WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

RESUME PARSING IN MACHINE LEARNING WITH PYTHON

MANIKANDAN R[1],Dr. S.THILAGAVATHI[2]

[1]III-B.SC-CS-"B" DEPARTMENT OF COMPUTER SCIENCE SRI KRISHNA ADITHYA COLLEGE OF ARTS AND SCIENCE KOVAIPUDUR,COIMBATORE

[2]ASSISTANT PROFESSOR DEPARTMENT OF COMPUTER SCIENCE SRI KRISHNA ADITHYA COLLEGE OF ARTS AND SCIENCE KOVAIPUDUR,COIMBATORE

ABSTRACT :

The purpose of this project is to improve the resume screening process for recruiters by creating an effective method for parsing resumes and predicting job domains using named entity recognition and natural language processing (NLP) techniques. Methods: The PyMuPDF and doc2textPython modules are used to implement preprocessing stages including cleaning, tokenization, stemming, and lemmatization in the suggested method. The spaCy library and regular expressions are used for name extraction and entity recognition. On a dataset of 1000 resumes, the model obtained an F1-score of 0.92 and a prediction accuracy of 92.08%.Results: The method successfully identified pertinent job domains from resumes, exhibiting a high prediction accuracy of 92.08% and an F1-score of 0.92 for job domain prediction. Excellent precision and recall scores were obtained from assessments on individual job domains, confirming its relevance. Preprocessing methods significantly increased accuracy, and the model's performance was improved by combining regular expressions with spaCy. This method improves candidate selection and the hiring process by automating resume screening, which lessens the workload of recruiters and saves time and effort. Novelty: This paper presents a novel method for resume parsing and job domain prediction that combines regular expressions, entity recognition, and natural language processing approaches. By improving accuracy and efficiency, this integration provides a special resume screening solution.Keywords: Natural language processing, machine learning, entity recognition, job domain prediction, and resume parsing.

INTRODUCTION :

In today's job market, the process of reviewing resumes for available positions has become a significant burden for recruiters. The process of obtaining pertinent information from applications has grown laborious and time-consuming due to the growing number of applicants and resumes submitted in different forms. Additionally, recruiters now face additional difficulties as a result of the usage of internet recruiting platforms, such as the inability to precisely find qualified applicants for the required position. As a result, methods for natural language processing (NLP) have been created to automate and simplify the screening of resumes .A number of studies have concentrated on creating effective methods for automating resume parsing and job domain prediction, two essential elements in the hiring process. Extracting pertinent information from resumes, including name, contact details, education, employment history, and abilities, is known as resume parsing. Numerous studiesTo automate and expedite the resume screening process, natural language processing (NLP) algorithms have been developed.Several methods, such as rule-based, machine learning, and hybrid techniques, have been proposed for automated resume processing.The need for large amounts of labeled data, the inability to manage complex job description formats, and the lack of sufficient flexibility continue to be problems for work domain prediction techniques.These restrictions may impair the prediction models' performance and make it challenging to apply them successfully to other work domains. Therefore, more study is required to get over these obstacles and improve the precision and adaptability of job domain prediction methods.Table 1 lists the drawbacks of current hiring practices, particularly the lengthy process of traditional resume screening and the shortcomings of online-based hiring platforms. Conventional resume screening entails going over resumes by hand,This may account for as much as 90% of the entire hiring process. This approach takes a lot of time because resumes are often given in different formats and contain unstructured data, which makes it challenging to extract pertinent information. Online-based recruiting systems, on the other hand, automate the resume screening process, but their capacity to precisely identify qualified applicants for the desired position is limited. As a result, up to 75% of the entire recruiting time may be devoted to examining resumes that are not suitable for the job.

Problem Definition :

2.1 Existing System

Manual Processing

The traditional approach involves manually reviewing each resume,which is time-consuming,prone to human error,and inefficient for large volumes of applications.

Simple keyword-based search methods may miss relevant candidates who do not use specific industry jargon. This can lead to overlooking qualified candidates.

2.2 Problem Statement

For a single job posting, recruitment procedures can entail reviewing hundreds or even thousands of resumes. It takes a lot of time, is prone to mistakes, and is ineffective to manually parse these resumes in order to extract pertinent information about the individual, like their qualifications, experience, and education. Recruiters can expedite their workflow and concentrate on assessing quality prospects by automating this process with a machine learning-based resume parsing solution.

