Machine Learning based Resume Feedback and Recommendation System to improve University IT Undergraduates/Fresh Graduates Employability

Team Clay Squad

Index Number	Name
184055D	Hindakaraldeniya T.M.
184126X	Perera N. N.
184186E	Warusawithana S.P.
184188L	Weerasinghe R.L.

Faculty of Information Technology

University of Moratuwa

2023

Machine Learning based Resume Feedback and Recommendation System to improve University IT Undergraduates/Fresh Graduates Employability

Team Clay Squad

Index Number	Name
184055D	Hindakaraldeniya T.M.
184126X	Perera N. N.
184186E	Warusawithana S.P.
184188L	Weerasinghe R.L.

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Honours Degree of Bachelor of Science in Information Technology.

2023

Declaration

We declare that this thesis is our own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Index Number	Name	Signature
184055D	Hindakaraldeniya T.M.	
184126X	Perera N. N.	
184186E	Warusawithana S.P.	
184188L	Weerasinghe R.L.	

Date:

Supervised by:

Dr. (Ms.) G. U. Ganegoda

Date:

Acknowledgement

The successful completion of this thesis is acknowledged with heartfelt gratitude to everyone who has contributed.

First and foremost, the guidance, support, and encouragement provided by our supervisor Dr.(Ms.) G.U. Ganegoda throughout this journey are deeply appreciated. Their expertise and mentorship have shaped the understanding of the subject and have been a source of inspiration to strive for excellence.

The invaluable feedback and insightful suggestions provided by the evaluators panel, Dr.(Ms.) A.T.P. Silva and Dr.(Ms.) Priyanga D. Thalagala, during the review process of this thesis are sincerely appreciated. The critical insights and expertise shared by the evaluators have played a pivotal role in refining and strengthening the content and quality of this work.

Appreciation is extended to the academic staff of Faculty of Information Technology, University of Moratuwa for providing a nurturing academic environment and imparting knowledge that has been fundamental to the development of this thesis.

The unwavering support and camaraderie of friends and colleagues are acknowledged with sincere gratitude. Their constructive feedback and discussions have enriched the research and have made this experience all the more enjoyable.

Gratitude is extended to the industry experts who willingly contributed their time and insights to this study. Without their cooperation, this thesis would not have been possible.

Last but not least, the love, encouragement, and belief in our abilities from our family are deeply appreciated. Their unwavering support has been a constant source of motivation throughout this academic endeavor.

In conclusion, the presence of each individual in our academic journey and their unique contributions are genuinely appreciated, shaping this thesis in significant ways.

Abstract

As a result of the rapid development seen in the field of IT, there has been a surge in the number of students choosing IT field related degrees in recent years. Consequently, the competition between job applicants to secure a better job position has also increased. As an undergraduate, to succeed in their future career, it is crucial that they secure a good job placement as a fresh graduate, since it helps shape his/her future. Resume plays a vital role in securing a job placement as it is often the first document a recruiter will see in the recruitment process. Yet there is no proper solution which automates the resume feedback process that helps undergraduates to prepare for the coming war. Proposed system in this thesis aims to be a supporting tool which enables them to prepare a competent resume by giving feedback regarding their resumes with regard to design and content.

Table of Contents

List of Figures/Tables

Figures

Tables

Chapter 1 - Introduction

1.1 Introduction

With the recent developments in the education system in Sri Lanka, there are many undergraduates in the IT industry who are looking for opportunities in the field of IT. However, due to limited openings in the IT companies, they can only accommodate a few opportunities for undergraduates, resulting in intense competition among undergraduates. When it comes to winning this competition, the resume plays a huge role. A strong resume increases the chance of landing the best opportunity. Yet there is no proper solution on the internet for getting feedback on the resumes prepared by undergraduates [1]. Not only does having a good resume help, but so does having prepared for the interview beforehand. This research presents a Machine Learning based solution for resume feedback and a content recommendation system while providing a possible set of interview questions based on their resume content.

1.2 Background & Motivation

The percentage of school leavers enrolling in higher education programs related to the IT field has surged in recent years owing to the increase in private universities and state universities offering more degrees related to the field. This has also been influenced by the development in the IT industry and job opportunities available globally. Nevertheless, due to the higher number of undergraduates seeking internship opportunities, finding the best opportunity which aligns with an individual's expectations and capabilities has become a challenging task.

Given the high competition in the field, having a well-prepared resume has become a vital factor nowadays even for an internship seeker. Preparing an impactful resume requires paying attention to even the most intricate details as it would most probably be the point where the recruiting team would get the first impression of the applicant. Most of the time, an undergraduate or a fresh graduate seeking a job placement is rather inexperienced in preparing a strong resume.

When preparing a resume, a number of factors need to be considered such as its structure, content, design, grammar, and spelling [2]. An undergraduate can easily be confused about what content to be included and how to properly structure them leading

them to either miss out on important details or include unnecessary details. The structure should be such that it highlights the important points and is easy to read through. Having proper grammar and having no spelling mistakes is another important factor when it comes to a strong resume. When an individual is working on the same document for a long time it is typical that he/she could miss these details, so it requires a third party to review and spot them. Also, one should pay attention to the design of the resume such that it looks impressive and attractive to the recruiter while being professional, and often newbies tend to lack this knowledge and tend to adopt just any design they come through. Further, choosing an appropriate and professional photo is crucial.

Apart from the problems faced in preparing a resume, choosing the most suitable companies to send their applications can be tough as in certain situations educational institutions can impose a limit on the number of companies a student can apply to. Once shortlisted, preparing for the interviews is also an important step in securing a placement. For this, the applicants will have to go through the hassle of browsing through many websites and searching for interview preparation questions which align with the skills and experiences they have mentioned in their resumes.

The hardships we faced in finding internship placements as undergraduate students as well as the lack of an efficient, domain-specific, online resume review system inspired us to carry out this research.

1.3 Problem in Brief

As previously said, to win fewer opportunities in the industry, as the first step, undergraduates need a strong, professional-looking, and content-rich resume. Even if they have a lot of skills, if they cannot structure them properly in their resume, then they will miss out on an opportunity and someone else will grab it. Which emphasis, one of their main weapons in this opportunity-grabbing competition is, a strong resume.

Even though a strong resume is a powerful tool for a candidate [3], undergraduates must go through several steps to create a strong resume. And the last and hardest step is reviewing the resume and it requires expert knowledge. When looking at the past, it can be seen that a lot of undergraduates have been unable to prepare a strong resume to match their skills and experience mainly due to the lack of this expert knowledge.

On the other hand, when they need to apply for an opportunity in the industry, they do not know what organization is more suitable for him/her to apply for. Therefore, it is mandatory to know whether this resume is worth applying for the relevant organization for getting the candidate's dream job. Also, being prepared beforehand for the interviews by refreshing the knowledge on the skills they have mentioned in their resumes is also very important. Sometimes the inexperienced applicants can be unfamiliar with the pattern of questions asked in an interview and hence fail to successfully face the interviews even though they have the knowledge and the skills.

1.4 Aim & Objectives

1.4.1 Aim

Building a Machine Learning based solution for resume feedback system under the criteria of design content, and also give recommendations under missing sections and organizations that a candidate can select for to make a higher chance of getting a job in the field of IT.

1.4.2 Objectives

- Critically review the literature on the existing resume analysis approaches that are being used to analyze resume text content.
- Critically review the literature on the existing resume design analysis approaches using image processing techniques that are being used to analyze resume design and develop a module to analyze resume design and suggest design improvements.
- Obtain the background knowledge required in the domain by experts in the field, resume analyzing techniques used by HR personnel and study in-depth of various technologies, techniques and methodologies that could be used to increase the efficiency and the effectiveness of the proposed solution.
- Study and analyze the background details about current organizations in the field and collect data and criteria with regard to resume analyzing that are currently used in the given organizations.
- Research about calculation methods to calculate the probability of getting selected to each organization based on a resume text content and design content.

- Design and develop a prototype system that could analyze resume content and give scores according to its content quality, company recommendations, feedback regarding missing sections and recommend possible interview questions based on the resume content, so that candidates will have a better chance of getting a job in the field of IT.

1.5 Proposed Solution

To address the above-mentioned problem, a platform is proposed to aid the undergraduates in the process of securing internship placements. The platform allows the users to upload their prepared resume and the companies they wish to apply for. Then based upon those inputs, scores for the design, content quality and feedback regarding missing sections is given to the undergraduates.

Also, the system will recommend a list of interview questions to match the technical skills mentioned in the resume. In addition to that, a list of companies which the uploaded resume is most likely to be shortlisted will be predicted and displayed. This platform consists of four main modules. High level architecture design for the proposed solution is shown in Figure 1.1.

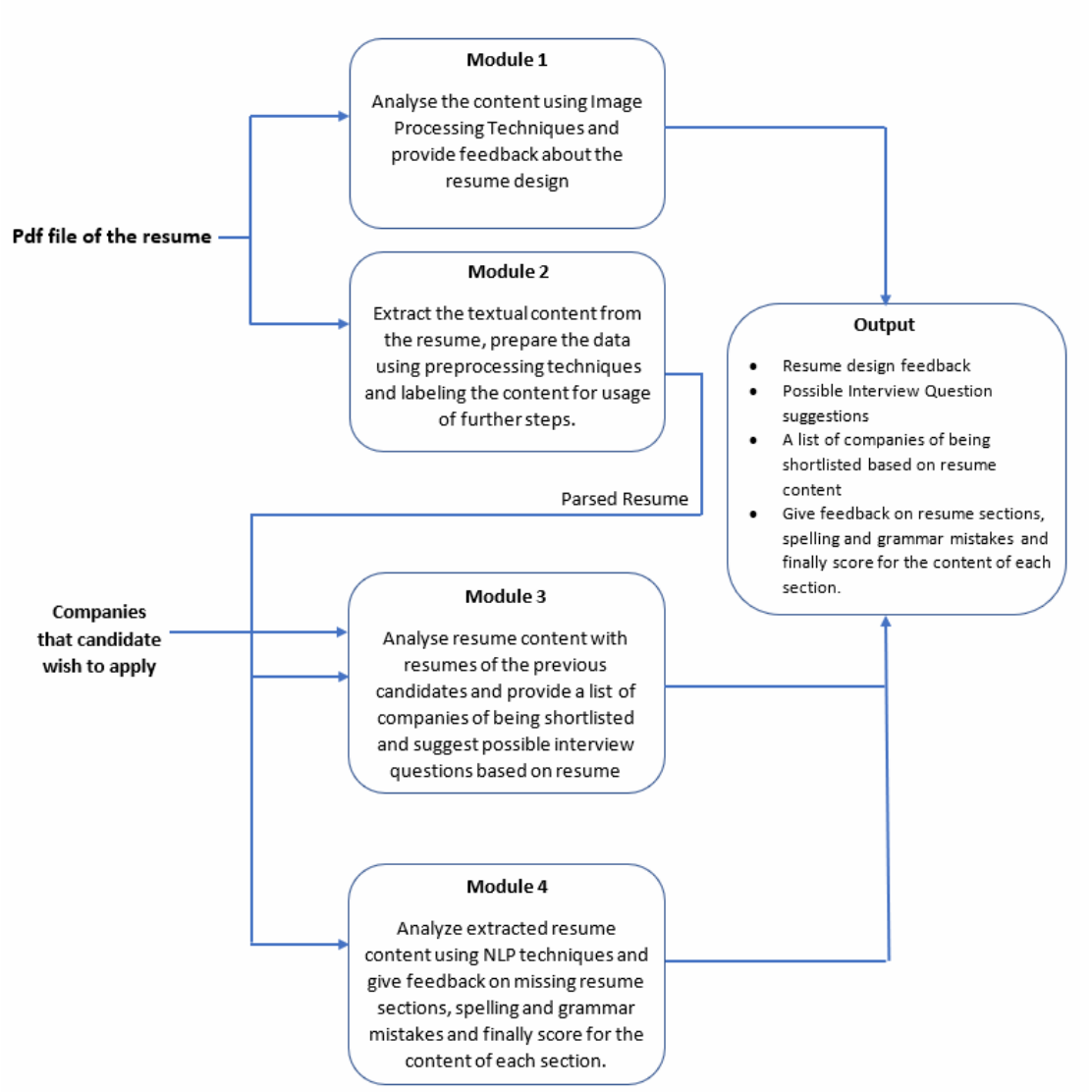


Figure 1.1: Structure of the proposed solution

1.6 Summary

In Chapter 1, introduction to the research, background, and motivation for carrying out this research, the problem in brief, and the aim and objectives of the research are described in detail. Existing work and their literary review is briefed in Chapter 2. Chapter 3, Chapter 4, and Chapter 5 present the technology adapted, approach taken to tackle the identified problems, and the analysis and design of the proposed solution respectively. Implementations of each module up to the level of interim is illustrated in Chapter 6. Chapter 7 includes the evaluations of each module and Chapter 8 consists of the discussion and conclusion which is followed by references and appendices.

Chapter 2 - Review of Literature

2.1 Introduction

In this chapter, existing methodologies, related works, research, and studies regarding resume feedback systems will be discussed. Even though in past research there is no existing machine learning based resume feedback systems, there are numerous studies which can be related to analyzing the resume content and graphical layout of the resume design. The chapter consists of four main subsections which discuss, those related works and literature reviews which can be related and aligned with individual modules.

2.2 Module 1 - Analyzing Resume Design

In this section existing approaches and related works for analyzing resume designs are discussed. In the past research work, even though there is not particular research which focused on machine learning based resume design aesthetic properties such as layout, color combination analysis, there are lots of studies which show the significance of having a good resume design to increase the employability. Recruiters get the first impression of the candidate through the graphical properties of the resume. They see the resume design first and then go through its content. Hence having a good resume design with regard to graphical properties will be beneficial for the candidates [4]. Therefore it is very crucial to have good sense about the recruiters perspective about resume designs which is a combination of user image, color combinations used and resume layout [5]. Furtmueller et al. and Schramm et al. prove this fact through the methods such as surveys, interviews that they have done during their studies.

In modern society there is a controversial opinion about having a creative, colorful resume design and formal simple design. But according to study that Arnulf et al. carried out [6] to check this fact, they have proved that having a professional resume format rather than colorful, creative designs will increase the impact of shortlisting for any job role despite the applying job role. What they have studied mainly about is the graphical layout of resume designs and they have been able to observe some very important facts about the resume graphical designs.

When considering the color combinations of resume designs [6] shows that designs which have colored backgrounds tend to make negative judgment over the candidate than flawless white or whitish colored backgrounds. Resumes which have white and

ivory colors as background were preferred over the colored backgrounds [6], [7]. Further resumed which had gold and blue backgrounds were the least preferred ones [5]. Even Though these studies are carried out using printed resumes, the same facts can be applied to digital resumes as well, since most of the time human perception on colors does not change with the tangibility of the document.

As mentioned above, in the earlier time, undergraduates used printed resumes for finding their job placements. Printed resumes were submitted to the company through posted mail along with a cover letter. With the beginning of the 2010's these practices began to change. Cover letters replaced with the emails; printed resumes replaced with PDFs. With this change, it made the researchers focus more about the impact of document readability based on color combinations. For evaluating color combinations, they introduced a metric called text contrast ratio. The color contrast ratio of a text is mainly based on the background color and the text color. Having a good color contrast ratio will enable the recruiters to grasp the important details very easily and quickly [7]. It will enhance the readability of the recruiter than the low contrast texts [8]. Therefore, having a good text-color-background-color contrast ratio is also a pivotal factor in resume designs.

Another key point which will be considered in analyzing resumes graphically, is the layout of the resume design. Throughout the time there came many common layouts for preparing resumes and some of the resume layouts were deprecated from using with the time. According to [9], ensuring standard level of margins and aligning dates to right or left separately but not with the content, also helps the resumes to have a quality layout.

2.3 Module 2 - Resume Parser

Through this section existing works related to resume parsing and text extraction from pdf files are described. In the past research work, the resume parsing was done in several ways with some limitations.

2.3.1 Resume Parsing and Finding Candidates for a Job Description using BERT

The usage of deep learning-based systems has given a complete solution for the companies for selecting the best applicants for various job vacancies in consideration

of their suitability. The system might be accomplished in two steps: first, by creating a resume parser that extracts all relevant information from applicant resumes, and second, by implementing a BERT classification model which pairs sentences using the BERT algorithm [10]. It had 72.77 percent accuracy in predicting the association between the job description and candidate profiles. In addition to that, the system used structured resumes such as LinkedIn profiles to build the model. As the future work, the system suggests investigating the vision-based site segmentation technique in order to improve structural comprehension of resumes.

As the limitations with the proposed solution, this system only worked for the structured resumes formats and the system provided a solution for the company not for the candidate.

2.3.2 Resume Information Extraction with a Novel Text Block Segmentation Algorithm

The system demonstrated a resume parsing pipeline that utilizes distributed embeddings and neural network-based classifiers. This pipeline helped to reduce the time which needs to manually review the resume content. The proposed system combines text block segmentation with resume fact identification using position-wise line information and integrated word representations and named entity recognition using multiple sequence labeling classifiers inside labeled text blocks [11]. They have concentrated on six important areas to standardize the resume parsing process: personal information, work experience, education, project experience, publications, and professional skills. When considering this system with the proposed system, this system only parses resume details to the company side.

2.3.3 Information Extraction from Resume Documents in PDF Formats

This system concentrates on the issue of data extraction from resumes in PDF format and suggests a hierarchical extraction method. The detailed information extraction problem is approached as a sequence labeling problem in some publications, and divide a page into blocks using heuristic criteria, categorize each block using a Conditional Random Field (CRF) model, and then extract the detailed information [12]. Top level diagram of this system is as follows (Figure 2.1).

