



The Islamia UNIVERSITY of Bahawalpur

Faculty of Computing

Bahawalpur Main CAMPUS

Semester Project Report

On

Resume Parser with Natural Language Processing

Submitted By:

Hamna Qaseem (F20BDATS1E01012)

Manahil Ahmad (F20BDATS1E01024)

Fiza Asghar (F20BDATS1E01008)

Samreen Jamil (F20BDATS1E01016)

Submitted To:

Supervisor: Dr. Muhammad Zeeshan Jhandir

Department of Data Science

Bahawalpur, Pakistan

May, 2023

DECLARATION

We hereby declare that the report of the project entitled “Resume Parser with Natural Language Processing” which is being submitted to the Department of Data Science, Islamia university of Bahawalpur, in the fulfillment of the requirements that are required for our semester project. This report has been prepared on the basis of our own work, and where we used helping sources these have been acknowledged.

Hamna Qaseem (f20bdats1e01012)

Manahil Ahmad (f20bdats1e01024)

Fiza Asghar (f20bdats1e01008)

Samreen Jamil (f20bdats1e01016)

DATE: May, 2023

CERTIFICATE OF APPROVAL

The undersigned certify that they have read, and recommended to the Department of Data Science for acceptance, a semester project report entitled **'Resume Parser with Natural Language Processg'** submitted by **Hamna Qaseem, Manahil Ahmad, Fiza Asghar, Samreen Jamil** in fulfillment of the requirements of semester project.

Supervisor: Dr. Muhammad Zeeshan Jhandir

Department of Data Science

Faculty of Computing, The Islamia Univeristy of Bahawalpur

Head of Department: **Dr. Muhammad Akmal Khan**

Department of Data Science

Faculty of Computing, The Islamia Univeristy of Bahawalpur

DATE OF APPROVAL: March, 2023

ACKNOWLEDGEMENT

This project is undertaken as part of our academic curriculum for the Bachelor's degree in Data science. We express our sincere appreciation to the **Department of Data Science (faculty of computing)**, for granting us the opportunity to work on this project.

We would like to extend our gratitude to our supervisor, **Dr. Muhammad Zeeshan Jhandir**, for their invaluable guidance throughout this project. Their expertise and support have played a crucial role in shaping our understanding of the subject matter. We are grateful for their continuous encouragement and insightful feedback, which have contributed significantly to our learning experience.

Developing a **Resume Parser with Natural Language Processing** has given us the opportunity to delve into the field of text analysis and gain hands-on experience with NLP techniques.

We are confident that the skill and knowledge required through this project will prove beneficial in our future academic and professional endeavors.

Hamna Qaseem (f20bdats1e01012)

Manahil Ahmad (f20bdats1e01024)

Fiza Asghar (f20bdats1e01008)

Samreen Jamil (f20bdats1e01016)

ABSTRACT

Our project is aiming to parse the information from PDF-formatted resumes using the power of Natural Language Processing. We collected data from various sources including Kaggle and Github. The data is in the form of PDF-files which are processed to extract the resume texts and their corresponding file path. We pre-process and clean the data and then perform exploratory data analysis to gain insights from the dataset. We conduct topic modeling and categorize resume according to their topics. This helps in identifying meaningful topics present in the resumes. We select appropriate number of topics and merge them for further analysis.

For classification, we experiment with different models and evaluate their performance. The result of the project show better performance. This demonstrates the effectiveness of deep learning algorithms in improving the accuracy of resume parsing as compared to traditional machine learning algorithms.

In conclusion, Resume Parser project successfully demonstrates the application of NLP techniques for resume information extraction.

Keywords: ResumeParser, NLP, PDF-files, TopicModeling, MachineLearning, DeepLearning, DataManipulation

TABLE OF CONTENTS

DECLARATION	i
CERTIFICATE OF APPROVAL	ii
ACKNOWLEDGEMENT	iii
ABSTRACT	iv
1 INTRODUCTION	1
1.1 Background	1
1.2 Motivation	2
1.3 Objectives	2
1.4 Problem statement	2
1.5 Scope of Project	3
2 LITERATURE REVIEW	4
3 REQUIREMENT ANALYSIS	5
4 SYSTEM ARCHITECTURE AND METHODOLOGY	7
4.1 Data Collection	7
4.2 Topic Modeling and Domain stopwords	8
5 IMPLEMENTATION DETAILS	9
6 Methodolgy	11
6.1 Comparing our results with the paper we followed:	12
6.2 "Deep Learning Algorithm" - BERT	13
7 RESULTS AND DISCUSSIONS	15
8 Future work	16
REFERENCES	17