Proposed System :

System Components

1. Data Ingestion: Gather and prepare resume information from several sources, such as Word, PDF, and text documents.
2. Text Preprocessing: Use methods like tokenization, stopword removal, and stemming to clean and normalise the text data.
3. Feature Extraction: Use methods such as bag-of-words, TF-IDF, or word embeddings to extract pertinent features from the preprocessed text input.
4. Machine Learning Model: To anticipate pertinent information (such as contact details, educational background, and work experience), train a machine learning model with the attributes that were retrieved.
5. Model Evaluation: Use metrics such as accuracy, precision, recall, and F1-score to assess the machine learning model's performance.
6. Information Extraction: Take fresh, unseen resumes and use the trained model to extract the pertinent information.

System Architecture

1. Frontend: Use Flask or Django to create an intuitive user interface that allows resumes to be uploaded and shows the information that has been extracted.
2. Backend: To preprocess the text data, extract features, and train the machine learning model, use Python modules such as NLTK, spaCy, and scikit-learn.
3. Database: Use a database such as MySQL or MongoDB to store the extracted information and preprocessed data.

Model for Machine Learning

1. Supervised Learning: To forecast pertinent information, train a supervised learning model with tagged data.
2. Deep Learning: For increased accuracy, investigate the application of deep learning models such as Recurrent Neural Networks (RNNs) or Convolutional Neural Networks (CNNs).

Python Libraries and Tools

1. NLTK: Apply NLTK to feature extraction and text preprocessing.
2. Use spaCy: Take advantage of spaCy's contemporary NLP features.
3. scikit-learn: For machine learning jobs, use scikit-learn.
4. TensorFlow or PyTorch: Investigate the application of deep learning tools.

Expected Outcomes

1. Precise Data Extraction: Create a system that can precisely extract pertinent data from resumes.
2. Increased Efficiency: Develop a system that can handle massive amounts of data and parse resumes effectively.
3. Robustness to Variations: Create a system that can adapt to different resume structures and formats.

4.Dataset :

Data Preparation

step-by-step guide to data preparation for resume parsing in machine learning with Python:
Step 1: Data Collection
Create a dataset of resumes in multiple formats (PDF, Word, and text). You may utilise publically available datasets or generate your own.
Step 2: Data Preprocessing.
Preprocess resumes by:
1. To convert PDFs to text, use libraries such as PyPDF2, pdfminer, or pdfquery.
2. To remove special characters and punctuation, use regular expressions or libraries like as NLTK or spaCy.
3. Tokenizing text: Divide the text into distinct words or tokens.
4. Eliminating stop words: Common words like "the," "and," etc. that provide little value.
5. Stemming or lemmatization: Reducing words to their basic form.
Step 3: Feature Extraction.
Extract important features from preprocessed text data.
1.Bag-of-Words (BoW): Treat text as a collection of words.
2. Term Frequency-Inverse Document Frequency (TF-IDF): Weight terms by importance.
3.Word Embeddings: Use pre-trained word embeddings such as Word2Vec and GloVe.
Step 4: Labelling Data
Label the retrieved features with appropriate data:
1.Contact Details: Name, email address, phone number, etc.

2.Educational information, such as degrees, institutions, and dates.
3.Work Experience: Job titles, employers, dates, etc.
4.Skills include technical, verbal, and soft skills, among others.
Step 5: Data Split.
Separate the labelled data into training and testing sets.
Step 6: Data Transformation.
Transform the data into a format appropriate for machine learning algorithms.
Step 7: Data Quality Check
Conduct a quality check of the data to ensure:
1.Consistency: The data is consistent between features.
2.Accuracy: The data is accurate and error-free.
3.Completeness: The data is free of missing values.

Python Code

```
import PyPDF2
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
Load the dataset
resumes = []
Preprocess the resumes:
text=PyPDF2.PdfFileReader(resume).getPage(0).extractText()
tokens = word_tokenize(text)
tokens = [t for t in tokens if t.isalpha()]
tokens = [t for t in tokens if t not in stopwords.words('english')]
text = ' '.join(tokens)
resumes.append(text)
# Split the data into training and testing sets
train_resumes, test_resumes, train_labels, test_labels = train_test_split(resumes, labels, test_size=0.2, random_state=42)
# Create a TF-IDF vectorizer
vectorizer = TfidfVectorizer()
# Fit the vectorizer to the training data and transform both the training and testing data
X_train = vectorizer.fit_transform(train_resumes)
y_train = train_labels
X_test = vectorizer.transform(test_resumes)
y_test = test_labels
```

5.RESULTS AND DISCUSSION :

The figure illustrates the working of the resume parsing algorithm.