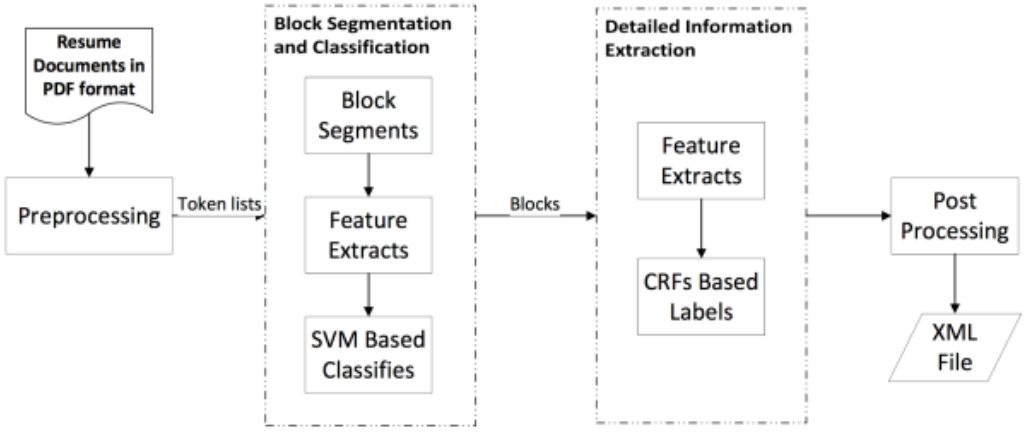


Figure 2.1: Top level diagram of the system proposed in 2.3.3

2.3.4 An Automated Resume Screening System Using Natural Language Processing and Similarity

The systems which are implemented in [13]–[15] are used to extract the text content from the resumes and then easily filter out the resumes for the job position by considering the included content in the resume. As the first step the system involves the text extraction method using natural language processing. After extracting the content from the resume, it performs the tokenization, lemmatization, part of speech tagging, chunking and the named entity recognition processes for the extracted text content. As the second approach the system includes a content-based candidate recommendation system which can utilize the content which is extracted from the first step and then recommend the most appropriate resumes for the given job description. TF-IDF vectorization techniques were used in the vectorization process and then the system calculated the Cosine similarity to find how much the two objects are alike. Finally, as the output the system generates a JSON file with the extracted entities (Figure 2.2).

```
{
 "name": "CHIRAG DARYANI",
 "email": "chiragdaryani28@gmail.com",
 "mobile_number": "9977777777",
 "skills": [
  "Debugging", "C", "Database", "Advertising", "Content", "Api",
  "Sql", "Html5", "Certification", "Js", "Javascript", "Java",
  "Android", "Technical skills", "C++", "Nltk", "Pandas",
  "Matplotlib", "System", "Programming", "Numpy", "Algorithms",
  "Analysis", "Mysql", "Spring", "JSF", "JDBC", "Writing", "Html",
  "Opencv", "Python", "R", "Textblob", "Css", "Testing", "Technical"
 ],
 "college_name": [
  "Medi-Caps University, Indore"
 ],
 "degree": [
  "Bachelor of Engineering – Computer Science"
```


Figure 2.2: Output JSON file format

When comparing this system with the proposed system, the layout of the resume would be considered in the proposed system, and it would extract content for each section. In addition to that, the proposed system would extract all the text content, but this system only extracts the entities.

2.4 Module 3 - Company recommendation system and Interview questions suggestion

2.4.1 Background research on problem

In the employee recruitment area, a considerable amount of research was found relating to incorporating content analysis and resume categorization.

In the online resume parsing system proposed by Chandola et al. [15], a knowledge base is first created with various keywords (and an associated value indicating the importance of each keyword) which are made from the initial training data. For preprocessing input data, a POS tagger and a chunker are used to split the text into sentences, which are then analyzed by a syntactic parser which labels all the words with their part of speech information. Sum of all keyword values obtained from analyzing the resume content is used to give a rank to the resume to compare with other different candidates resumes. The algorithms for matching keywords have been implemented using Python facilitated by MySQL connector. Score is also used to categorize candidates as low, average, and high. This categorized level is considered as the expertise level of the candidate.

Anand et al. [16] proposed a system to reduce the complexities in the candidate recruitment process using machine learning approaches. In their proposed system, resumes were taken as input from clients and the recruiters were allowed to post job descriptions. Keyword extraction is performed on input resumes using section-based segmentation with NLP. Also, in this system a model is trained for a job posting where resumes of similar type of job postings are collected and relevant skills are extracted. Once a candidate applies to a job posting, this system will compare the keywords extracted from resume against the job posting and will output a score to the recruiters so they can select candidates accordingly. The score has been calculated by giving more weightage to sections as 50% weightage to skill set, 20% weightage to education, 20% weightage to work experience and 10% weightage to number of companies the candidate has previously worked in.

A system named 'ATS Breaker' was proposed by Mathew et al. [17] for comparing the candidate resumes and company requirements. This system has been developed to aid the job seekers to have an idea about how to match their resumes with the job postings they are interested in. Initially, the user is asked to upload their resume along with the job description of the job which he/she intends to apply for. Then the system extracts the information about work experience, education, and other basic details from the resume and the job description and stores them in two separate dictionaries. In this extraction process, if an unseen skill is detected it is categorized using regular expressions. To output the similarity, the proposed model had used the word counts present in resume and job description to calculate the match percentage.

Through the research conducted on the background of the problem within our domain, a reasonable amount of research was found on checking similarity between job descriptions and resume content. Also, it was researched before on how to rank a given set of resumes considering their content, to assist recruiters shortlist candidates. Considering the proposed solution, there was no research on a system to predict the suitability of a resume to a certain company based on resume content that could be found. In our domain, the approach of matching job description to resume content was found to be not effective, as through the knowledge gathered through conducting meetings with industry experts it was found that they do not consider much about candidate's familiar technologies versus available job openings for each technology when shortlisting fresh graduates and interns. Therefore, the research on a correlation

between a candidate being selected in the first resume screening process of a certain company and the resume content poses a research gap in the employee recruitment domain.

2.4.2 Multi-label Classification

In the context of artificial intelligence, the objective of classification is to create a model that has the ability to accurately classify a new unseen and unlabeled data instance by training that model priorly with a set of labeled data. One of the oldest machine learning tasks is traditional single-label categorization. It offers quick and precise predictions and has a wide range of successful applications in a variety of domains. Single label classification can be categorized further as binary and multi-class classifications, which are concerned with learning from a group of samples connected to a single label. Multi-label classification (MLC) assigns a collection of pertinent labels to an instance simultaneously, in contrast to traditional classification [18]. In the problem focused on in this research, the task requirement is to output a list of companies matching for a single resume instance. Hence, addressing the issue as a multi-label classification problem was found to be best suited.

2.5 Module 4 - Analyze resume content and suggest content improvements

In relation to the content of a resume, there are numerous studies which were carried out throughout the past few decades. Most of the studies are carried out by focusing on the recruiter perspective to rank a set of resumes for a relevant job position. During this process resumes are ranked by considering the content on the resume. First part of this section will be focused on the studies carried out related to the contents of a resume and the latter part is focused on analyzing the resume content.

2.5.1 Contents of a Resume

The First impression about the job applicant is usually made through the content of the resume [19]. Hence it is important to have good and quality content on the resume. In the study by Burns et al.[9], it was found that there exists a correlation between the content included in a candidates' resume with his/her personality.

Schramm et al. [5] carried out a research about what constitutes the ideal resume by collecting information through a questionnaire from recruiters. This research pointed

out the importance of the content, format and appearance of a resume. When it comes to studying resumes by a recruiter, as shown in the results, 26.1% of recruiters spend only 30-60 seconds and around 30% of the recruiters spend 1-2 minutes while 27.5% spend 2-3 minutes. Hence, it is important to mention content in a way to gain attention in such a small time. According to the study, 90.8% of recruiters agreed that the permanent address should appear on the resume. When it comes to the education details, almost all the recruiters agreed that the college where the candidate graduated should be mentioned in the resume. Also, more than 82% of recruiters mentioned it is better to add the other colleges attended to the resume. Almost every recruiter mentioned that working experience should be included in the resume while 94% of the recruiters indicated that the date of employment also should be there. In addition to that, 74% of the recruiters preferred to have an explanation of the work experience as well. Moreover, more than 85% of the respondents concurred that extracurricular activities as well as achievements ought to be listed in the resume. According to the study, when considering the length of the resume, a maximum of two pages were favored by two thirds of the recruiters. Between 70% and 80% of the recruiters concurred that a disinterest in a prospect will result from poor arrangement and a lengthy resume. More than 80% of those surveyed gave the order in which the things appeared on a resume some level of significance. When it comes to the grammar and other mistakes, 95% of respondents believed that more than one spelling mistake or poor grammar will make some recruiters less interested in a candidate. More than 80% of the recruiters said that having multiple typing errors on a resume would make it more difficult for the applicant to have an interview. Through this study it has identified the importance of quality content, format, order of a resume as mentioned above.

In the study by Risavy [20] also mentioned the importance of the resume content. According to that study, the following information should be included in a resume.

- Personal information
- Personal opening, job objective, career objective, and summary of qualifications
- Education
- Work experience
- References
- Scholarships, awards, honors
- Hobbies, interests, and extracurricular activities

- Willingness to relocate and travel

Additionally, this study mentioned what are the most important details that should be included in the above sections. According to the research applicant's name, phone number and address are mandatory information that should be included as personal information. Moreover, it has identified that early and more contemporary resume research both concur that formal educational qualifications, such as degree or the designation and the major, minor, and if applicable, the anticipated graduation date should be listed on resumes. Moreover, according to this study, candidates should include information about their former jobs, including the dates they worked there, the position they had, and whether or not it was a full or part time one. Apart from that information, it also mentioned about the length of the resume and the order of the contents. When considering the length of the resume, this study too mentioned that three-page resumes should be avoided in favor of one-to-two-page resumes. Moreover, this study mentioned that the content should better be in the order of personal information, education, work experience, and finally extracurricular activities. Additionally, it mentioned that the details such as height, weight, race, religion, birth date, marital state, number of dependents, physical/health status, and social security number should not be included in the resumes.

According to both early and recent research, it is important to have good content, ordered perfectly within one or two pages when preparing the resume. Because a resume is the first thing which creates the image about the applicant on recruiters head and recruiters only spend very little time to go through a resume. Therefore, it is important to gain the attention of the recruiter for the resume within a few seconds.

2.5.2 Analyzing Resume Content and Scoring

There are several research which have been done for analyzing the resume contents. But most of them focus on how to shortlist candidates for a job position by identifying the talents of the candidate mentioned in the resume. Those approaches didn't focus on scoring the resume content based on quality and the contents. Primary focus of those approaches was to extract skills and other prominent information and ranking the candidate based on that data.

In the study by Hewage et al. [21], a method has been proposed to create resumes using Gensim LDA unsupervised machine learning model according to Hoffman method. This method only gives an overall rating for the resume by only considering the relevant skills and particular job. It does not rate the content of each section.

In the study by Zimmermann [22], scores are given for only the extracted content of the education, skills and work experience sections based on the predefined score criteria by considering the job description. This approach does not consider the quality of the overall content of the section, only pay attention to the contents they are looking for. For candidates to create a proper resume, they need to have quality content on each section.

The study by Shestakova [23] proposed a method to analyze contents of the resume and recommend job vacancies by considering the similarities of the job description and resume content. BERT and DistilBERT models applied when developing this approach. To measure the similarity, the method has utilized cosine similarity approach. Even though this approach analyzes the contents of the sections, it doesn't provide any feedback about the quality of the contents of each section.

2.5.3 Spelling and Grammar Checking

There are numerous research studies which have been done for developing spelling and grammar checkers for various languages.

The goal in the approach [24]is to develop a text2text language model specifically designed for "single-shot grammar correction". This model aims to effectively correct grammatical errors in text while preserving the original meaning and information. It focuses on texts that may contain numerous mistakes, ensuring that only grammatical improvements are made without introducing any semantic changes to the content.

The model developed in the library Gramformer[25] exposes 3 separate features to detect, highlight and correct grammatical and spelling errors. This model has been trained on 64 long sentences and operates at the sentence level. There are more than 80 models listed on hugging face that were developed through various research studies. Among those approaches, Gramformer is one of the most utilized models for grammar and spelling check.

Moreover, there are some APIs like Bing, Ginger, LanguageTool, and Sampling. These APIs also could be integrated for spelling and grammar checking when developing a platform. Some of those have free versions while providing pro versions with more features.

2.6 Summary

As described in the above subsections, there is no direct implementation which focuses on developing machine learning based resume feedback systems. Yet the available techniques and practices used in resume parsing, color combination and layout analysis of resumes, resume content matching with another context, and resume quality ranking are utilized in the proposed solution to overcome the identified problem in mentioned context, IT undergraduates.

Chapter 3 - Technology Adapted

3.1 Introduction

This chapter focuses on describing the programming languages, development platforms, algorithms and technologies, libraries used to tackle the above-mentioned problems. After the initial research and literature review of each module, these technologies are adapted in the proposed solution. Further in this chapter it is clearly described why the respective technologies are adapted to develop the proposed solution.

3.2 Programming Languages

3.2.1 Python

Python is used as the primary programming language used in all four modules for the implementations. Python is the most commonly used programming language for implementing machine learning algorithms. Due to its simplicity, consistency, wide range of libraries, and the large community support it has proven to make implementing machine learning applications much easier.

3.3 Development Tools and Platforms

3.3.1 Google Colaboratory

Google Colaboratory is a platform introduced by Google Research which integrates the Jupyter development environment in a cloud platform. It enables the researchers to easily share their colab notebooks which makes collaborated development more fluid. Also, most importantly it provides its users with the ability to carry out high performance computing with its' free of charge GPU (Graphical Processing Unit) access. This makes it more accessible and affordable for researchers who have no access to GPU computing which has more parallel computing abilities than normal CPU computing.

3.3.2 Jupyter Notebook

As a tool which is used for interactive data science and scientific computing, Jupyter Notebook is used as the main development environment for the proposed solution. Due to its flexibility in coding and executing, it makes development of machine learning applications faster.

3.3.3 Label Studio

Label studio is an open-source tool which can be used for labeling and analyzing multiple types of data including text, audio, images, structured data. This can be integrated with machine learning models to generate predictions for labels. In the proposed solution, this tool is used for annotating the data from resumes and creating the dataset to implement a model to the process of resume parsing.

3.3.4 NER Annotator for spaCy

NER annotator for spaCy is an open-source software tool that aids in the generation of training data for named entity recognition (NER) models in spaCy. Users can quickly and simply annotate text with named entities using this tool, which is based on the ipywidgets library.

3.4 Algorithms and Techniques

3.4.1 Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) is a state-of-art machine learning technique most commonly used for Natural Language Processing (NLP) and Natural Language Understanding (NLU) tasks. For the implementation of predicting the most matching company list for a given resume, this model is used due to its ability to identify the context in large text.

3.4.2 Naive Bayes

Naive Bayes is one of the well-researched probabilistic models which has shown higher accuracy levels in binary classification and multiclass classification tasks. To appropriately output the relevant resume section when given the content of a section, Multinomial Naive Bayes algorithm is used in our context, which is a variant of the Naive Bayes algorithm.

3.4.3 Support Vector Machines (SVM)

A Support Vector Machine is a supervised machine learning algorithm used for classification and regression tasks. It is particularly effective when working with

complex datasets with a clear separation between classes or when dealing with high-dimensional feature spaces.

3.4.4 Linear Regression

A machine learning approach based on supervised learning to perform regression tasks. In the proposed solution, a linear regression algorithm is used for finding out the relationship between the resume content of the given section and the given score for that content.

3.4.5 Decision Tree

Decision Tree is a supervised machine learning approach for classification and regression tasks. It is a model that resembles a flowchart, with each internal node standing in for a feature or characteristic, each branch standing in for a decision rule, and each leaf node standing in for the result or prediction. In order to create predictions, the decision tree algorithm divides the data into multiple categories based on those properties.

3.4.6 Random Forest

In order to perform classification, regression, and other tasks, the random forest ensemble learning method builds a large number of decision trees during the training phase. The class that the majority of the trees chose is the output of the random forest for classification problems. The mean or average prediction of each individual tree is returned for regression tasks.

3.4.7 Machine Learning & Deep Learning approaches for multi-label text classification

3.4.7.1 Problem transformation methods

In problem transformation approach it converts the original multi-label problem into one or more single-label classification or regression problems that can be resolved with the aid of already existing conventional algorithms[26] such as, Bayesian classifier, logistic regression, and support vector machines. Any of these classic algorithms can be adapted to address multi-label classification problems by incorporating the following techniques for the transformation.

Binary relevance - Each label is treated as a separate binary classification task using the Binary Relevance (BR) approach. A separate classifier is created for each label, which is trained separately using the features of the input data. The classifiers individually assess each label's existence or absence during prediction (multi label problem transformation methods).

Label powersets - Label Powerset (LP) approach turns a multi-label classification problem into a single multi-class problem. Each separate class in LP is handled as a unique combination of labels. With this approach, the multi-label classification problem is effectively treated as a multi-class classification task, and a classifier is trained to give each occurrence the appropriate set of labels (multi label problem transformation methods).

Classifier chains - Classifier chains are a technique that improves conventional classification algorithms by taking label dependencies into account. By building a series or chain of binary classifiers, each of which is trained to predict a single label based on the input features and the predictions of earlier chained classifiers, classifier chains try to capture the correlations between labels[27].

3.4.7.2 Algorithm adaptation methods

Without the need for problem transformation, some classification algorithms/models, such as (k-nearest neighbor, Decision trees), have been extended to handle the multi-label data directly[28].

K-Nearest Neighbors - KNN can be used in multi-label classification, where one instance may be simultaneously associated with numerous labels, by expanding the method to take surrounding instances' label sets into account. The Multi-Label K-Nearest Neighbors (ML-KNN) extension extends the standard KNN method to deal with multi-label scenarios. Based on the labels of its k nearest neighbors, ML-KNN assigns labels to an instance[28].