1. INTRODUCTION

Everyday, corporation and recruiting organizations must process a vast number of resumes. Dealing with a lot of text data is typically time consuming and stressful. In today's digital age, recruitment processes have become more automated, and employers receive hundreds of resumes for a single job opening. It can be a challenging task for hiring managers to manually review each resume and select the most suitable candidates. To overcome this challenge, many companies have started resume parsers, which are programs that can automatically extract relevant information from a resume and store it in a structured format. The information collected from various resumes can be presented in a variety of formats such as .pdfs, docx and others. Parsed information includes names, experience, keywords and clusters of resumes (e.g. computer science, human resource, etc) . [1]

Thus it is simpler to examine, analyze and interpret a resume if it is converted into prepared language or structured information. As a result a lot of businesses and institutions rely on information extraction which involves taking crucial information from unstructured data and converting it into more comprehensible and organized data formats. By parsing information, recruiters and hiring managers can quickly and efficiently screen candidates and identify those who may be a good fit for a particular job.

After performing all the initial steps, next for improving the accuracy of NLP-based resume parser we start with traditional machine learning algorithms and then moved to deep learning algorithms such as BERT, RNNs, CNN these algorithms can learn from data and extract more complex features to improve the accuracy.

1.1. Background

The recruitment process is a crucial aspect of any organization's success. However, the traditional recruitment process is time-consuming and tedious, requiring hiring managers to sift through numerous resumes to find the most qualified candidates. As a result, many companies have turned to technology to streamline the recruitment process. One such technology is the Resume Parser, which can automatically extract relevant information from a resume and store it in a structured format. This technology has gained popularity in recent years due to its ability to reduce the time and effort required in the recruitment process.

1.2. Motivation

The motivation behind developing a Resume Parser is to make the recruitment process more efficient and effective. With the increasing number of applicants for each job opening, it is becoming more challenging for hiring managers to review each resume manually. A Resume Parser can help in automatically extracting relevant information, such as work experience, education, and skills, from a resume, and store it in a structured format. This can help in quickly identifying the most qualified candidates and reducing the time and effort required in the recruitment process. The motivation behind the project stems from the limitations of traditional machine learning algorithms in resume parsing. It discusses the need for advanced techniques, such as deep learning models, and highlights the potential of BERT (Bidirectional Encoder Representations from Transformers) in achieving improved performance.

1.3. Objectives

The objective of our project is to extract relevant information from resumes and present it in structured format that can be easily integrated into database.

1. The objective of this project is to develop a Resume Parser using Python programming language. The specific objectives include data acquisition, exploratory data analysis, text pre-processing, topic modeling, classification, and future enhancements such as database integration and the creation of a streamlit app.
2. The project aims to provide an efficient and effective solution for automating the recruitment process.

1.4. Problem statement

The traditional recruitment process requires hiring managers to manually review each resume, which can be a time-consuming and tedious task, especially when hundreds of resumes are received for a single job opening. This can lead to delayed hiring decisions and the possibility of missing out on highly qualified candidates. People with diverse personalities come from a variety of fields and backgrounds. In the same way, their resume writing style varies. They've worked on a variety of projects, and each of them has unique writing style. As a result, each resume is unique. Some people work in the human resources department. They will have to go through hundreds of resumes on the internet. Executives summarize the resume, enter specific information into their database, and then call the applicant for job

counselling after obtaining the resume. An executive spent around 10-15 minutes on each resume, summarizing it and entering the information into the database. This project will help in the automation of this procedure. [2]

1.5. Scope of Project

The scope of this project includes developing a Resume Parser using Python programming language that can extract relevant information from a resume and store it in a structured format. The Resume Parser will be able to extract information such as work experience, education, and skills. The project will use Natural Language Processing (NLP) techniques to analyze the text in the resume and extract relevant information.