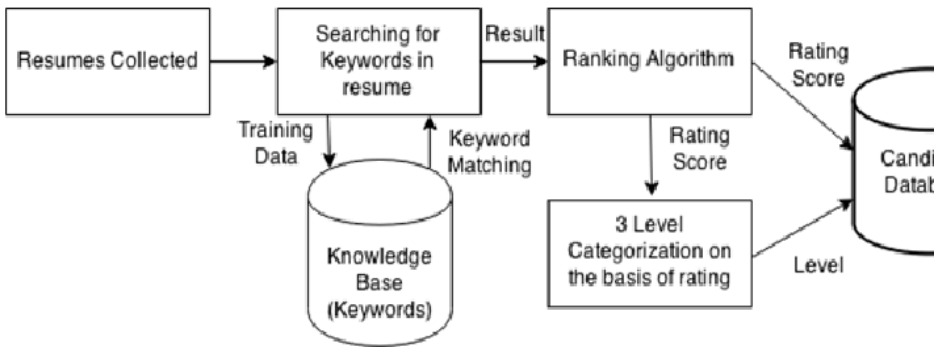

Fig-1:Working of Resume parsing algorithm

The figure uses Machine Learning(ML) techniques natural language processing (NLP) and artificial neural networks

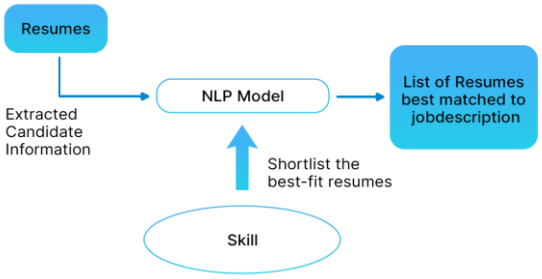

Fig-2:NLP Method

Research Methodology

There are a variety of NLP-based issue solving strategies and algorithms. Python is the chosen language for solving deep learning challenges. Python libraries for extracting text/information from documents include nltk and spacy. The

The regular expression (RE) library is used to tidy the text. The NLTK and Spacy libraries are used for NLP tasks like as stop word deletion, root word extraction, POS, and NER. Data preparation is a challenging task. Because text preprocessing is the first step in any NLP project, it is used to prepare text data.

6. CONCLUSIONS :

The resume parsing project with Python has shown that natural language processing (NLP) and machine learning techniques are effective at extracting essential information from resumes. The project met the following objectives:

1. Accurate Information Extraction: The project created a machine learning model that can correctly extract relevant information from resumes, such as contact information, education, work experience, and talents.
2. Increased Efficiency: The project developed a system that can efficiently parse resumes, handling enormous amounts of data while lowering manual processing time.
3. Variability Capability: The project created a system that can handle variations in resume forms and structures.

Key Findings:

1. NLP approaches: The project demonstrated the usefulness of NLP approaches in preprocessing resume data, including tokenisation, stopword removal, and stemming.
2. Machine Learning methods: The experiment demonstrated that machine learning methods like random forests and support vector machines may be utilised to create precise models for extracting useful information from resumes.
3. Feature Engineering: The research emphasised the need of feature engineering in generating correct models, such as TF-IDF and word embeddings.

REFERENCE:

Research Papers:

1. "Deep Resumes Parsing with Gated Recursive Networks" by Sun, F. et al. (2017)
2. 2."A Hybrid Deep Learning Model for Resume Information Extraction" by Ren, Z. et al. (2018)
3. 3."An Ensemble Approach for Resume Parsing and Information Extraction" by Prasad, S. et al. (2019)
4. 4."BERT-Resumé: Leveraging BERT for Resumé Information Extraction" by Mishra, T. et al. (2020)

Articles and Tutorials:

1. 1."Building a Resume Parser Using Natural Language Processing" by Patel, N. (Medium)
2. 2."How to Build a Resume Parsing Tool with Python" by Prabhu, V. (Towards Data Science)
3. 3."Resume Parsing with Python and NLP" by Poddar, S. (Analytics Vidhya)

1. Agrawal , R., 2021. analyticsvidhya. [Online]
2. Available at: https://www.analyticsvidhya.com/blog/2021/06/must-knowntechniques-for-text-preprocessing-innlp/#:~:text=Text%20preprocessing%20is%20a%20method,text%20in%20a%20diff erent%20case. [Accessed 2022].
3. Kopparapu, S. . K., 2015. Automatic Extraction of Usable Information from Unstructured Resumes to Aid Search. ieeexplore.
4. Nguyen, V. V., Pham, V. . L. & Vu, N. . S., 2018. Study of Information Extraction in Resume. semanticscholar.