Decision Trees - Although they are more frequently used for single-label classification, decision trees can also be used for multi-label classification tasks. By changing the splitting criteria and decision rules at each node, decision trees can be expanded to handle multiple labels in the context of multi-label classification[28].

3.4.7.3 Ensemble methods

Ensemble approaches combine the predictions of various distinct models, frequently referred to as "base models," to make predictions that are more precise and trustworthy.

Bagging Classifier - The Bagging Classifier is an ensemble technique that makes predictions by combining several base classifiers. By using a technique called bootstrap sampling, each base classifier is trained using a random subset of the training data. The Bagging Classifier aggregates the base classifiers' predictions during prediction, to produce the final prediction[29].

Random Forests Classifier - To produce precise and reliable forecasts, Random Forests aggregate the predictions of various decision trees. By extending the decision trees to handle several labels, Random Forests can be modified for use in multi-label classification[29].

Gradient Boosting Classifier - With gradient boosting, weak models, such decision trees, are sequentially trained, with each new model aiming to correct the flaws of the prior ones[29].

3.4.7.4 Neural Networks

3.4.7.4.1 CNN

Despite being created mainly for image processing, CNNs have been successfully applied to text classification problems as well [20]. In Natural Language Processing based classification tasks, the sentences are inputted into the CNN as a matrix where a single row in the matrix will represent word embeddings for the given sentence [21]. The issue with using CNNs for text classification tasks is that unlike the limited number of channels available in image processing, it will have a larger number of channels leading to high dimensionality issue [22]. When the length of the sentences inputted into the CNN increases, the computation cost for recognizing the relationships between the words in a sentence also substantially increases [23] in this architecture making it unsuitable for analyzing large textual content.

3.4.7.4.2 RNN

RNN was popular at the time it was introduced due to its ability to work with sequential data by showing higher performance levels. These architectures try to capture latent

links between context words by extracting information about sentence structure. An RNN model typically receives a sentence as input, represented by a series of word embeddings, with each word entering the model one at a time [21]. One of the major bottlenecks in large text analysis using RNNs is that it will process the text in a sequential manner [23]. Further, RNNs are very slow to train and have a very poor memory, as in it cannot remember old connections well. This is termed as the vanishing gradient issue.

3.4.7.4.3 LSTM

LSTM was introduced to address the issue of vanishing gradient observed in traditional RNNs [24]. Since its introduction, LSTMs has been widely used in various NLP tasks making it the most popular variant of RNN [21]. Though LSTMs have a stronger memory than basic RNNs, still it has a limited memory capacity.

3.4.7.4.4 Neural networks with attention mechanism

Attention mechanism was first introduced by Bahdanau et al. [25] and the idea was to make the learning process to give more attention to more relevant parts of the input sentences. In language models, attention can be thought of as a vector of importance weights. In order to predict a word within a phrase, we assess its correlation strength with other words using the attention vector. Thereafter, the sum of these calculated estimated correlation values weighted by the attention vector is taken for the prediction [23].

To push the boundaries beyond the capabilities of different RNN architectures, in 2017, Vaswani et al. [26] proposed the transformer architecture. The high computational cost generated when analyzing large text using RNNs owing to their sequential processing was addressed by the encoder-decoder based transformer architecture by applying the self-attention mechanism. This architecture allows the input tokens to be processed simultaneously rather than sequentially like in RNNs and CNNs [23]. This self-attention mechanism enables the modeling of dependencies regardless of the distance of input and output sequences [26], making this architecture efficient in analyzing large text content while also taking context into consideration.

3.5 Libraries

3.5.1 TensorFlow

For the purpose of training neural networks used in the proposed solution TensorFlow is used as one of the main libraries. What makes TensorFlow suitable for our implementations is its features provided such as the ability to track models, model training, and performance monitoring.

3.5.2 PdfMiner

PDFMiner is a library which can be used to extract text data from resumes. This library is written in pure python language, and it obtains the exact location of text as well as other layout information. In the proposed solution, this library is mainly used to extract all the text content from resumes without considering the layouts.

3.5.3 Pdf2Image

For converting PDF files to images for design analysis, Pdf2Image python library is used. This module reads the PDF file and returns a PIL image converted array, which will be utilized and analyzed in the further processes in design analysis.

3.5.4 spaCy

spaCy is an open-source library which can be used to deal with the Natural Language Processing tasks in python. In the proposed solution, spaCy is used for the data preprocessing tasks and extracting relevant details using Named Entity Recognition which is an inbuilt feature in the spaCy library.

3.5.5 NLTK

The Natural Language ToolKit (NLTK) is a python library for natural language processing for the English language. This library processes natural language by considering statistical data. In the proposed solution NLTK library is mainly used for preprocessing the input data.

3.5.6 Layout Parser

Layout parser is a unified toolkit library for deep learning-based document image analysis. Layout Parser also comes with full support for customized layout model

training on a custom dataset. This enables us to achieve optimal prediction accuracy on a custom dataset and can simplify the pipeline. In the proposed solution, this library is used to read the annotated dataset from the ‘coco’ file format.

3.5.7 OpenCV

As one of the most commonly used computer vision libraries, OpenCV is used in reading and manipulating images in image processing tasks. After reading the PDF files as images for preprocessing tasks such as converting colors to RGB, resizing, and cropping images in the proposed solution, OpenCV is used.

3.5.8 Tesseract OCR

Tesseract OCR is used as a primary library which is used to extract text data from an image. In the proposed solution, this library is mainly used to extract text data from separate image blocks which are identified from layout parser. Further, this library is used to identify the bounding boxes for individual text blocks for the process of identifying color contrast ratio of texts.

3.5.9 Pandas

Pandas is a python library for working with datasets and it contains functions for analyzing and manipulating data. When it comes to the development of the proposed solution, the pandas library is mainly utilized to work with datasets.

3.5.10 Gramformer

Gramformer is a library developed for detecting, highlighting, and correcting grammatical errors on natural language text. It includes a quality estimator to guarantee the advised repairs and highlights are of the highest caliber.

3.6 Summary

Under chapter 3, it briefly explains the technologies utilized for the development of the proposed solution. As mentioned above Python is the main programming language used for the developments. Due to the requirement of high-performance computation, platforms like Google Colaboratory are utilized for the developments. When it comes to the creation of the dataset, we use Label Studio for the purpose of annotating the

dataset. Libraries like ‘sklearn’ [30], ‘numpy’, ‘re’, are not described above because they are commonly used libraries in most ML-based projects.

Chapter 4 - Approach

4.1 Introduction

This chapter focuses on describing the proposed system to tackle the above-mentioned problem. The first sub chapter contains the details of the proposed solution with reference to its data collection, users, inputs and outputs. The final subchapter consists of the details on how each module addressed the identified problems using the technologies adapted.

4.2 Proposed System

A platform is proposed to assist undergraduates in the process of finding internship assignments and fresh graduates in the process of securing their first job placements in order to alleviate the issues encountered in the resume preparation stage. Users may post their prepared resumes and the companies they would like to apply to on the portal. The undergraduates will then get suggestions for improving the content. Feedback will be offered based on factors including design and content quality.

Additionally, the system will suggest a set of interview questions that correspond to the technical skills listed in the resume. Additionally, based on historical selection data, a list of companies that a resume would be most likely to be shortlisted will be displayed. There are four primary modules on this platform.

4.2.1 Data collection

Data collection step is considered to be one of the most crucial steps when building a machine learning model as the quality, volume, dispersion, accuracy and reliability will have a significant impact on the output of the model.

Several different data collection methods had to be adopted in order to collect data to suit the requirements of different modules. To successfully train the proposed models, a large number of resumes of previously selected candidates were required. The resume list and their details including the selected company were obtained from the system administrator of the industrial training platform of Faculty of Information Technology, University of Moratuwa.

After acquiring the required number of resumes the biggest challenge was annotating the resumes according to the model requirements. To tackle it, label studio was utilized to annotate the resumes manually into appropriate sections. Annotated resume using Label Studio is shown in Figure 4.1.

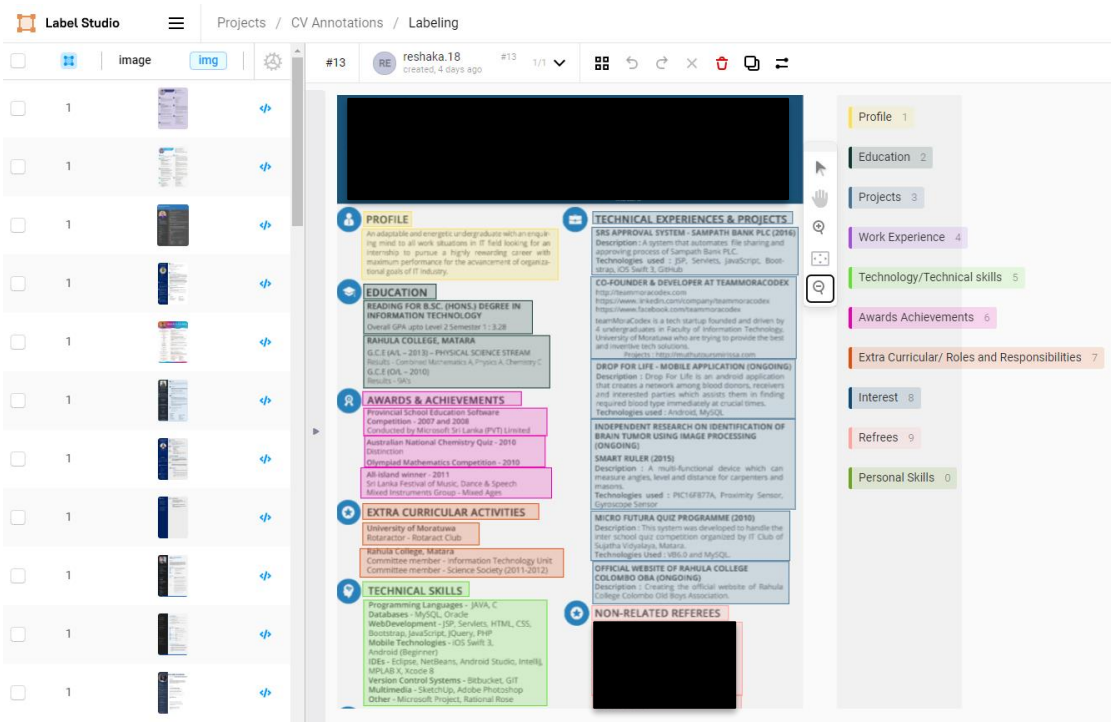
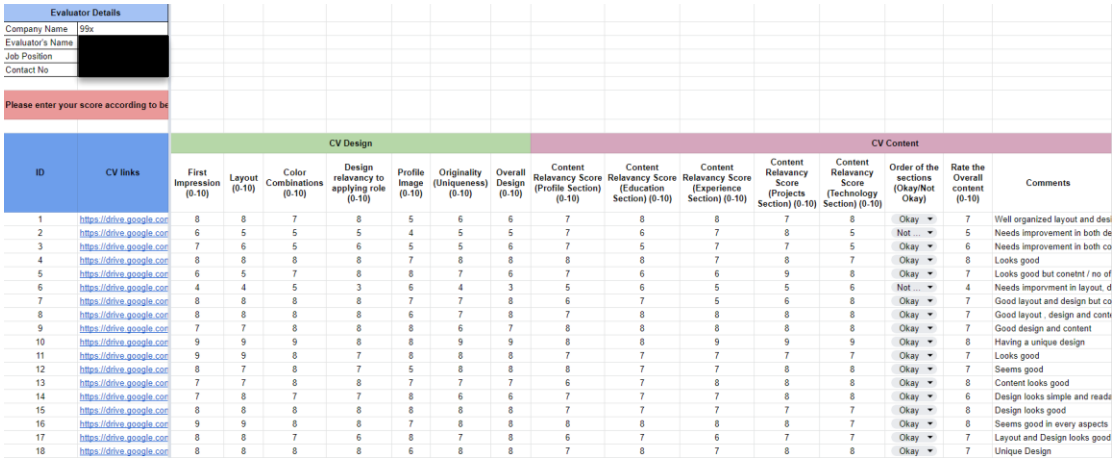


Figure 4.1: Annotated resume using Label Studio

Several meetings were held with HR experts from the industry to get a clear idea on the criteria they take into account when shortlisting the resumes, the sections they focus the most during initial resume screening, and other related details. According to the knowledge gathered through these meetings, a scoring sheet (Figure 4.2) was created to distribute among the HR personnel in the industry.



Evaluator Details	
Company Name	99x
Evaluator's Name	[redacted]
Job Position	
Contact No	

Please enter your score according to be

		CV Design							CV Content							
ID	CV links	First Impression (0-10)	Layout (0-10)	Color Combinations (0-10)	Design relavancy to applying role (0-10)	Profile Image (0-10)	Originality (Uniqueness) (0-10)	Overall Design (0-10)	Content Relavancy Score (Profile Section) (0-10)	Content Relavancy Score (Education Section) (0-10)	Content Relavancy Score (Experience Section) (0-10)	Content Relavancy Score (Projects Section) (0-10)	Content Relavancy Score (Technology Section) (0-10)	Order of the sections (Okay/Not Okay)	Rate the Overall content (0-10)	Comments
1	https://drive.google.com	8	8	7	8	5	6	6	7	8	8	7	8	Okay	7	Well organized layout and desi
2	https://drive.google.com	6	5	5	5	4	5	5	7	6	7	8	5	Not ...	5	Needs improvement in both de
3	https://drive.google.com	7	6	5	6	5	5	6	7	5	7	7	5	Okay	6	Needs improvement in both co
4	https://drive.google.com	8	8	8	8	7	8	8	8	8	7	8	7	Okay	8	Looks good
5	https://drive.google.com	6	5	7	8	8	7	6	7	6	6	9	8	Okay	7	Looks good but conetnt / no of
6	https://drive.google.com	4	4	5	3	6	4	3	5	6	5	5	6	Not ...	4	Needs imporvment in layout, d
7	https://drive.google.com	8	8	8	8	7	7	8	6	7	5	6	8	Okay	7	Good layout and design but co
8	https://drive.google.com	8	8	8	8	6	7	8	7	8	8	8	8	Okay	7	Good layout , design and conte
9	https://drive.google.com	7	7	8	8	8	6	7	8	8	8	8	8	Okay	7	Good design and content
10	https://drive.google.com	9	9	9	8	8	9	9	8	8	9	9	9	Okay	8	Having a unique design
11	https://drive.google.com	9	9	8	7	8	8	8	7	7	7	7	7	Okay	7	Looks good
12	https://drive.google.com	8	7	8	7	5	8	8	8	7	7	8	8	Okay	7	Seems good
13	https://drive.google.com	7	7	8	8	7	7	7	6	7	8	8	8	Okay	8	Content looks good
14	https://drive.google.com	7	8	7	7	8	6	6	7	7	7	8	8	Okay	6	Design looks simple and reada
15	https://drive.google.com	8	8	8	8	8	8	8	7	7	7	7	7	Okay	8	Design looks good
16	https://drive.google.com	9	9	8	8	7	8	8	8	8	8	8	7	Okay	8	Seems good in every aspects
17	https://drive.google.com	8	8	7	6	8	7	8	6	7	6	7	7	Okay	7	Layout and Design looks good
18	https://drive.google.com	8	8	8	8	6	8	8	7	8	7	8	8	Okay	7	Unique Design

Figure 4.2: A completed scoring sheet which was distributed with an HR personnel

4.2.2 Users

Our system is primarily designed for the use of internship candidates and fresh graduates of the Faculty of Information Technology, University of Moratuwa. Users will be able to review their resumes and get feedback, suggested questions for interview and a list of companies that the resume could be shortlisted.

4.2.3 Inputs

The main input of the system are the resume and the list of companies that the graduate wishes to apply for.

4.2.4 Outputs

Primary outputs of the system are feedback on the design of the resume, a section wise score for the quality of their contents, feedback on missing sections if there is any, and a list of companies that the resume is most likely to be shortlisted.

4.2.5 Process

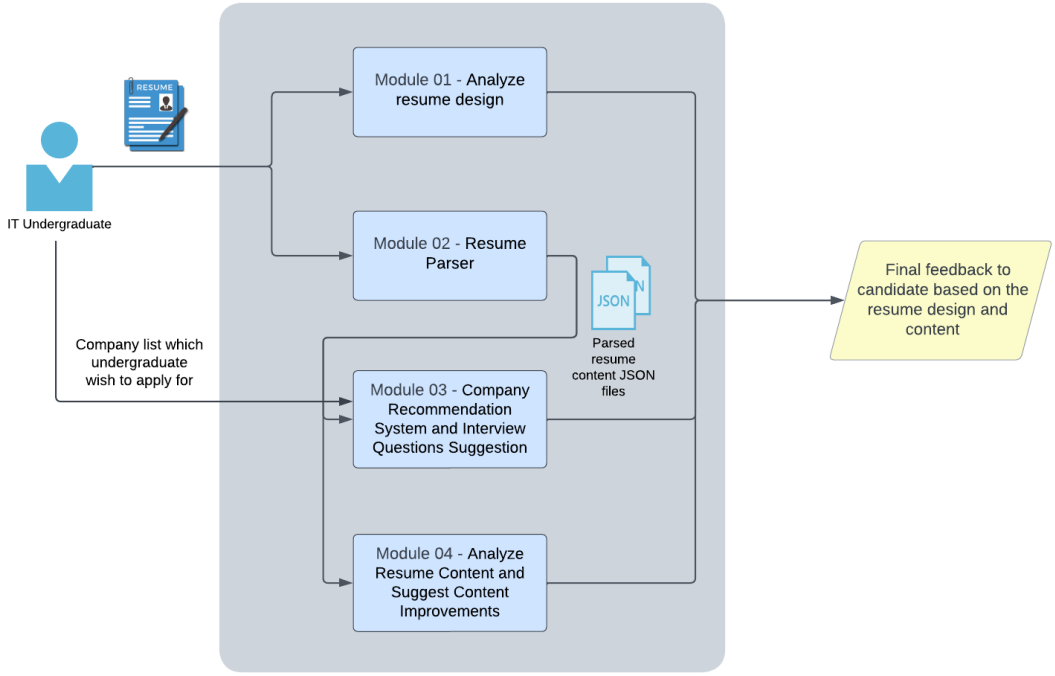


Figure 4.3: Overall process of the proposed system

Figure 4.3 describes all the modules and the overall process of the proposed system. System consists of four individual modules and each individual module is briefly explained in the following subsections.