A resume parser can automatically take out relevant information from a resume, such as a candidate's name, education, work experience, and skills. The scope of the resume parser would typically include:

- Data manipulation and cleaning: Pandas and Numpy, Matplotlib Data Storing: Sqlite3 and pickle
- Text precossing: NLTK, spaCy, gensim, scikit-learn , pyPDF2
- Topic Modeling: NMF (Non-negative Matrix factorization), LDA (Latent Derilicht Analysis)
- Visualization: matplotlib, seaborn

2. LITERATURE REVIEW

Resume parsing using Natural Language Processing(NLP) techniques has emerged as a powerful tool for automating the extraction of relevant information from resumes. Prior to this advancement of NLP based resume parsers, resumes were manually reviewed by HR, which were tiring and time consuming. Resume parsing techniques has gained significant attention in recent years. Different research papers have explored various approaches and methodology to extract relevant information from resumes.

Following are few of research papers that we reviewed and gain insight from their work and study:

- The ongoing approaches for resume parsing mainly use regular expressions, chunking, keyword based models and entity recognition models. Relevant to this context, this paper proposes a system for resume parsing using deep learning models such as the convolutional neural network (CNN), Bi-LSTM (Bidirectional Long Short-Term Memory) and Conditional Random Field (CRF). [3]
- 'Resume Parser with Natural Language Processing', Parse information from a resume using natural language processing, find the keywords, cluster them onto sectors based on their keywords and lastly show the most relevant resume to the employer based on keyword matching.[1]
- This paper aims to solve these issues by automatically suggesting the most appropriate candidates according to the given job description. Our system uses Natural Language Processing to extract relevant information like skills, education, experience, etc. from the unstructured resumes and hence creates a summarised form of each application. [4]

3. REQUIREMENT ANALYSIS

Requirements analysis for the Resume Parser project involves identifying and documenting the functional and non-functional requirements of the system. Requirements analysis includes functional and non functional requirements.

Functional Recuirements:

Everyone wants to do jobs as quickly as possible in today's environment, but how quickly they do so depends entirely on the strategies they employ. There might come a time when we have to write a lot of lines of code; if we do this all by ourselves, it will take a while. A library is a collection of related modules. It offers code bundles that can be applied to other programs. In Python, modules also have a function. Instead of using the same code repeatedly across several programs and complicating the code, we can define frequently used functions as modules and import them wherever they are needed. [2]. Some of the most significant libraries and modules that we used are listed below:

1. Pandas

Pandas is a Python library that is open source. It's a tool for analyzing data.

2. OS

Python, the OS module has methods for creating and deleting folders, retrieving their contents, altering and identifying the current directory, and so on.

4. Utils

Python Utils is a collection of simple Python methods and classes that simplify and shorten common patterns.

5. Spacy

aCy is an open-source package library for advanced natural language processing implemented in Python. SpaCy has pre-trained pipelines with tokenization support.

6. Warnings

Warning messages are shown using the warn () method from the 'warning' module.

7. Nltk.corpus

PDF SCANNING

```
In [1]: !pip install PyPDF2

Requirement already satisfied: PyPDF2 in c:\users\hamna\anaconda3\lib\site-packages (3.0.1)
Requirement already satisfied: typing_extensions>=3.10.0.0 in c:\users\hamna\anaconda3\lib\site-packages (from PyPDF2) (4.3.0)

In [2]: import PyPDF2
import os
from os import listdir
from os.path import isfile, join
from io import StringIO

In [3]: !pip install spacy

Requirement already satisfied: spacy in c:\users\hamna\anaconda3\lib\site-packages (3.5.1)
Requirement already satisfied: packaging>=20.0 in c:\users\hamna\anaconda3\lib\site-packages (from spacy) (21.3)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in c:\users\hamna\anaconda3\lib\site-packages (from spacy) (1.0.4)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in c:\users\hamna\anaconda3\lib\site-packages (from spacy) (3.3.0)
Requirement already satisfied: Jinja2 in c:\users\hamna\anaconda3\lib\site-packages (from spacy) (3.1.2)
Requirement already satisfied: thinc<8.2.0,>=8.1.8 in c:\users\hamna\anaconda3\lib\site-packages (from spacy) (8.1.9)
```

Figure 3.1: python libraries used while scanning pdfs

The modules in this package provide functions for reading corpus files in a variety of formats. A corpus is simply a set of texts that are used as input.

