine learning Exprience Less than 1 year months Sklearn Exprience Less than 1 year months Scipy Exprience Less than 1 year months Keras Exprience Less than 1 year monthsCompany Details company Matelabs description ML Platform for business professionals dummies and enthusiasts 60 A Koramangala 5th block Achievements Tasks behind sukh sagar Bengaluru India Developed and deployed auto preprocessing steps of machine learning mainly missing value treatment outlier detection encoding scaling feature selection and dimensionality

```
reduction Deployed automated classification and regression model linkedin com in aditya rathore b4600b146 Reasearch and deployed the time series forecasting model ARIMA SARIMAX Holt winter and Prophet Worked on meta feature extracting problem github com rathorology Implemented a state of the art research paper on outlier detection for mixed attributes company Matelabs description

Process finished with exit code 0

 [('Details', 484), ('Exprience', 446), ('months', 376), ('company', 330), ('description', 310), ('1', 290), ('year', 232), ('January', 216), ('Less', 204), ('Data', 200), ('data', 192), ('Skill', 166), ('Maharashtra', 166), ('6', 164), ('Python', 156), ('Science', 154), ('I', 146), ('Education', 142), ('College', 140), ('The', 126), ('project', 126), ('like', 126), ('Project', 124), ('Learning', 116), ('India', 114), ('Machine', 112), ('University', 112), ('Web', 106), ('using', 104), ('monthsCompany', 102), ('B', 98), ('C', 98), ('SQL', 96), ('time', 92), ('learning', 90), ('Mumbai', 90), ('Pune', 90), ('Arts', 90), ('A', 84), ('application', 84), ('Engineering', 78), ('24', 76), ('various', 76), ('Software', 76), ('Responsibilities', 76), ('Nagpur', 76), ('development', 74), ('Management', 74), ('projects', 74), ('Technologies', 72)]
```





VII. Acknowledgment

The authors express gratitude for any acknowledgments, recognizing contributions, support, or resources that facilitated the completion of this project.

References

[1] https://www.geeksforgeeks.org/nlpt-7/
References for backtracking algorithms

[2] https://www.coursera.org/projects/create-your-own-sudoku-solver-using-ai-and-python
On creating resume screening classification using deep learning techniques