4.2.5.1 Module 1 - Analyzing Resume Design

Due to the inexperience and lack of knowledge on the appropriate design to be used when preparing a successful resume, undergraduates and fresh graduates face difficulties in designing the resume. Selection of bad color combinations, inappropriate profile image, and inappropriate layouts will reduce the chances of a resume to be shortlisted. Addressing these issues, multiple machine learning models would be deployed in the system to give proper feedback about the resume design.

Mathematical algorithms-based edge and line detection techniques, custom built algorithms to extract dominant colors and text colors in the resume, and deep learning based and traditional machine learning based regression algorithms are used as the main approaches in this module.

4.2.5.2 Module 2 - Resume Parser

As inputs for the other modules in the system, it is required to extract all the text content of resumes. Since resumes are highly structured documents, it is not enough just extracting the content but need to be aware about the layout as well. But through research it was discovered that there doesn't currently exist any system which is capable of extracting section-based content as a whole. Rather, what the existing systems do is extract only prominent phrases from the entire section. but such an extraction will not be suitable for the system.

To tackle the aforementioned issue, a layout-aware extraction technique was implemented in this module. Furthermore, a multiclass classification algorithm which can predict the most appropriate section given the content, is used to optimize the performance of layout-aware extraction.

Additionally, a custom Named Entity Recognition (NER) component is also included which is built using the spaCy library. The NER component was instructed to identify various important entities included in resumes, such as names of the candidate, their interests, personal skills, their contact information, and their email addresses. Further, regular expressions and metadata extraction tools are also used in the system to get important entities. In addition to that, an effective algorithm is used to identify the distinct sections using spaces between bounding boxes around words.

Finally, two different JSON files representing the extracted text content from the resume parsing module will be produced to other modules for further processing.

4.2.5.3 Module 3 - Company Recommendation System and Interview Questions Suggestion

The undergraduates and fresh graduates can be confused as to which companies to apply for when applying for job postings. Sometimes they can be applying for companies which do not match their resume contents and have a very low chance of being selected as well as not applying for companies for which there is a high chance of being shortlisted according to their resume content.

For implementing the company recommendation system, three approaches were tried out. As the first approach, the entire text content of the resume was analyzed to make

predictions. In the second approach, a few selected key features namely GPA, technical skills, and project keywords were analyzed, and their combined results were used to make the predictions. In the third approach the GPA value and existence and the nature of the resume in terms of different sections in a resume were analyzed.

Further, an interview question suggestion system was implemented under this module. The technical skills in a resume are captured by using the NER developed in module 04 and that list of technical skills is used to generate relevant questions by using an existing generative model.

4.2.5.4 Module 4 - Analyze Resume Content and Suggest Content Improvements

The target users of our system have lesser knowledge on how to properly write the content of the resume as this would be most likely their first experience in building a professional resume. Also, they could sometimes mistakenly add important sections to the resume and make grammar mistakes when preparing the resume.

To tackle this issue, a model was implemented to give a score for each section content based on the quality of content. When considering the process of scoring the content quality, it cannot be addressed as a classification problem since it may get continuous values. Hence, a regression model can be used to give a score for the content quality which would be more accurate. Further, a rule-based approach is used to suggest missing sections.

4.3 Summary

As discussed in the above section, the proposed system is to assist the undergraduates to get reviewed their resumes and get feedback about the resume prior to applying to companies. Creating required datasets for the training of the models was a crucial part. Hence, expert knowledge from the industry experts was used to create proper datasets. Based on the knowledge gained through research and related works, and studying the dataset, multiple approaches were adapted for the development of modules.

Chapter 5 - Analysis and Design

5.1 Introduction

This chapter contains a detailed analysis of the system design proposed by the research. Further, the overall system and each individual module along with the module interactions are elaborated using high-level design diagrams.

5.2 System Overview

The system proposed by this research is implemented under four distinct yet interrelated modules as below.

1. Module 1 - Analyzing resume Design
2. Module 2 - Resume Parser
3. Module 3 - Company Recommendation System and Interview Questions Suggestion
4. Module 4 - Analyze resume Content and Suggest Content Improvements

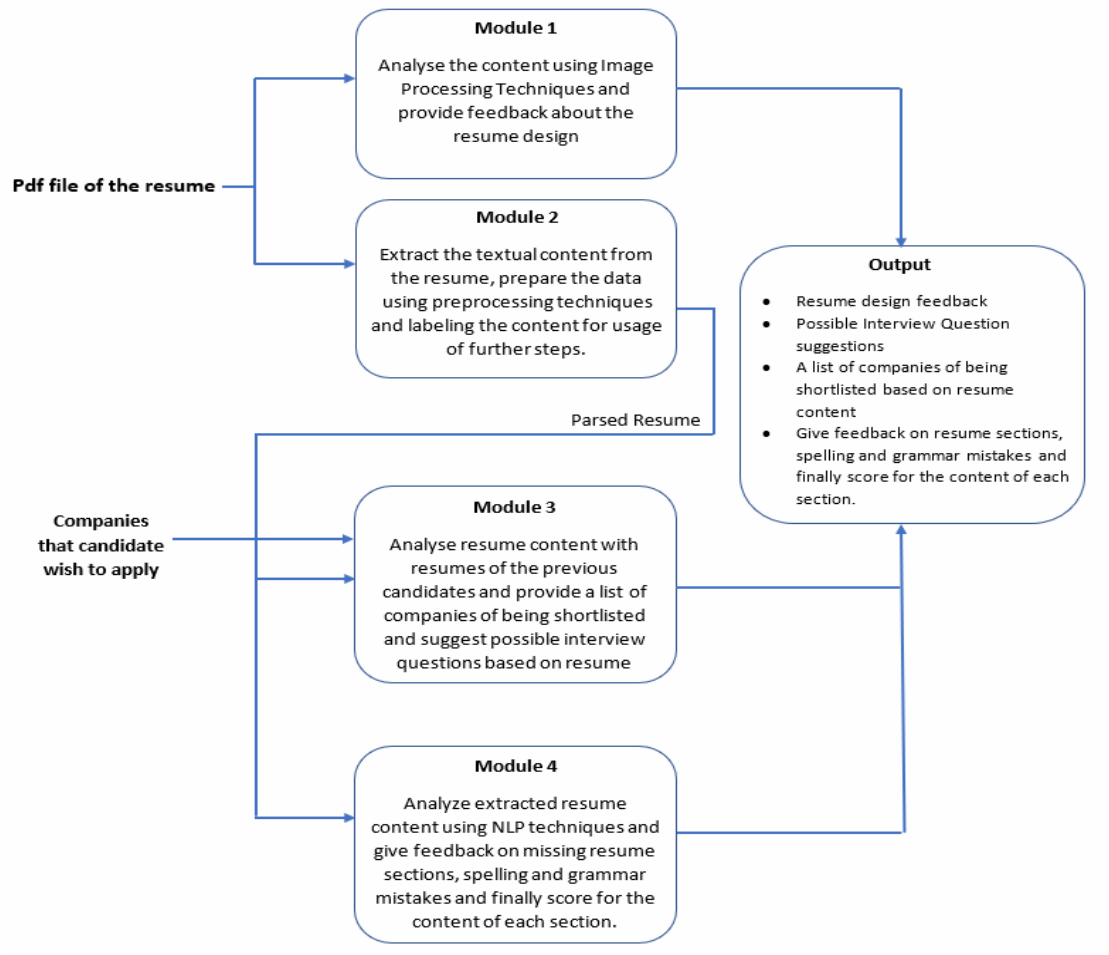


Figure 5.1: High level architecture of the proposed system

Figure 5.1 shows the high-level architecture of the proposed system. It clearly shows how each module interconnected with other modules.

The first module focuses on the design aspects of the resume such as the color combinations and it will output feedback on the overall design. This module stands independent from the other modules in the system since it will not take any inputs from other modules nor will provide an output to them. Module two is mainly performs the text extraction and labeling of the content to extract section-wise textual content. This module will be providing outputs for both module three and module four. The third module of this system is focused on generating a list of companies for which a given resume has the highest possibility of being shortlisted. This will only take into consideration the resume content which is taken as an input from module two. The fourth and final module of the proposed system aims to analyze the extracted text content to output a score for each section based on the content quality.

As the final output of the overall system, the outputs of the first, third, and fourth modules will be output through the platform.

5.2.1 Module 1 - Analyzing Resume Design

When considering resume screening there are two main things that a HR personnel in the industry mainly focus on,

1. Visual features of resume (Design of the resume)
2. The richness of the content

Although the richness of the content is the most important factor which will impact on getting shortlisted through the shortlisting process, the visual features of the resume also play a huge role in this regard. When the resume of the applicant has a good design, better color combination, originality, and easily understandable format, it will make that resume stand out among the other resumes, forcing the person who shortlists the resumes to have a more positive impression about the candidate.

Considering and utilizing that fact, this module is implemented to predict what will be the opinion regarding the design of the resume of the person who shortlists the resumes. Figure 5.2 shows how all models and components in the module interconnected with each other.

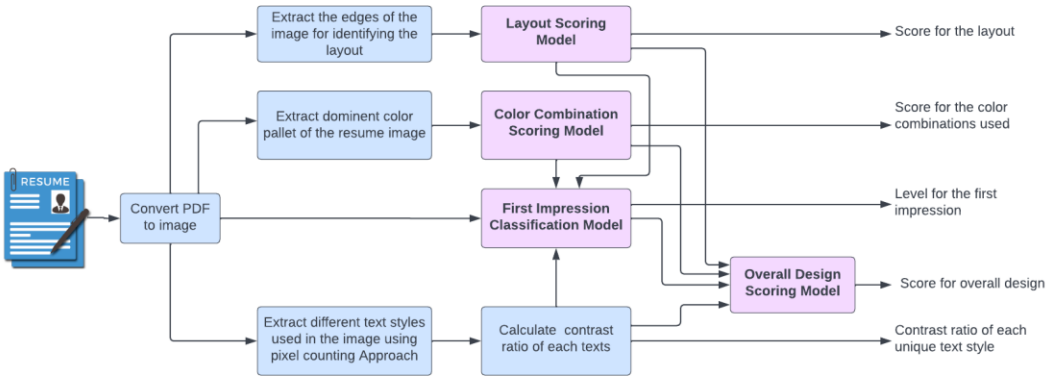


Figure 5.2: Module architecture for analyzing resume design

First, a candidate who needs to analyze his resume design uploads his resume. As the first step of the process, resume pages will be converted into an image for further processing. After the conversion, image-converted resumes are fed into four individual sub-modules for further processing and analyzing.

The first sub module is concerned with extracting the contrast ratios of different text styles that have been used in the resume. An algorithm which uses a color pixel counting approach is used for this, along with an Optical Character Recognition (OCR) task to identify the individual text blocks and its boundary boxes. Using the extracted background color and foreground color of each text block, contrast ratio is calculated and output. This output is,

1. Used as an input for the First Impression scoring model
2. Directly shown as an output to the user

The second sub module is used for extracting the dominant color palette of the resume design. This color palette is extracted based on the same algorithm used in the extracting colors of the first submodule. Then the extracted color palette is fed into the color combination scoring model and will be output a score for the color combinations.

The third submodule is used for generating a score value for the layout by extracting layout using edge detection techniques. Using the extracted edges and expert score value for the layout, this model is trained. Using all the outputs from contrast ratio, layout score and color combination score, First Impression level is classified as

excellent, average, or poor. Finally using these all outputs score value for overall design is calculated.

5.2.2 Module 2 - Resume Parser

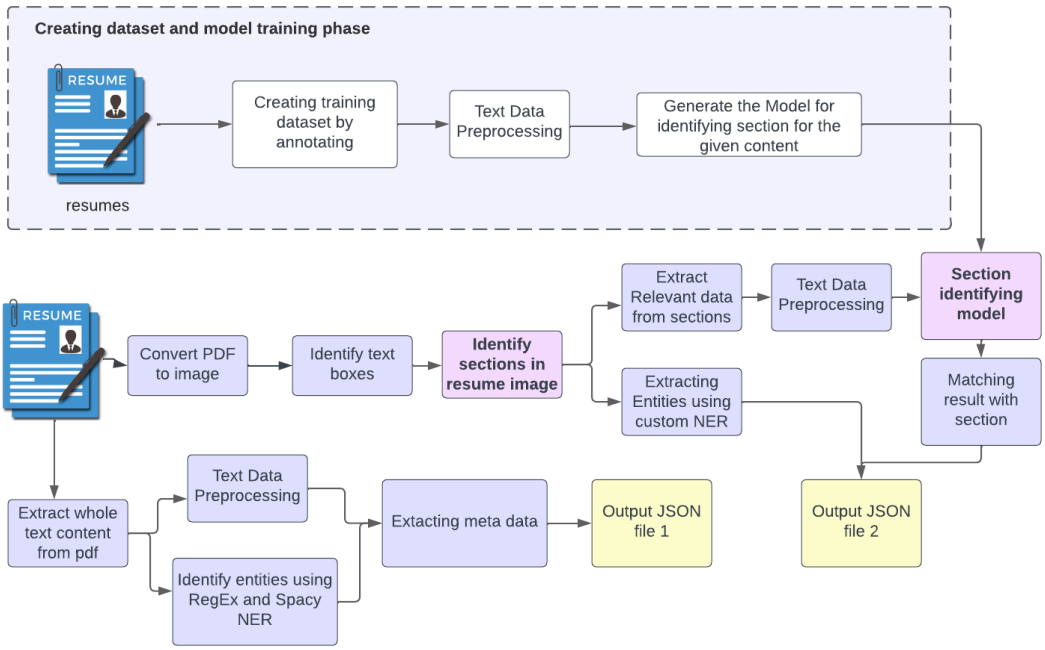


Figure 5.3: Module architecture for resume parser

When considering a resume, it is a highly structured document which would have different types of layouts. In regular text extraction methods, they extract all the text content without considering the layouts and the positions of the text. Therefore, there should be a method which can extract text content from the resumes according to the sections. Further, section-wise contents are required to implement the other modules in the proposed system.

As shown in Figure 5.3, in the training phase of this module, a model is implemented which can predict the section for a given content. A dataset which includes annotated text content and sections is used to train this model. This is a multiclass classification problem since there were several classes to predict in the current problem. As the first step in implementing the model for this multiclass classification problem, the extracted text content has been preprocessed using the main text preprocess techniques and then used the Support Vector Machine algorithm with the SGDClassifier, which is a supervised learning algorithm that has previously shown higher accuracy levels for

similar problems to build the model. This approach is used in this module to predict the section of a given content.

After implementing the section predicting model, in order to accurately predict the section of a given text in the resume, the system needs to detect the layouts and differentiate between paragraphs. To achieve this, an effective algorithm has been used to utilize the spacing between word boxes in the resume document. By analyzing the spaces between the word boxes, the system will identify natural breaks in the text, indicating the presence of paragraph boundaries.

The above algorithm considered the spaces between consecutive word boxes and applied a threshold value to determine whether a space indicated the end of a paragraph. If the resulting space exceeded the threshold value, it was considered significant enough to indicate a paragraph break. Based on this analysis, the text content was divided into different paragraphs.

In the prediction phase of this module, users can upload their resume as a pdf document and from the pre-trained model and the layout detection algorithms, the resume is divided into distinct sections and then the system extracts all the text content for each section. Further, extracted text content will be pre-processed and detects the relevant section from the section predicting model. Additionally, the pre-trained custom NER model will predict significant entities from the resume.

As the output of this module, the extracted text content from the multiclass classification model is outputted as two separate JSON files. The first JSON file will contain significant entities which are extracted from the custom NER, custom regular expressions and the data which are extracted from the metadata extraction process such as personal accounts URLs, emails, contact numbers, other URLs and raw text content which is not divided into sections. This structured data can be used by other modules for further analysis and processing. The second JSON file will include section wise extracted data which are extracted by the multiclass classification model and the layouts detecting algorithm and the system gives the section order as well. By providing these two JSON files as outputs, our system enables seamless integration with other modules, allowing for comprehensive resume analysis, scoring, and recommendation functionalities.

5.2.3 Module 3 - Company recommendation system and Interview questions suggestion

The first job placement or the first internship placement can be a huge contributing factor for a person's future career, as in most cases this would be the first industry experience that they would be getting. Hence securing the best possible opportunity out there and the suitability of a candidate for their selected organization/placement plays a huge role in modeling their career path. Having a bad first experience can often demotivate a person and will not be able to perform their best. So, it's essential for a candidate to be aware of what companies and opportunities out there are best suited for them in order to decide which companies to apply for or to get an idea about if their intended company and their resume matches. The main aim of implementing this module is to aid the candidates in this process.

This module intends to address this issue is by researching the existence of a relationship between the content in a resume and the possibility of getting shortlisted in the first resume screening stage in the screening process of different companies. To carry out this research, the context of this module is limited to a selected few reputed Sri Lankan IT organization. The design analysis diagram for implementing this module is depicted by Figure 5.4.