8. Re

A RegEx, also known as a Regular Expression, is a string of characters that defines a search pattern. This module's functions allow to see if a given string matches a given regular expression.

9. NLTK

NLTK is a Python toolbox for working with natural language processing. It gives us access to a number of text processing libraries as well as a large number of test datasets.

10. Pdfminer

PDFMiner is a tool that extracts text from PDF files. It focuses completely on gathering and processing text data, unlike other PDF-related tools.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

The Resume Parser system follows a modular architecture that encompasses various components working together to extract and analyze resume information. The key components of the system architecture are as follows:

4.1. Data Collection

The resume datasets were collected from different sources such as kaggle [5] and Github [6]. The dataset is in Pdfs file format. The number of resume instance for each class job category. After gaining access to the dataset, the resume PDF files are downloaded from the available sources. The number of files and the size of the dataset may vary depending on the sources. It is important to ensure that the downloaded files are representative of the target population and cover a diverse range of industries, job roles, and experience levels.

Data Preprocessing

Natural Language Processing is a subfield of data science that works with textual data. When it comes to handling the Human language, textual data is one of the most unstructured types of data available. NLP is a technique that operates behind the it, allowing for extensive text preparation prior to any output. Before using the data for analysis in any Machine Learning work, it's critical to analyze the data. To deal with NLP-based problems, a variety of libraries and algorithms are employed. For text cleaning, a regular expression(re) is the most often used library. The next libraries are NLTK (Natural language toolkit) and spacy, which are used to execute natural language tasks like eliminating stopwords. Preprocessing data is a difficult task. Text preprocessing is done in order to prepare the text data for model creation. It is the initial stage of any NLP project [1]. The following are some of the preprocessing steps:

- Removing Stop words
- Lower casing
- Tokenization
- Lemmatization

4.2. Topic Modeling and Domain stopwords

Removing Stop words:

To eliminate noise from data, data cleaning is essential in NLP. Stop words are the most frequently repeated words in a text that give no useful information. The NLTK library includes a list of terms that are considered stop words in English. [I, no, nor, me, mine, myself, some, such we, our, you'd, your, he, ours, ourselves, yours, yourself, yourselves, you, you're, you've, you'll, most, other] are only a few of them. The NLTK library is a popular library for removing stop words, and it eliminates about 180 stop words. For certain difficulties, we can develop a customized set of stop words. Using the add technique, we can easily add any new word to a collection of terms.

Lemmatization:

The process of reducing inflected forms of a word while verifying that the reduced form matches to the language is known as lemmatization. A lemma is a simplified version or base word. Lemmatization uses a pre-defined dictionary to save word context and verify the word in the dictionary as it decreases. Organizes, organized, and organizing, for example, are all forms of organize. The lemma in this case is organize. The inflection of a word can be used to communicate grammatical categories such as tense (organized vs organize). Lemmatization is required since it aids in the reduction of a word's inflected forms into a particular element for analysis. It can also assist in text normalization and the avoidance of duplicate words with similar meanings [2].

Tokenization:

The initial stage in text analysis is tokenization. It enables to determine the text's core components. Tokens are the fundamental units. Tokenization is beneficial since it divides a text into smaller chunks. Internally, spaCy determines if a "." is a punctuation and separates it into tokens, or whether it is part of an abbreviation like as "B.A." and does not separate it. Based on the problem, we may utilize sentence tokenization or word tokenization.

Topic Modeling and Removing Domain Specific Stop Words Iterations

```
In [62]: vectorizer = CountVectorizer(max_features=20000,
                                     stop_words='english', token_pattern="\b[a-z][a-z]+\b",
                                     binary=True)

doc_word = vectorizer.fit_transform(df_after.TEXT)
words = list(np.asarray(vectorizer.get_feature_names_out()))

In [63]: topic_model = ct.Corex(n_hidden=6, words=words, seed=1)
topic_model.fit(doc_word, words=words, docs=df_after.TEXT)

Out[63]: <corextopic.corextopic.Corex at 0x168922c7d30>

In [64]: # Print all topics from the CorEx topic model
topics = topic_model.get_topics()
for n,topic in enumerate(topics):
    topic_words,_ = zip(*topic)
    print('{:}'.format(n) + ', '.join(topic_words))
topics

0: society,national,group,review,committee,association,director,public,world,human
1: including,strategic,new,major,programs,implemented,million,plan,goals,process
2: testing,user,developer,worked,users,test,used,visual,programming,languages
3: dayjob,readnxcxa,infodayjob,copyright,welcome,nxcopyright,permission,download,jobseekers,template
4: availability,performed,etl,tr,structures,backups,dba,environments,estimating,cobol
5: monthscompany,details,pvt,ltddescription,pune,like,ltdskill,description,various,mvc

Out[64]: [('society', 0.112053327798441, 1.0),
```