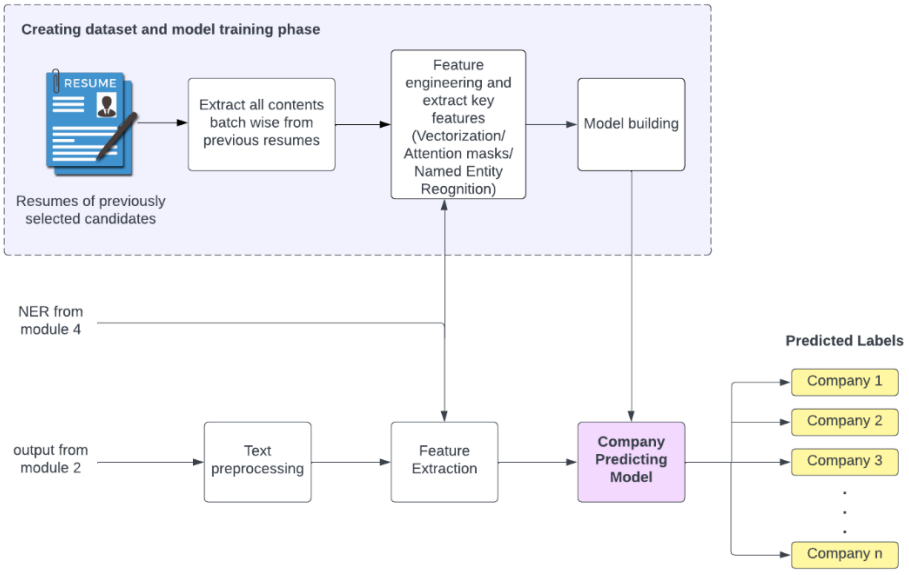


Figure 5.4: Module architecture for analyzing company recommendation system and interview questions suggestion

Once the resume content is extracted by module two from the uploaded resume to the system, it is taken as the main input to this module. The input text data is preprocessed such that it is suitable for feature extraction purposes and to be fed into the company predicting model. Also, the NER from module 04 will be utilized to extract features. What the company predicting model does is, once the resume content is fed into the model, it will predict and output a list of companies for which the resume matches the most.

After conducting research on how to derive a relationship among the resume content and the list of organizations for which that particular resume has a high possibility of getting shortlisted, it was found that this can be achieved by approaching this scenario as a multi-label classification problem.

To implement this classification module, the company prediction is tried out under three different approaches as below.

1. Approach 01
    - Under this approach, prediction of companies which have the highest possibility of a candidate getting shortlisted is tried to predict by taking into consideration the whole text content of the resume.

2. Approach 02
    - By the discussions had with the industry experts, few key features of the resumes were identified which were primarily considered when shortlisting resumes. Then for each of these key features, the company prediction is carried out and a combined output of their results were taken as the final prediction.

3. Approach 03
    - Through the third approach the keyword frequencies of different sections in the entire resume are taken into account along with the GPA value to make the predictions.

For training the different models for our task, it is required to feed the model with appropriate training datasets containing textual content of resumes of previously selected candidates. So appropriate datasets were prepared by extracting text content of all historical resumes gathered in the initial data collection phase and by the details of

what companies each resume owner had received first calls from. The extracted content is then pre-processed, and all the important features are extracted. The prepared dataset is split to train and test sets for the purposes of training the modules and evaluating them. Additionally, a list of interview questions are being displayed for the user through the portal by considering the technical skills they have mentioned in their resume.

5.2.4 Module 4 - Analyze resume content and suggest content improvements

When it comes to shortlisting resumes, content quality holds a prominent value. Resume content builds the first impression of the image of a person. Hence, good quality content could build a better image of a person. Even though a person who is more talented and well-suitable for a specific job position, that person may not be selected when applying if he/she doesn't have a resume with good quality content. On the other hand, a person with less talent but a good resume with quality content may be selected for the same position. Therefore, creating a resume with quality content is required when it comes to applying for a job.

Due to the lack of experience and knowledge, undergraduates fail to create good resumes with quality content. They often make mistakes when adding content to their resume. To assist undergraduates to overcome the issue regarding the quality of the resume content, the proposed system includes this module. This module outputs a score for major sections by considering the quality of the content of the section. Additionally, spelling and grammar mistakes, important missing sections or content and content improvement suggestions will be pointed out by this module.

When it comes to the development of this module, two major approaches were considered. In the first approach, section-wise extracted resume content as whole and the section-wise score were fed into the module for future predictions, while in the second approach, it was fed a dataset with section-wise extracted specific features and section-wise score to the module to predict the scores for unseen resumes. The design plan for the module is denoted by Figure 5.5.

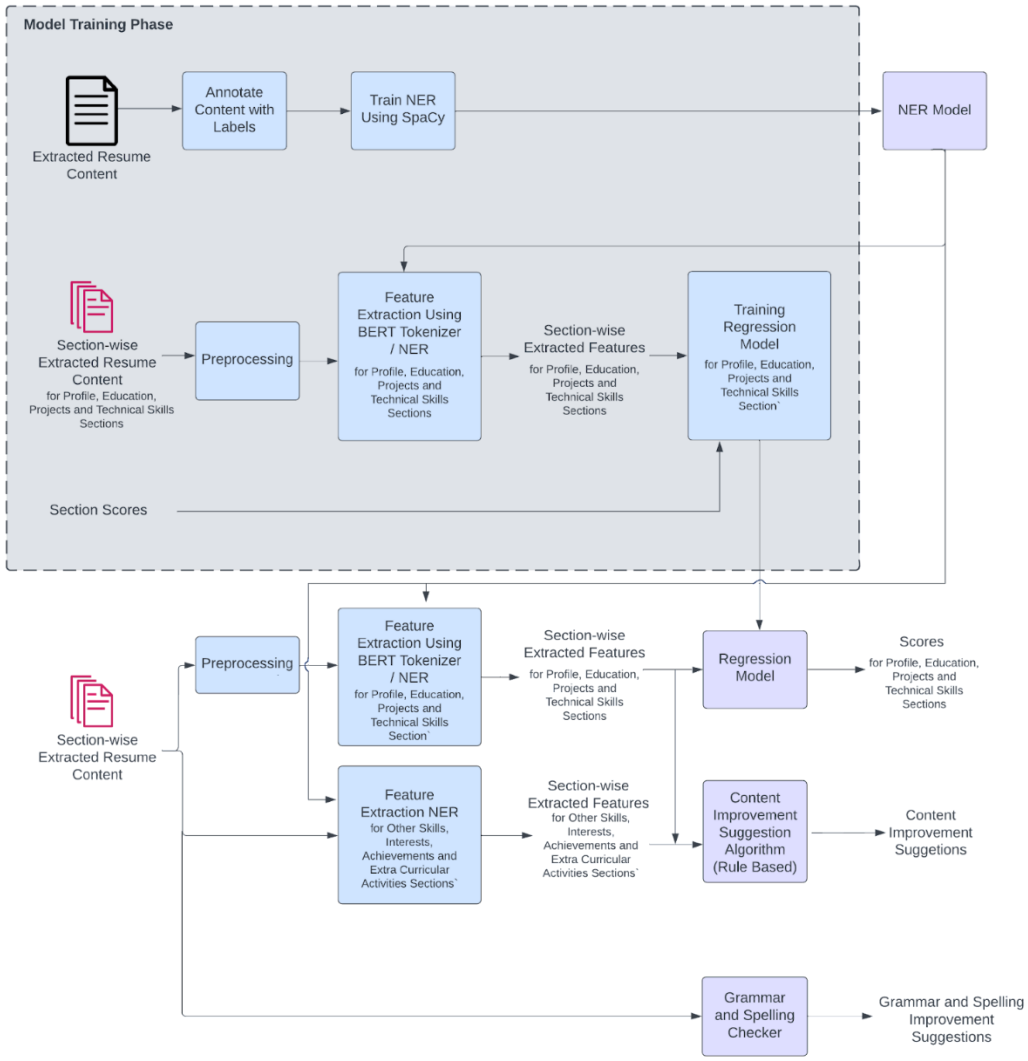


Figure 5.5: Module architecture for analyze resume content and suggest content improvements

As the first step, extracted resume content is preprocessed with NLP techniques for further process. A separate NER model was implemented for feature extraction purposes. Then the feature extraction was performed section-wise to get the features of each section which plays a major role in training of the module. Then using extracted features and section score data, the regression model was trained to predict the score for future resume contents. In the final system, the input of the module is the output data from module 2. An existing model was utilized for spelling and grammar checking. Section scores, grammar and spelling improvement suggestions and missing sections are the final outputs of the module.

5.3 Summary

As stated above, this chapter describes the proposed system by elaborating the individual modules within the system, their design approaches, and interactions between different modules. The different machine-learning techniques used in each module to tackle the issues identified are further discussed in this chapter.

Chapter 6 - Implementation

6.1 Introduction

This chapter describes the module-wise implementations carried out through this study. Here, the implementations are described in detail using code snippets, graphs, and other images as appropriate.

6.2 Module Implementation

The following subsections describes implementation approaches and methods which were taken to fulfill the requirement and designated module design in chapter 5.

6.2.1 Module 1 - Analyzing Resume Design

When considering the Resume Design Analysis module, it consists of five subcomponents as shown in Figure 5.2.

1. Text contrast ratio identification algorithm
2. Color combination extraction and scoring model
3. Layout extraction and scoring model
4. First Impression classification model
5. Overall design scoring model

First, to feed all these five subcomponents, PDF to image conversion has been done to the uploaded Resume. Then the converted image is fed into all five subcomponents.

```
from pdf2image import convert_from_path
import numpy as np

cv_name = 'Sample Resume.pdf'
dpi = 250
pages = convert_from_path(cv_name, dpi=250)
pages = [np.asarray(page) for page in pages]
```

Figure 6.1: Converting the PDF of the resume to images

As shown in Figure 6.1 using the ‘convert_from_path’ method in the ‘pdf2image’ library this task is achieved. For further processing, extracted images are converted into numpy array.

6.2.1.1 Text contrast ratio identification algorithm

When considering a digital document, the text contrast ratio is a key factor which will enable the evaluators to evaluate and grasp the key points in your resume at once. Therefore, when preparing a digital document like a resume, it is better to have a good track on text contrast ratios used in the applicant's own resume.

6.2.1.1.1 Identifying text blocks for calculating text contrast ratio

For text contrast ratio calculation, the first task is to mark the text regions which need contrast ratio calculation. This is also called Region Of Interest (ROI). For this task two methods were tried.

a. Mark the regions using thresholding and finding contours

Thresholding is a special mechanism used in identifying and segmenting areas in images which has significant differences with the background. To use that mechanism at first, BGR to RGB color conversion is applied. BGR color mode is the default color model that image gets when it is opened through OpenCV. Even though pdf-to-image gives an RGB image, to view in the OpenCV window we have to apply the RGB-to-BGR conversion for the image. Comparison between RGB and BGR colors modes are shown below (Figure 6.2).

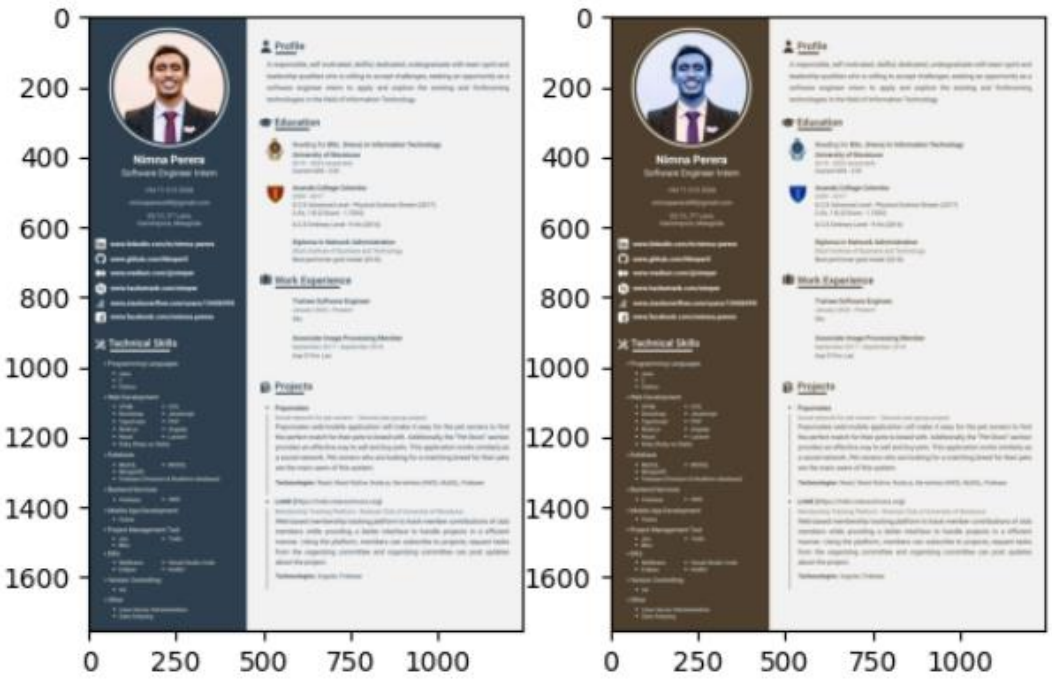


Figure 6.2: Resume viewed using RGB and BGR color modes respectively

After that, before applying the adaptive thresholding, Gaussian blur is applied to smooth and denoise the image. Otherwise, noise will be a cause for the low accuracy of the output. Then adaptive thresholding is applied to the PDF image following a morphological operation called dilate. Dilate operation has the ability to fill the gaps between two letters or words and mark it as a whole word or a line respectively. Already implemented methods in the OpenCV library are used for all these image operations. Finally, contours are being extracted to identify the boundary boxes of the text areas.

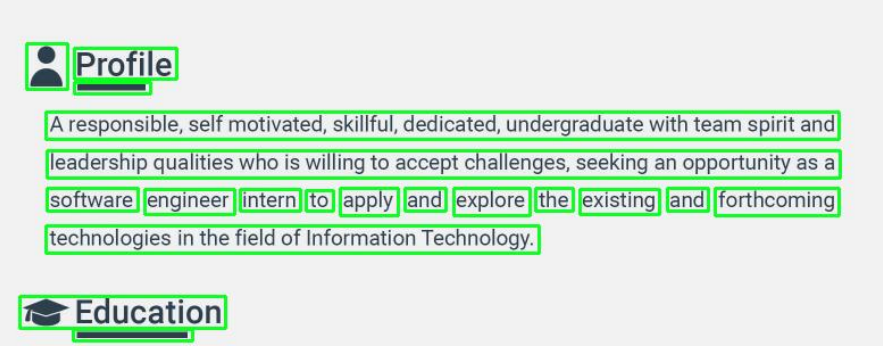


Figure 6.3: Identified boundary boxes

As shown in Figure 6.3, even though it can identify text regions, there is a huge issue in this method. The issue is this method is not aware whether it is a text region or not. According to the above Figure 6.3, the icon in the profile section and its underline is also identified as text since it has significant differences with background. It only looks for differences with backgrounds and marks it as a ROI. The other major problem of using this approach is it tends to mark text segments along with other areas in the image (Figure 6.4) which will ultimately force the text contrast ratio calculation algorithm to return a false prediction.

Figure 6.4: Instead of marking the text region it marks the whole image segmentation

As mentioned above, these two issues will make the algorithm more computationally expensive and inaccurate. Therefore, this method is rejected for identifying individual text blocks.

b. Mark the regions using Tesseract OCR word boundary boxes

Tesseract OCR is a commonly used Optical Character Recognition tool in the field of image processing. Tesseract is an open-source OCR tool which is maintained by Google. According to [31] using its unique line finding, feature selection and adaptive classifying models, Tesseract is able to show outstanding performance in OCR. In this context, because of its very accurate text boundary-box finding technique, Tesseract is used as the OCR tool for identifying text-regions.

Since it is only needed to identify unique text style, After the boundary-box extraction, boxes with lesser confidence level are removed from the box list. Further filtering rules such as removing symbols only text regions, removing empty text regions, removing texts which are not lengthy enough for the calculation are being done to increase the performance of the algorithm.

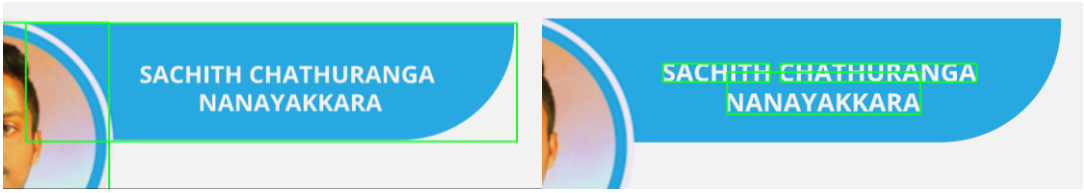


Figure 6.5: Text regions marked using method (a) (thresholding and finding contours) and method (b) (boundary-boxes in tesseract)

As shown in Figure 6.5, it is clearly visible that using method (b) will be more accurate for marking text regions for the contrast ratio calculation algorithm.

6.2.1.1.2 Extracting background color and text color of marked texts

For calculating text contrast ratio, the next task is extracting background color and text color of marked texts.

a. Count each color of the pixel and get the most, second most occurred colors as the background color and text color

Due to this approach being very ineffective for the task, it is rejected. When the ROI is smaller the more pixels will get different shades of the background color (Figure 6.6, Figure 6.7 and Figure 6.8). Then even in the 10th most occurred colors won't contain the actual text color.

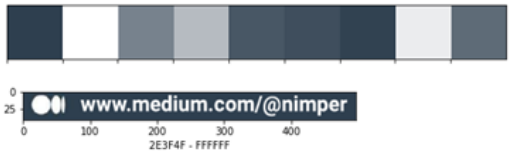


Figure 6.6: Correctly extracted color palette and ROI (ROI width > 450px)

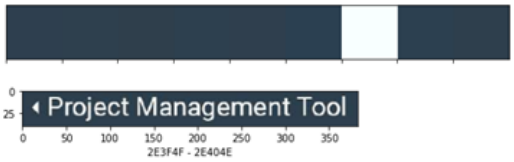


Figure 6.7: Incorrectly extracted color palette and ROI (ROI width < 400px)

Figure 6.8: Incorrectly extracted color palette and ROI (ROI width < 120px)

b. Cluster colors together and extract two clusters

In this method the same kind of colors are being clustered together using KMeans algorithm [32], so that the two dominant colors of the ROI (background color and text color) will be grouped and extracted from the image. After extracting, using a counter to count clustered data points, background color and text color are determined. Some of the extracted background colors and text colors are displayed in the following Figure 6.9.

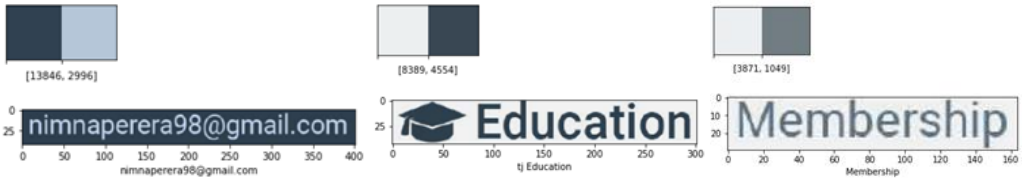


Figure 6.9: Almost correctly extracted background and text colors

The issue of using this method is when there are more variant colors in the background color of an image, it gets averaged. This will ultimately result in contrast ratio

prediction with more Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

The graph below (Figure 6.10) shows how the RMSE and MAE vary for different DPI (Dots Per Inch) values used for converting a PDF page to an image. According to its observation 280 is selected as the DPI value for PDF-to-image conversion.

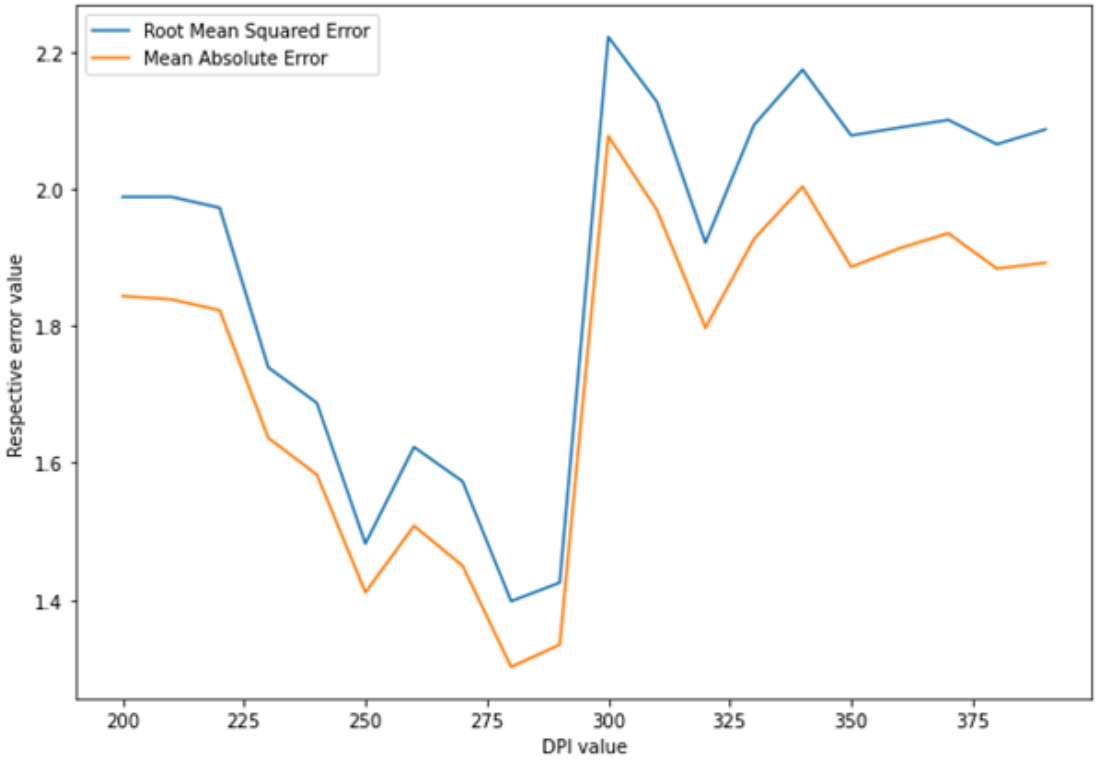


Figure 6.10: RMSE and MAE method (b) and method (c) against different DPI values used in PDF-to-image conversion

6.2.1.1.3 Calculate text contrast ratio for unique styled texts

Final text contrast ratio calculation is done using the extracted color method. To distinguish unique text styles in resume, cross checking over found text styles has been done and same styled texts are dropped in that process. Figure 6.11 shows the final output after calculating contrast ratio over the complete resume.

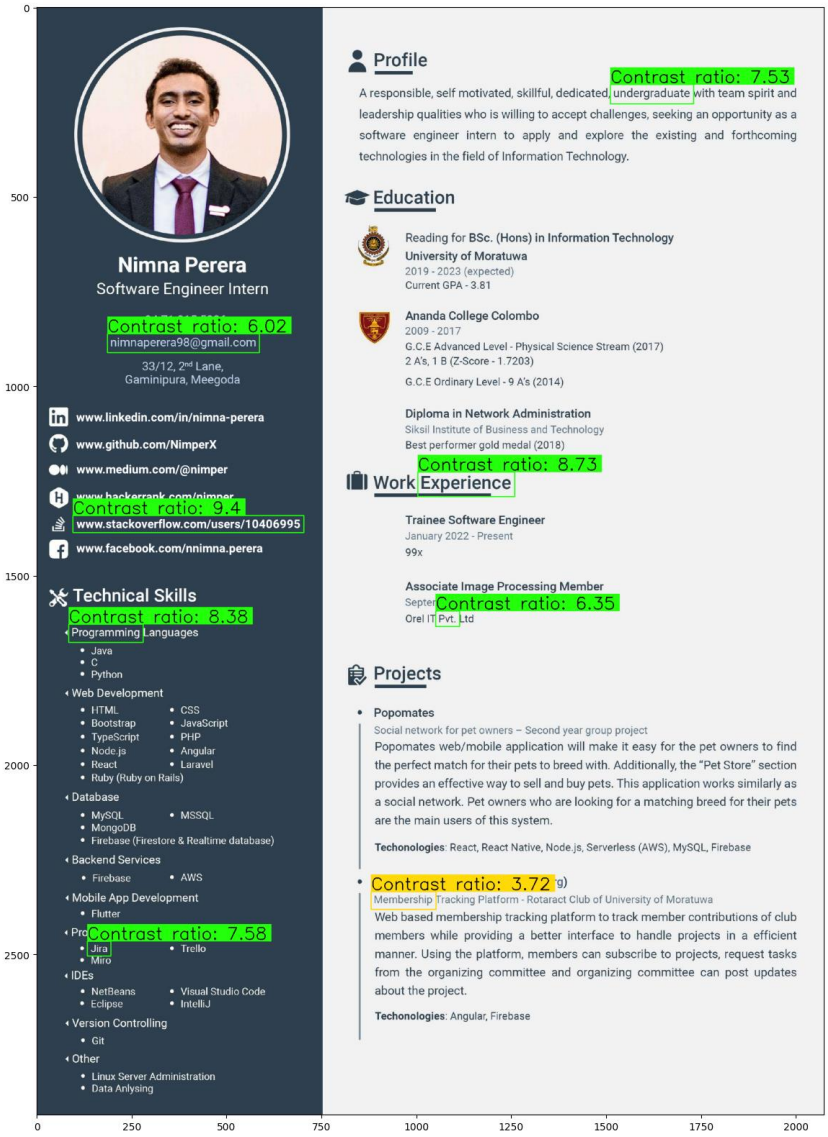


Figure 6.11: Final output after calculating contrast ratio over the complete resume

6.2.1.2 Color combination scoring model

6.2.1.2.1 Dominant color palette extraction

For extracting the dominant color palette in addition to the color clustering method used in text contrast ratio algorithm, a modified color occurrence approach was tried out. In this algorithm color pixels are counted, and shades of the same color were identified using the Euclidean distance between colors. Finally shades of the same color were dropped out to output the final color palette.

Figure 6.12: Dominant colors palettes extracted using modified color occurrence algorithm and color clustering respectively

Since both algorithms output almost the same outputs (Figure 6.12), the best algorithm was determined by comparing the performance of the two algorithms.

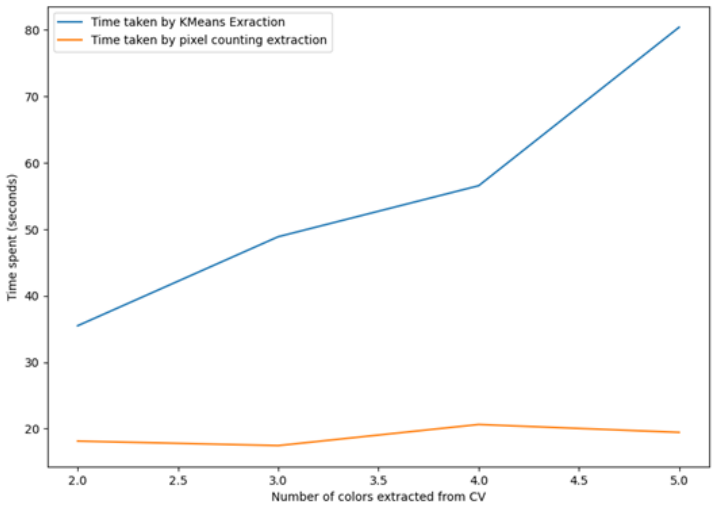


Figure 6.13: time taken to extract 2, 3, 4, and 5 colors pallets from 01 page resumes using two methods

Since according to Figure 6.13, the modified color occurrence approach shows more performance for the designated task, it is selected as the method for extracting the dominant color palette of resumes.

6.2.1.2.2 Color combination scoring prediction

After extracting the colour palette, the feature engineering is applied for building the scoring model. According to [33] intensity of a colour and contrast ratio is mainly affected for human readability. Therefore, the grey intensity of each colour, highest and lowest contrast ratios are extracted as the features. The correlation between the extracted feature and the colour combination scores is as follows (Figure 6.14).

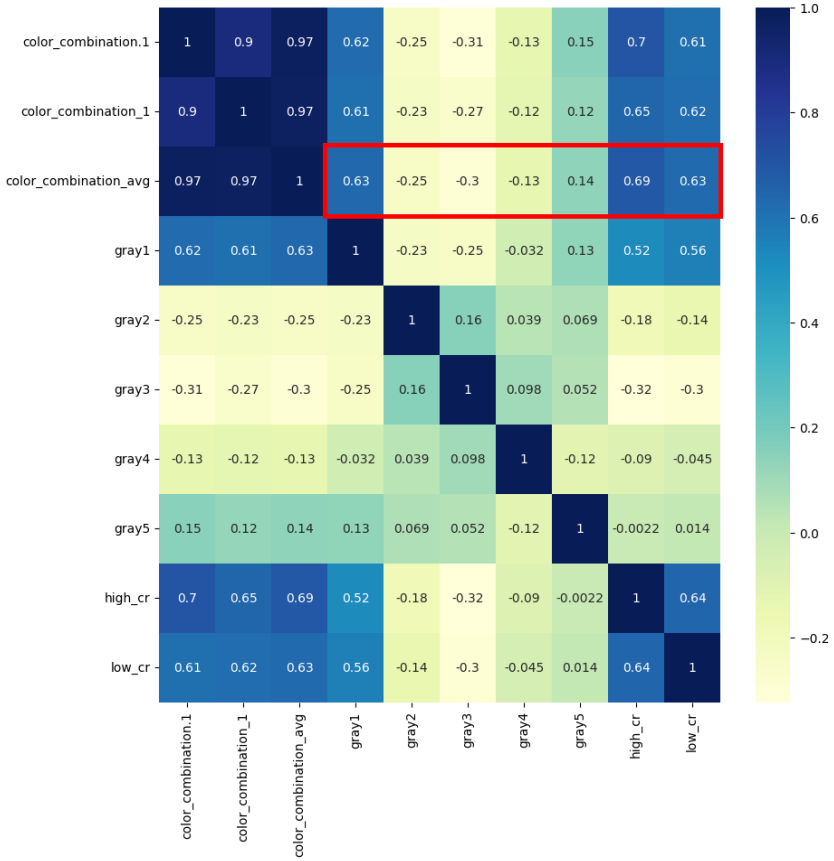


Figure 6.14: Correlation between extracted features and color combination score

After the feature engineering, five algorithms were trained for the prediction of the scores, which is a Keras Artificial Neural Network, LinearRegressor, Decision Tree regressor, Support Vector Regressor and a Random Forest regressor. The Keras Artificial neural network is consisted with 15 input units, 100 fully connected Dense units, 200 Dense units, 50 Dense units and 1 Dense unit respectively. Between each

dense layers Rectified Linear Unit (ReLU) activation function and at the end of the network, Sigmoid activation functions are used. Network is trained for 500 epochs. The code segment for assembling the neural network and variation of the loss values and error metric graph plotted against the number of epochs are illustrated in Figure 6.15 and Figure 6.16 respectively.

```
def assemble_model(input_layers):
  inputs = Input(shape=(input_layers,))

  nn = Dense(100)(inputs)
  nn = Activation('relu')(nn)
  nn = BatchNormalization()(nn)
  nn = Dropout(0.2)(nn)
  nn = Dense(200)(nn)
  nn = Activation('relu')(nn)
  nn = BatchNormalization()(nn)
  nn = Dropout(0.2)(nn)
  nn = Dense(50)(nn)
  nn = Activation('relu')(nn)
  nn = BatchNormalization()(nn)
  nn = Dropout(0.2)(nn)
  nn = Dense(1)(nn)
  nn = Activation('sigmoid', name='color_combination_output')(nn)

  model = Model(inputs=inputs, outputs=[nn], name='color_combination_score_model')
  return model
```

Figure 6.15: The code segment for assembling the neural network

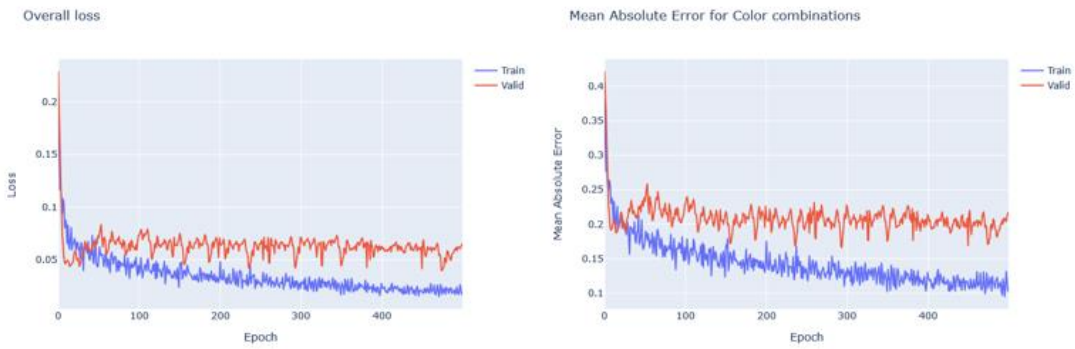


Figure 6.16: Variation of the loss values and error metric graph plotted against the number of epochs

6.2.1.3 Layout extraction and scoring model

6.2.1.3.1 Extract the layout using edge detection

When considering the first impression of a resume, layout of the resume plays a huge role. Humans tend to perceive according to their own cognitive models. Cognitive models are influenced with the previous experiences of humans[34]. Therefore, humans usually find the previously reviewed and liked layouts more attractive than the new

layouts which they have not interacted with. By following this concept, this model has been built for categorizing the layout of the resume for commonly used layouts and based on that classification, feedback will be given for the resume.

First the edges of the resume image which has been generated in the first module is extracted using Canny's edge detection[35]. After the extraction of the edges, extracted edges are fed to the “generalized Hough transform” algorithm for finding the lines of the identified edges[36]. These edges later on used for finding the class of the resumes that resume belongs to. For better performance and better results, neither horizontal nor vertical lines and lines which do not have specific minimum length are filtered out. After filtering out the lines, the XY positions of the lines are fed to an artificial neural network for classifying the class. The full model architecture of this model is depicted in Figure 6.17. This neural network is trained for 1000 epochs and the model has converged after 400 epochs.