Figure 4.1: Topic modeling

5. IMPLEMENTATION DETAILS

The Resume Parser system follows a modular architecture that encompasses various components working together to extract and analyze resume information. The key components of the system architecture are as follows:

Data Visualization:

Data visualization plays a crucial role in the Resume Parser project as it helps in understanding and presenting the parsed resume data in a meaningful and intuitive way. We played with sns plots and heatmap to check the relationship between them.

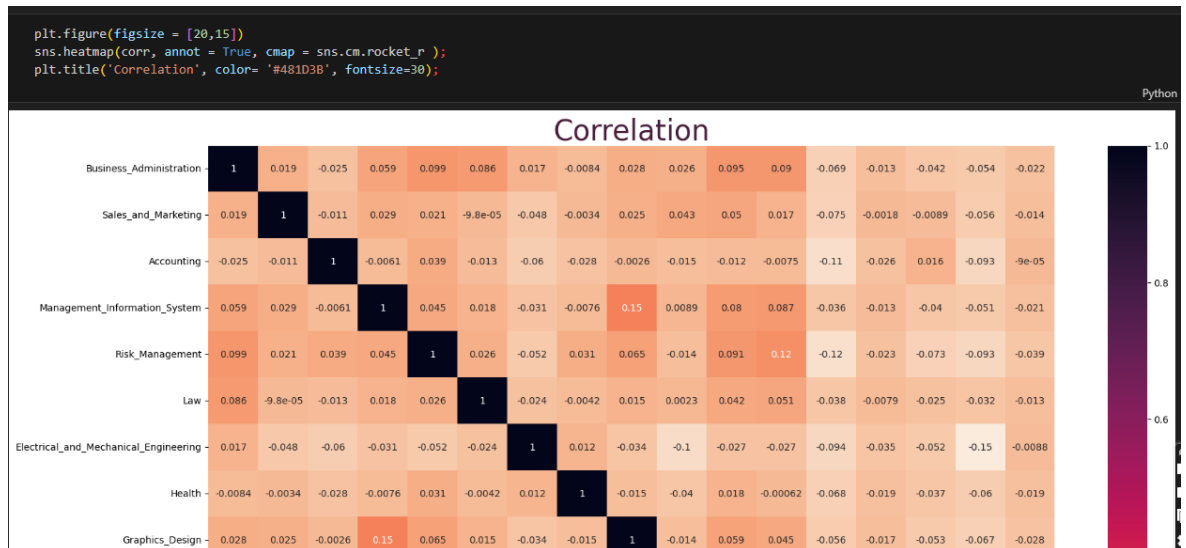


Figure 5.1: Correlation heatmap



Figure 5.2: sns plots

6. Methodolgy

This section discuss the proposed methodology for building an efficient and accurate resume parser. To achieve the objective Machine learning, deep learning, NLP techniques are applied using the best practices.

In the Resume Parser project, several machine learning models were implemented and evaluated for their performance. So in our project we applied several machine learning algorithms. Following are our implemented models discussed with accuracy rate:

Logistic Regression:

Training Accuracy: 0.674297, Validation Accuracy: 0.695376, F1 Score: 0.695376, Recall: 0.695376, Precision: 0.695376,

Logistic Regression is a linear model used for binary classification. It achieved moderate training and validation accuracies, indicating that it could effectively classify the resume data into relevant categories.

Random Forest:

Training Accuracy: 0.998640, Validation Accuracy: 0.950136, F1 Score: 0.950136, Recall: 0.950136, Precision: 0.950136,

Random Forest is an ensemble model that combines multiple decision trees to make predictions. It achieved high training and validation accuracies, suggesting that it performed well in classifying the resume data and avoiding overfitting.