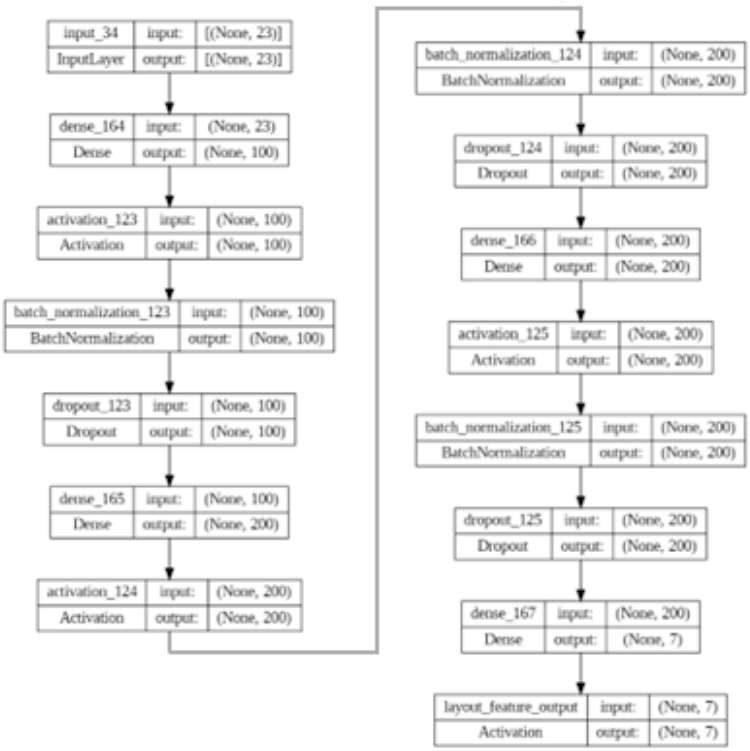


Figure 6.17: Model architecture of the layout classification neural network

6.2.1.3.2 Layout scoring model

Since layout scoring is also a regression task which predicts a number between 0 to 10, the same algorithms used in color combination scoring (chapter 6.2.1.2.2) have been utilized. The loss values and error metric graph plotted against the number of epochs

are illustrated in Figure 6.18. For this training, 1000 epochs have been used and around 200 epochs, the neural network has been converged. Other traditional regression algorithms are also trained and evaluated accordingly.

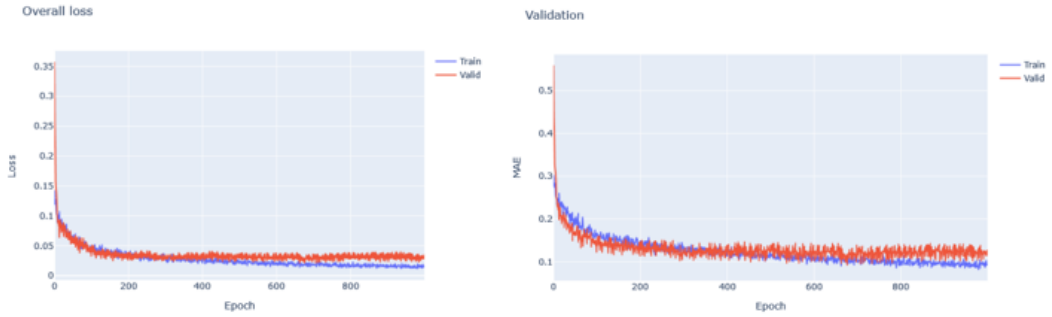


Figure 6.18: Loss graph and validation error graph against number of epochs trained

6.2.1.4 First Impression classification model

When considering the first impression, it is primarily segmented into three main levels, which is excellent, average and poor. For predicting these levels, the extracted contrast ratios, predicted color combination score and predicted layout score are used. Therefore, this model is inherently reliant on the three aforementioned models and algorithms. The correlation between the aforementioned score values and first impression classes are depicted in Figure 6.19.

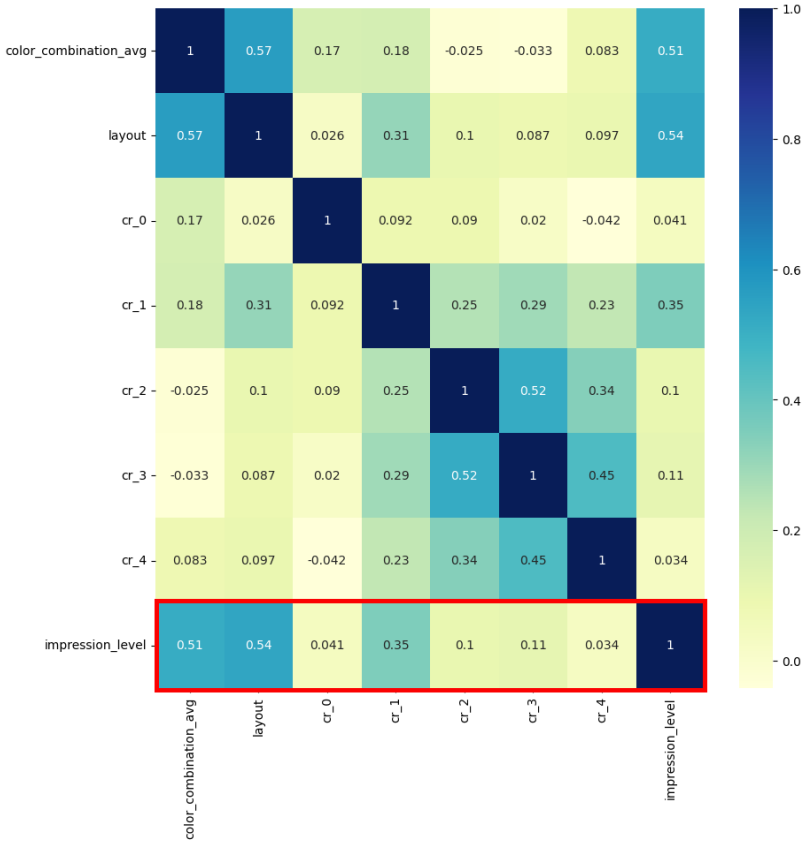


Figure 6.19: The correlation between first impression level and other extracted values

As depicted in Figure 6.19, the impression level has more correlation between color combination and layout score. Finally, using the dataset that we collected in chapter 4.2.1 a Gaussian naive bayes has been used for training the model for the stated classification task.

6.2.1.5 Overall design scoring model

Similar to the preceding model, the overall design scoring model is dependent upon the extracted contrast ratios, color combination score, layout score, and additionally first impression class is also used for predicting the overall design score. The correlation between above mentioned values and overall design score that HR personnel given are shown below in Figure 6.20.

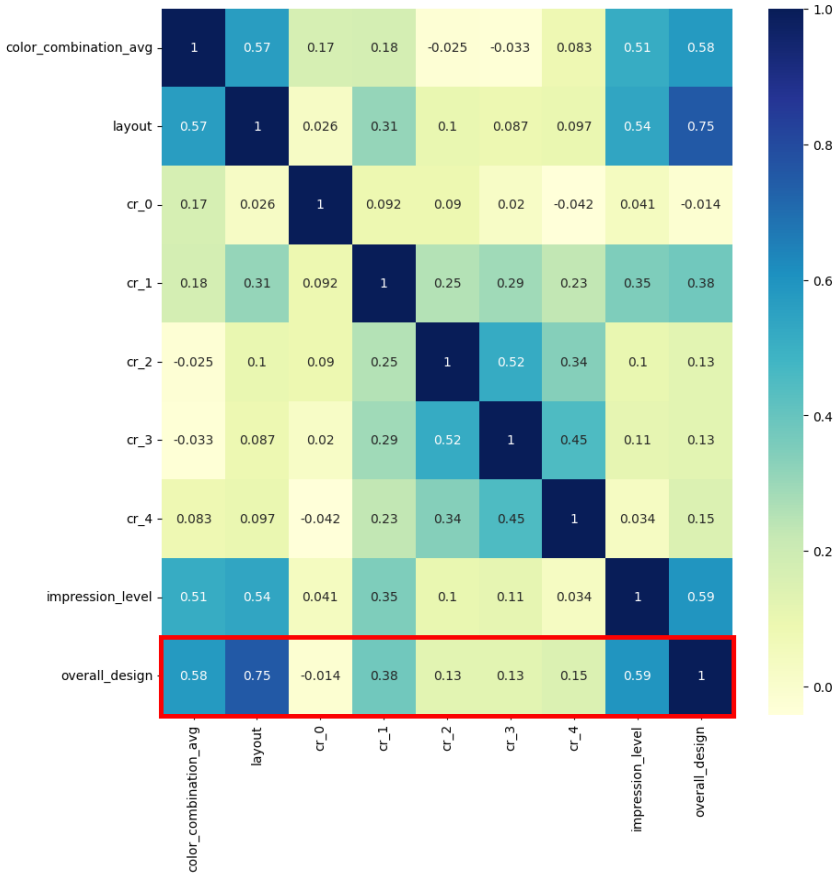


Figure 6.20: The correlation between first impression level and other extracted values

According to the depicted correlation graph, it is clearly visible that overall design score correlates more with the color combination score, layout score and impression level which is an obvious fact. Since the correlation that has in-between the independent variables and dependent variable, only traditional machine learning algorithms are used for the prediction model training. According to that, a Linear regression model, a Decision Tree regressor, a Support Vector Regressor and a Random Forest regressor have been trained for the stated regression task. After the training phase, by comparing the model using error metrics and other metrics such as R2 score, the best model for predicting the overall design score was selected. The code segment for training the stated algorithms and evaluating the models is depicted in Figure 6.21.

```
models = [LinearRegression(), DecisionTreeRegressor(), SVR(), RandomForestRegressor()]

for m in models:
    print(m)
    m.fit(X_train, y_train)

for m in models:
    overall_score = m.predict(X_test)
    print("")
    print("============================================================")
    print("")
    print("Model\t\t", m)
    print("MAE:\t\t", mean_absolute_error(y_test.to_list(), overall_score))
    print("RMSE:\t\t", sqrt(mean_squared_error(y_test.to_list(), overall_score)))
    print("R2 score:\t", r2_score(y_test.to_list(), overall_score))
```

Figure 6.21: The code segment for training and predicting using stated algorithms

6.2.2 Module 2 - Resume Parser

6.2.2.1 Extracting All Text Content, Entities spaCy NER, RegEx and Meta Data from Resume for Other Modules

As the first task of this module, all the text content from the uploaded resume was extracted using python language and the libraries. Further, several important entities such as emails, links and phone numbers also have been extracted using spaCy NER, regular expressions and metadata extraction process.

```
from PyPDF2 import PdfReader
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
import spacy
from spacy.lang.en.stop_words import STOP_WORDS
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.layout import LAParams
from pdfminer.converter import TextConverter
from io import StringIO
from pdfminer.pdfpage import PDFPage
import re

nlp = spacy.load("en_core_web_sm")
stemmer = PorterStemmer()
```

Figure 6.22: Code block for importing libraries

In Figure 6.22, all the necessary libraries were imported. spaCy was used for the NLP tasks while the PdfMiner libraries were used to extract the text content.

```
pdf_doc = nlp(extract_text_from_pdf('11.pdf'))

final_text = str(pdf_doc)

def clean_text(text):
    text = text.lower()
    text = text.replace("\"","")
    text = text.replace("\'","")
    string_encode = text.encode("ascii", "ignore")
    string_decode = string_encode.decode()
    return string_decode

final = clean_text(final_text)

content  = " ".join(final.split())
print(content)

# print("***********************************")
# getting emails using spacy
emails = []
for token in pdf_doc:
    if token.like_email:
        emails.append(str(token))
print(emails)

# # getting links using spacy
otherUrls = []
for token in pdf_doc:
    if token.like_url:
        otherUrls.append(str(token))
print(otherUrls)

#getting phone numbers
phone_numbers = re.findall(r'(?:\+\d{1,3}\s?)?(?:(?:\(?:\d{3}\)\s?)|(?:\d{3})(?:\s|-?))(?:\d{3}(?:\s|-?))(?:\d{4})(?:\s?(?:(?:[E
print(phone_numbers)

# print("***********************************")
```

Figure 6.23: Code block for reading the pdf and extracting details using regex

Figure 6.23 shows the code base for extracting the content from the resume, and then using the spaCy Named Entity Recognition feature and the regex, the system extracted emails, URLs of the applicant's accounts and the phone numbers. Phone numbers can be in several formats and as a result, there is a generic regular expression which can match all identical phone number patterns.

```
#getting accounts links from meta data

import fitz

linkedin = ''
github = ''
stackoverflow = ''
hackerrank = ''
medium = ''

urls = []

with fitz.open('11.pdf') as my_pdf:
    for page_number in range(1, len(my_pdf)+1):
        page = my_pdf[page_number-1]

        for link in page.links():
            if 'uri' in link:
                print(link['uri'])
                urls.append(link['uri'])
                if re.findall(r'http[s]?:\/\/www\.linkedin\.com\S+|www\.linkedin\.com\S+|http[s]?:\/\/linkedin\.com\S+|linkedin.
                    linkedin = link['uri']
                elif re.findall(r'http[s]?:\/\/www\.github\.com\S+|www\.github\.com\S+|http[s]?:\/\/github\.com\S+|github.com\S+
                    github = link['uri']
                elif re.findall(r'http[s]?:\/\/www\.stackoverflow\.com\S+|www\.stackoverflow\.com\S+|http[s]?:\/\/stackoverflow\.
                    stackoverflow = link['uri']
                elif re.findall(r'http[s]?:\/\/www\.hackerrank\.com\S+|www\.hackerrank\.com\S+|http[s]?:\/\/hackerrank\.com\S+|ha
                    hackerrank = link['uri']
                elif re.findall(r'http[s]?:\/\/www\.medium\.com\S+|www\.medium\.com\S+|http[s]?:\/\/medium\.com\S+|medium.com\S+
                    medium = link['uri']
```

Figure 6.24: The code block for extracting meta data from resume

In order to obtain personal accounts links such as LinkedIn, Medium, and others from the resume, the system has a metadata extraction method as coded in Figure 6.24. This method allowed the system to extract relevant metadata associated with the resume document, including any URLs or hyperlinks present within the text. By analyzing the metadata, we were able to identify and extract personal accounts links that might be mentioned in the resume. By employing this metadata extraction method, the system was able to capture personal accounts links that might not be directly accessible through text extraction alone.

```python
import json

# some JSON:
result = {
"emails":emails,
"account&MailLinks":urls,
"github":github,
"linkedin":linkedin,
"stackoverflow":stackoverflow,
"hackerrank":hackerrank,
"medium":medium,
"phone_numbers":phone_numbers,
"content":content,
"otherUrls":finalOtherUrls
}

# parse x:
y = json.dumps(result)

# Writing to sample.json
with open("extracted_data.json", "w") as outfile:
    outfile.write(y)
```

Figure 6.25: Code block for writing the JSON file

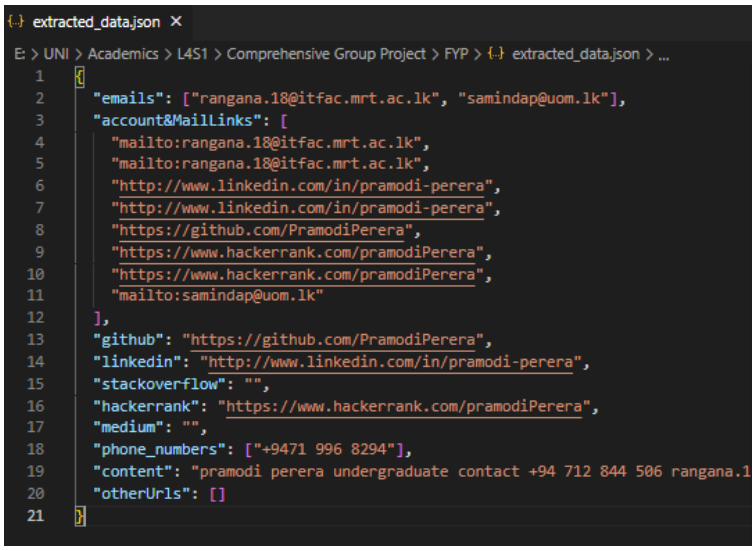


Figure 6.26: output JSON file1

Figure 6.25 and Figure 6.26 show the code block and output of the first task in this module. A JSON format is used to output the retrieved data. This extracted content can be used for the other modules in the proposed system for their implementation process.

6.2.2.2 Creating dataset for the classification model

The main objective of this task is to annotate and create the required dataset for the multiclass classification model which the system needs to predict the section for the given content. Label Studio software was used for the annotation process, and it exports the final annotated data as a coco file with the relevant image files. That coco file includes a JSON file which includes all the annotation details with their positions coordination and the relevant section class label details. Figure 6.27 shows the format of the exported JSON file.