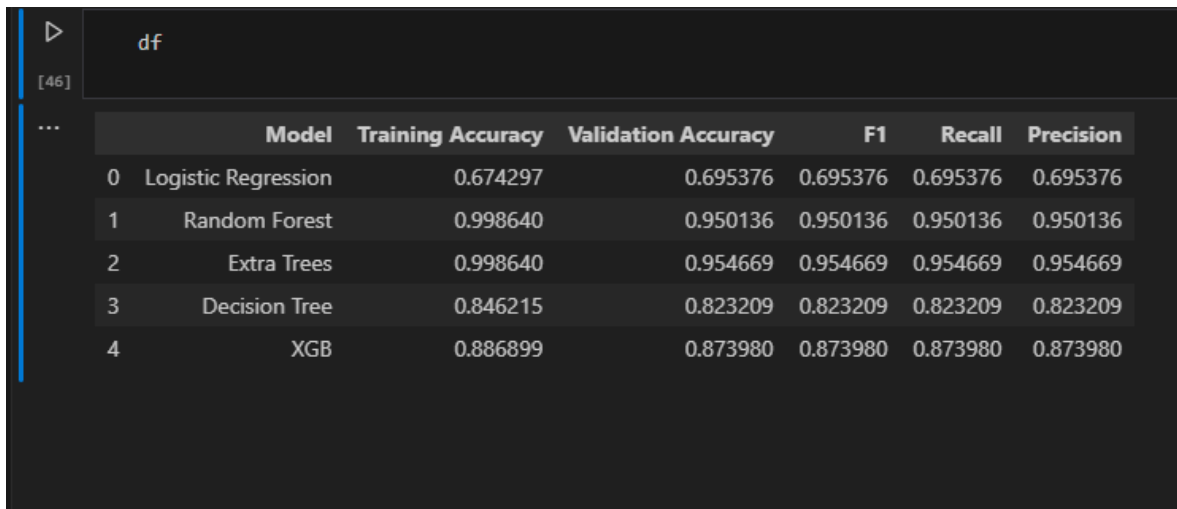
Extra Trees:

Training Accuracy: 0.998640, Validation Accuracy: 0.954669, F1 Score: 0.954669, Recall: 0.954669, Precision: 0.954669,

tra Trees is another ensemble model similar to Random Forest. It also achieved high training and validation accuracies, indicating its effectiveness in classifying the resume data.

Decision Tree:

Training Accuracy: 0.846215, Validation Accuracy: 0.823209, F1 Score: 0.823209, Recall: 0.823209, Precision: 0.823209,



	Model	Training Accuracy	Validation Accuracy	F1	Recall	Precision
0	Logistic Regression	0.674297	0.695376	0.695376	0.695376	0.695376
1	Random Forest	0.998640	0.950136	0.950136	0.950136	0.950136
2	Extra Trees	0.998640	0.954669	0.954669	0.954669	0.954669
3	Decision Tree	0.846215	0.823209	0.823209	0.823209	0.823209
4	XGB	0.886899	0.873980	0.873980	0.873980	0.873980

Figure 6.1: Machine Learning models result

Decision Tree is a non-linear model that splits the data based on different features. It achieved relatively lower training and validation accuracies compared to the ensemble models, suggesting that it may have overfit the data to some extent.

XGB:

Training Accuracy: 0.886899, Validation Accuracy: 0.873980, F1 Score: 0.873980, Recall: 0.873980, Precision: 0.873980,

XGB (Extreme Gradient Boosting) is a gradient boosting model known for its efficiency and performance. It achieved relatively high training and validation accuracies, indicating its effectiveness in classifying the resume data.

Applied supervised ML model's conclusion:

Overall, the Random Forest and Extra Trees models demonstrated the highest performance with high accuracies, F1 scores, recall, and precision. These models are suitable for the Resume Parser project as they can effectively classify the resume data and make accurate predictions.

6.1. Comparing our results with the paper we followed:

The paper [7] that we followed contains all the requirements and detail about resume - parser. But their Machine learning models and my applied models are showing different results. They applied, 7 models were tried and played with to get the best model that goes hand in hand with the dataset. After performing simple train and validation on the 7 models one

was chosen for further investigation. The best model was Decision Tree Classifier. But our applied models showed best models Random forest and extra trees. For improvement of model's performance we then played with deep learning algorithms, BERT, RNNs, CNN. But CNN and RNN didn't showed good performance as compared to other traditional machine learning algorithm then we tried to worked with BERT.

6.2. "Deep Learning Algorithm" - BERT

For evaluating the suitable candidates for various job openings in consideration to their compatibility, the use of deep learning-based system has provided the end-to-end solution for people seeking employment. We focuses on the problem of extracting data from PDF-format resumes and proposes a hierarchical extraction technique. [2] .

BERT

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art natural language processing model introduced by Google in 2018. It revolutionized the field of NLP by leveraging the power of transformer neural networks for pretraining on large amounts of unlabeled text data, followed by fine-tuning on specific downstream tasks.

The key idea behind BERT is to capture the bidirectional context of words in a sentence by using a transformer architecture. Transformers are neural networks that can process sequential data by attending to all the words in the input sequence simultaneously, rather than sequentially. This parallel processing allows the model to capture complex relationships and dependencies between words.

After pretraining, BERT can be fine-tuned for a wide range downstream tasks, such as text classification, named entity recognition, question answering, and more. Fine-tuning involves taking the pretrained BERT model and training it on task-specific labeled data. By fine-tuning, BERT adapts its learned representations to the specific task at hand, which enables it to achieve high performance on various NLP tasks. [3]

BERT model performance: After working with several traditional machine learning algorithms we planned to switch towards deep learning for better performance of our model. After performing all steps including:

Import the necessary libraries and setting up the environment:

such as torch and transformers, are imported. The device is also set to GPU if available,

which enables faster computation using the GPU.

Load the BERT model:

The BERT model is loaded using the BertForSequenceClassification class from the transformers library. This model is specifically designed for sequence classification tasks.

Setting up the optimizer and scheduler:

The optimizer (e.g., Adam) is initialized, and a scheduler (e.g., linear decay with warmup) is set up. These components are crucial for optimizing the model during training.

Training the BERT model:

The model is trained for a specified number of epochs.

Evaluation on the test set:

After training, the model is evaluated on the test set. Similar to the training process, the test data is iterated over in batches. The model's outputs are obtained by passing the test data through the BERT model.

BERT model performance

So after training and testing The reported test accuracy of 96-percent suggests that the BERT model achieved a high level of accuracy in classifying the samples in dataset. The test accuracy is calculated by dividing the total number of correctly predicted samples by the total number of samples in the test set. This accuracy metric provides an indication of how well the BERT model performs on unseen data. This means that out of all the test samples, the model accurately predicted the correct labels for 96-percent of them.

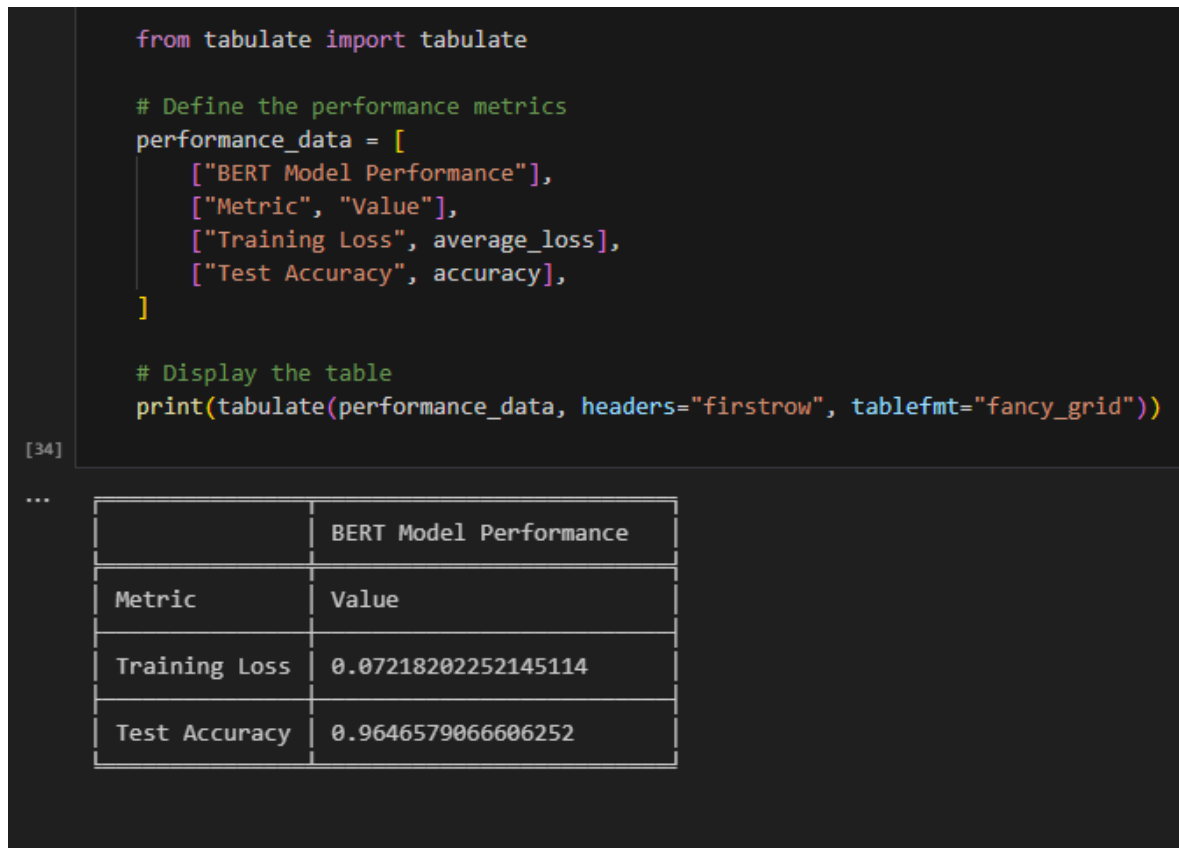


Figure 6.2: Result

7. RESULTS AND DISCUSSIONS

When comparing the ML models' results to the BERT model's performance, it is evident that the BERT model outperformed the traditional ML models in terms of test accuracy. The BERT model achieved a significantly higher accuracy of 96-percent, while the ML models ranged from 69.54 to 95.47. Overall, the results and comparison indicate that BERT, with its advanced deep learning capabilities, offers significant advantages over traditional ML models for resume parsing and classification. Further analysis, including evaluation metrics such as precision, recall, and F1 score, can provide deeper insights into the models' performance across different classes and help identify any potential trade-offs between precision and recall.

8. Future work

In the future, there are several avenues for further development and enhancement of the Resume Parser project. One important aspect is the integration of a database to store the parsed resume data. This will enable efficient data management, retrieval, and scalability as the volume of resumes increases. Additionally, a user-friendly Streamlit application can be developed to allow candidates to easily upload their resumes and receive instant feedback on the parsed information.

Another potential direction for future work is the integration of a chatbot into the application. The chatbot can serve as a virtual assistant, providing guidance to candidates, answering their queries, and offering personalized feedback based on the analysis performed by the resume parser. This interactive feature will enhance the user experience and provide valuable insights to candidates seeking to improve their resumes.

Although these future developments were not implemented due to time constraints in our this semester project with other academic burden, but they offer exciting opportunities for further research and improvement in the field of resume parsing using NLP techniques.

References

- [1] S. Sanyal, S. Hazra, S. Adhikary, and N. Ghosh, “Resume parser with natural language processing,” *International Journal of Engineering Science*, vol. 4484, 2017.
- [2] P. Pokharel, “Resume parser using nlp,” 07 2022.
- [3] C. Ayishathahira, C. Sreejith, and C. Raseek, “Combination of neural networks and conditional random fields for efficient resume parsing,” in *2018 International CET Conference on Control, Communication, and Computing (IC4)*. IEEE, 2018, pp. 388–393.
- [4] N. Bhaliya, J. Gandhi, and D. K. Singh, “Nlp based extraction of relevant resume using machine learning,” 2020.
- [5] “Pdfs resume dataset,” 2021. [Online]. Available: <https://www.kaggle.com/aishikai/resume-dataset?select=pdf>
- [6] 2021. [Online]. Available: <https://github.com/JAIJANYANI/Automated-Resume-Screening-System>
- [7] 2021.