Figure 6.27: Position coordinates of the annotations

After exporting the annotated data from the label studio as a coco file, then the system should read those files to create the dataset. Figure 6.28 shows how to read the relevant image files from the OpenCV library and then output the annotated images with the relevant section class (Figure 6.29).

```
color_map = {
    'Profile':'#FFDF4F',
    'Education':'#0D3730',
    'Projects':'#457598',
    'Work Experience':'#A450DC',
    'Technology/Technical skills':'#66DF34',
    'Awards & Achievements':'#EB00AD',
    'Extra Curricular/ Roles and Responsibilities':'#E44E0C',
    'Interest':'#104A77',
    'Personal Skills':'#6EA221',
    'Refrees':'#FFA39E'
}

for image_id in random.sample(coco.imgs.keys(), 1):
    image_info = coco.imgs[image_id]
    annotations = coco.loadAnns(coco.getAnnIds([image_id]))

    path = COCO_IMG_PATH+'images/'+image_info["file_name"][9:]
    image = cv2.imread(f'{path}')
    layout = load_coco_annotations(annotations, coco)
    #view cv with annotations
    viz = lp.draw_box(image,
              [b.set(id=f'{b.id}/{b.type}') for b in layout],
              color_map=color_map,
              show_element_id=True, id_font_size=16,
              id_text_background_color='black',
              id_text_color='white')
    display(viz)
```

Figure 6.28: Code block for reading the annotations

Figure 6.29: Shows the annotations

After identifying the relevant sections in resumes, then there should be a method to read the text content from each boundary box. Tesseract OCR was used to read the text which were included in boundary boxes. Figure 6.30 shows the code base for reading the annotated data with their classes and writing extracted text content (Figure 6.31) to a csv file.

```
ocr_agent = lp.TesseractAgent(languages='eng')
data = []
count = 0

main_folder = 'All Annotated CVs'
folders = ['Folder 1-3, 18', 'Folder 4-8', 'Folder 9-13', 'Folder 13-17']

for folder in folders:
    COCO_ANNO_PATH = main_folder + '/' + folder + '/result.json'
    COCO_IMG_PATH  = main_folder + '/' + folder + '/'

    coco = COCO(COCO_ANNO_PATH)

    for image_id in coco.imgs.keys():
        image_info = coco.imgs[image_id]
        annotations = coco.loadAnns(coco.getAnnIds([image_id]))
        path = COCO_IMG_PATH+'images/'+image_info["file_name"][9:]
        image = cv2.imread(f'{path}')
        layout = load_coco_annotations(annotations, coco)
        for block in layout:
            segment = block.crop_image(image)
            block.text = ocr_agent.detect(segment)

        image_id = image_info["id"]
        count = count + 1
        print(folder, ' - ', count)

        file_id_array = image_info["file_name"].split('-')
        array_seg1 = file_id_array[1].split('_')
        image_folder = array_seg1[0][2:]
        cv_number = array_seg1[1]
        page = file_id_array[2][3:4]

        for text in layout:
            related_text = text.text.replace('\n',' ')
            data_item = [image_folder, cv_number, page, related_text, text.type]
            data.append(data_item)

all_data = pd.DataFrame(data, columns=['Folder','Id','Page','Text', 'Section'])
all_data.to_csv(main_folder + '/Annotated_data.csv')
```

Figure 6.30: Code block for creating the dataset using annotations

Text	Section
Dedicated third year undergraduate with strong interpersonal skills a	Profile
ABOUT ME	Profile
Dedicated third year undergraduate with strong interpersonal skills a	Education
EDUCATION	Education
B.SC. (HONS.) IN INFORMATION TECHNOLOGY UNIVERSITY OF MORAT	Education
HOLY FAMILY CONVENT, COLOMBO 04 G.CE (A/L - 2013) - PHYSICAL SC	Education
PROFESSIONAL QUALIFICATIONS	Education
Successfully completed a Certificate course in Computer Science at N	Education
AWARDS & ACHIEVEMENTS	Awards Achievements
Dean's list in Level 1 Semester 2	Awards Achievements
EXTRA CURRICULAR ACTIVITIES	Extra Curricular/ Roles and Responsibilities
University of Moratuwa Rotaractor - Rotaract Club (2016-Present)	Extra Curricular/ Roles and Responsibilities
Holy Family Convent Vice President- Buddhist Union (2011) Member	Extra Curricular/ Roles and Responsibilities
NON-TECHNICAL PROJECTS	Extra Curricular/ Roles and Responsibilities
The Exemplar- Are You Ready? 2016 Session co-chair person Organize	Extra Curricular/ Roles and Responsibilities
TECHNICAL SKILLS	Technology/Technical skills
Programming Languages - C, Java Databases - MySQL, Oracle Web de	Technology/Technical skills

Figure 6.31: Extracted dataset

6.2.2.3 Implementing the classification model

Main objective of this task in this module is to build a multiclass classification model which can predict the section class for a given content. As the first step, the dataset was read by using the Pandas python library and then removed all the unnecessary columns from the DataFrame. After removing the column, all the tuples which include null

values also have been removed using the dropna() method. Figure 6.32 shows the sample data from the DataFrame after removing unnecessary details.

```
df = pd.read_csv('All Annotated CVs/Annotated_data.csv')
df = df.drop(df.columns[[0, 1, 2, 3]],axis = 1)
df = df.dropna()

df.head()
```

	Text	Section
0	Dedicated third year undergraduate with strong...	Profile
1	ABOUT ME	Profile
2	Dedicated third year undergraduate with strong...	Education
3	EDUCATION	Education
4	B.SC. (HONS.) IN INFORMATION TECHNOLOGY UNIVER...	Education

Figure 6.32: Dropping unnecessary columns

As the second step all the text content was preprocessed before inputting that data to the model training process. As in Figure 6.33 first it converts all the text to the lower case and a regex which catches and removes special characters was used in this process. In addition to that, stopwords from NLTK library also have been used to remove unnecessary words from the dataset. Further, input text will be tokenized and then all the tokenized words will go through the lemmatization process. As the final step in the text preprocessing step all the lemmatized words will join again as a sentence.

```
special_character_remover = re.compile('[/(){}\[\]\|@,;]')
STOPWORDS = set(stopwords.words('english'))
```

```
def clean_text(text):
    text = text.lower()
    text = special_character_remover.sub(' ', text)
    text = word_tokenize(text)
    text = [lemmatizer.lemmatize(word) for word in text if not word in set(all_stopwords)]
    text = ' '.join(text)
    return text

df['Text'] = df['Text'].apply(clean_text)
```

Figure 6.33: Code block for preprocessing the text content

In this step, first the system splits the preprocessed dataset as the training dataset and the testing dataset. Since the test size = 0.2, the system takes 80% for the training dataset and 20% for the testing dataset.

After splitting the dataset, a pipeline was used to train the model which includes the CountVectorizer, TfidfTransformer and the Support Vector Machine with SGD Classifier.

CountVectorizer is a famous tool which is provided by the scikit-learn library. It is used to convert a given text into a vector based on the number of times (count) that each word appears across the full text. It creates a matrix, and each unique word is represented by a column in the matrix, and each sample of text from the document is represented by a row in the matrix (Figure 6.34).

```
document = ["Java, Python are the Technical Skiils",
            "A motivated individual",
            "Working Experience in the company"]
```

```
Vocabulary:  {'java': 5, 'python': 7, 'are': 0, 'the': 10, 'te
chnical': 9, 'skiils': 8, 'motivated': 6, 'individual': 4, 'wo
rking': 11, 'experience': 2, 'in': 3, 'company': 1}
Encoded Document is:
[[1 0 0 0 0 1 0 1 1 1 1 0]
 [0 0 0 0 1 0 1 0 0 0 0 0]
 [0 1 1 1 0 0 0 0 0 0 1 1]]
```

Figure 6.34: Output of simple sentence array after Count Vectorizing

In term TF-IDF, Tf is the term frequency while the Idf stands for the Inverse document frequency. In Tf it calculates the frequency of each term in the given text content and the Idf finds how a term is common among the given documents.

The SGDClassifier can be used for both linear regression and support vector machine (SVM) tasks, depending on the choice of loss function and the specific parameters used. Here the system uses the SGDClassifier with Support Vector Machine algorithm. (Figure 6.35).

```
from sklearn.model_selection import train_test_split
X = df.Text
y = df.Section
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 42)
```

```
#svm model
from sklearn.linear_model import SGDClassifier

text_clf_svm = Pipeline([('vect', CountVectorizer()),
                         ('tfidf', TfidfTransformer()),
                         ('clf-svm', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, random_state=42))
                         ])

text_clf_svm = text_clf_svm.fit(X_train, y_train)

y_pred_svm = text_clf_svm.predict(X_test)

print(f'accuracy {accuracy_score(y_pred_svm,y_test)}')
```

```
accuracy 0.8881431767337807
```

Figure 6.35: Code block for splitting dataset and training the model

Figure 6.36 shows how the system predicts the section class for unseen data using the pre-trained multiclass classification model.

```
test_data ="An enthusiastic and passionate individual who like to work with
cleaned_text = clean_text(test_data)

print('Predicted Section - '+ text_clf_svm.predict([cleaned_text])[0])
```

```
Predicted Section - Profile
```

Figure 6.36: Shows the predicted output for given text content

```
f1_svm = f1_score(y_pred_svm, y_test, average="weighted")

print("F1_svm Score:", f1_svm)

print(classification_report(y_pred_svm, y_test))
```

```
F1_svm Score: 0.8923621561035323
                                        precision    recall  f1-score   support

                         Awards Achievements       0.85      0.85      0.85       106
                                   Education       0.94      0.88      0.91       155
Extra Curricular/ Roles and Responsibilities       0.78      0.80      0.79       112
                                    Interest       0.84      0.93      0.88        41
                              Personal Skills       0.84      0.91      0.88        47
                                     Profile       1.00      0.82      0.90        66
                                    Projects       0.93      0.92      0.92       131
                                     Refrees       0.98      0.96      0.97       101
                  Technology/Technical skills       0.94      0.94      0.94       126
                             Work Experience       0.27      0.67      0.39         9

                                    accuracy                           0.89       894
                                   macro avg       0.84      0.87      0.84       894
                                weighted avg       0.90      0.89      0.89       894
```

Figure 6.37: Classification for the SVM model

The performance assessment of our multiclass classification model for predicting resume sections is shown in Figure 6.37. It gives a thorough breakdown of the model's precision, F1-score, and full categorization report. The accuracy measure shows the model's capacity to correctly categorize the resume sections and demonstrates the general accuracy of its predictions.

6.2.2.4 Dividing section using algorithm and extracting section wise data from the resume

As the first task of this, the system must divide the section from the resume. The system made use of the layout parser tool to simplify the division of sections and the extraction of section-wise data from resumes. To detect individual words and create bounding boxes around them in the system process (Figure 6.38), the layout parser proved to be

a useful tool. The layout parser enables the system to examine the spatial layout of the resume while precisely identifying and isolating each word that is used therein. Drawing bounding boxes around these words gave the resume's content a structured representation, facilitating processing and section identification later on (Figure 6.39). By utilizing the positioning information of words inside the resume document, this method ensured a more granular level of analysis and allowed us to properly extract section-wise data.

```
pdf_tokens, pdf_images = lp.load_pdf(pdf_path, load_images=True)

text = ''
lines = []

for i in range(len(pdf_tokens[0])):
    block = pdf_tokens[0][i]
    next_block = None
    if not i+1==len(pdf_tokens[0]):
        next_block = pdf_tokens[0][i+1]

    if text == '':
        text = block.text

    if not next_block == None and (abs(next_block.block.x_1 - block.block.x_2) < 10 or abs(next_block.block.y_1 - block.block.y_
        text += (' ' + next_block.text)
    else:
        text = ''

lp.draw_box(pdf_images[0], pdf_tokens[0])
```

Figure 6.38: Identifying word blocks from resume

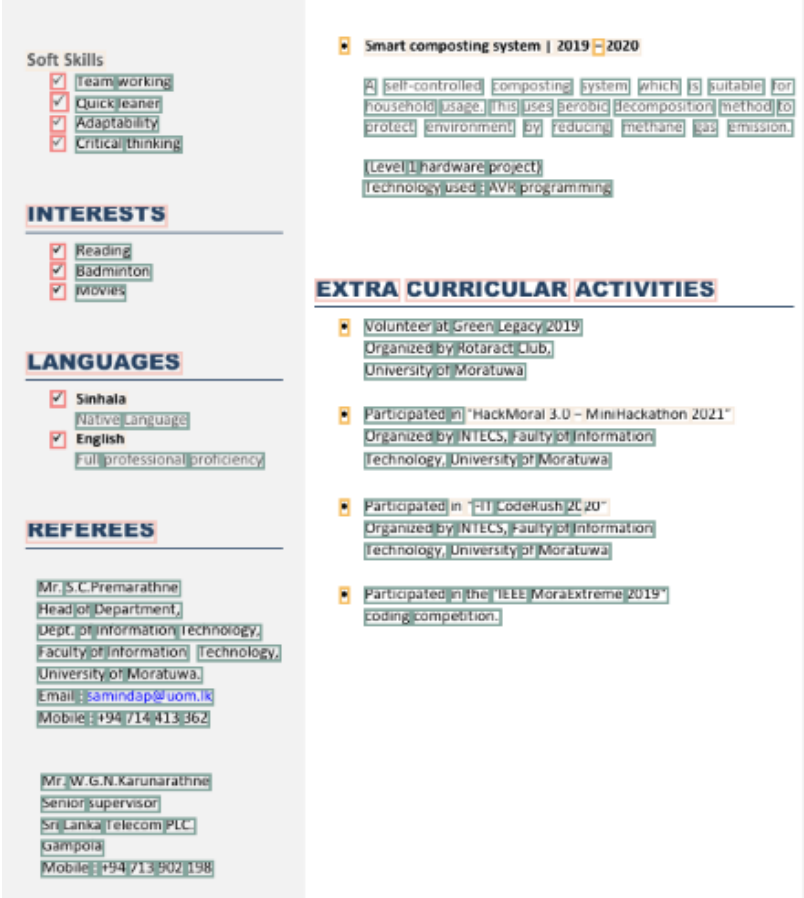


Figure 6.39: Identified word boxes and bounding boxes around them

Then the system demanded the challenging work of locating distinctive paragraphs within the resume document. The system used a straightforward yet efficient approach to accomplish this goal by utilizing the bounding boxes acquired in the previous phase and the spaces between these boxes. It established a threshold value that indicates the relevance of a space as a marker of paragraph borders by carefully examining the positional information of the bounding boxes and taking the gaps between them into account. This criterion was used to determine whether a gap between two bounding boxes was significant enough to indicate the start of one paragraph and the conclusion of another (Figure 6.40).

```
block_sets = []

first_block = None
last_block = None
x_first = 0
x_last = 0
y_last = 0

for y in range(len(pdf_images)):
    temp_block_sets = []
    for i in range(len(pdf_tokens[y])):
        block = pdf_tokens[y][i]
        next_block = None

        if not i+1==len(pdf_tokens[y]):
            next_block = pdf_tokens[y][i+1]

        if text == '':
            text = block.text
            first_block = block
            x_first = block.block.x_1
            x_last = block.block.x_1
            y_last = block.block.y_1

        if not next_block == None and (abs(next_block.block.x_1 - block.block.x_2) < 10 or abs(next_block.block.y_1 - block.bloc
            text += (' ' + next_block.text)
            if next_block.block.x_1 < x_first:
                x_first = next_block.block.x_1
            if next_block.block.x_2 > x_last:
                x_last = next_block.block.x_2
            if next_block.block.y_2 > y_last:
                y_last = next_block.block.y_2
        else:
            first_block.block.x_1 = x_first
            first_block.block.x_2 = x_last
            first_block.block.y_2 = y_last
            first_block.text = text
            first_block.page = y
            block_sets.append(first_block)
            temp_block_sets.append(first_block)
            text = ''
```

Figure 6.40: Algorithm for dividing distinct paragraph from resumes using gaps

Through this algorithm, the system successfully delineated and differentiated various paragraphs within the resume. This approach allowed the system to organize the resume content into coherent and distinct sections, facilitating subsequent analysis and extraction of section-specific information as shown in the Figure 6.41.

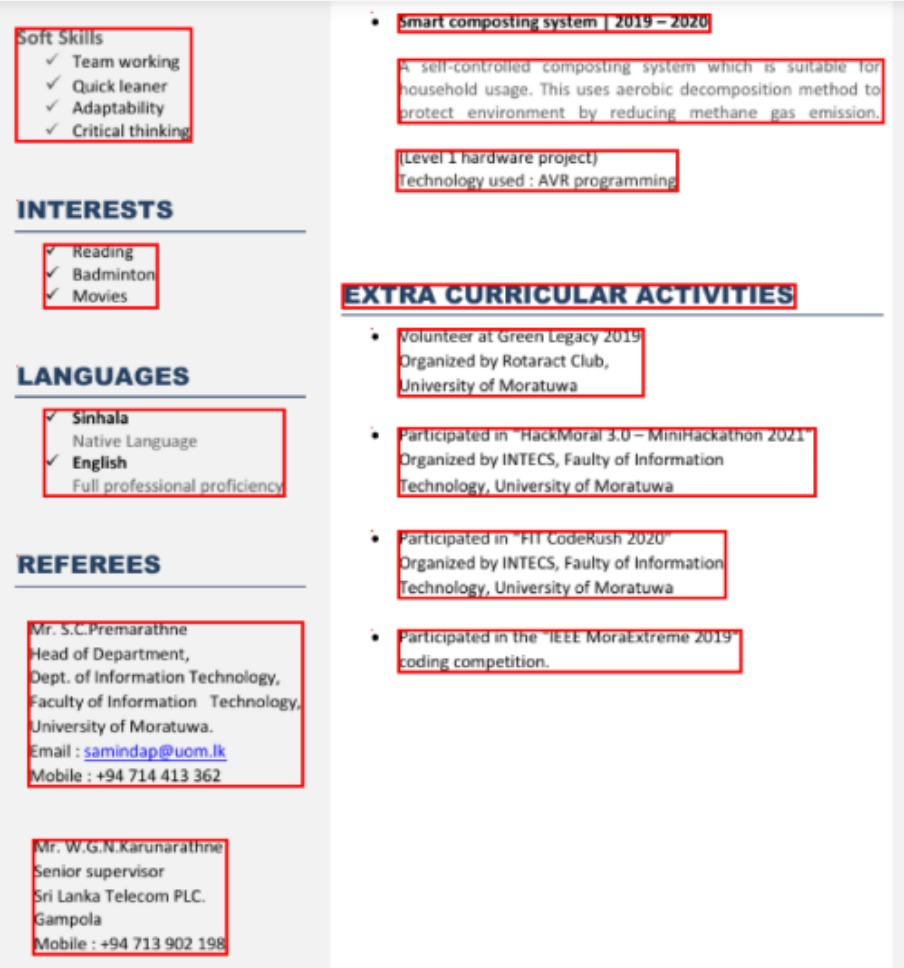


Figure 6.41: Identified distinct paragraphs

Once the different paragraphs in the resume were recognized, the system automatically applies the trained multiclass classification model to each paragraph, essentially giving each paragraph the appropriate section. In this step, the content of each paragraph was precisely classified into categories like personal information, education, work experience, or abilities using the power of machine learning. To increase the accuracy of that, the system uses a few rule-based approaches as well as shown in Figure 6.42.

```
for txt in block_sets:
    cleaned_text = clean_text(txt.text)
    pre = loaded_model.predict([cleaned_text])[0]

    if pre == 'Profile':
        if ((cleaned_text=="profile") or (cleaned_text=="about me") or (cleaned_text=="objective") or (cleaned_text=="summary")):
            profileContent.append(cleaned_text)
            profileContentBox.append(txt)
        elif (len(cleaned_text.split(" ")) > 5):
            profileContent.append(cleaned_text)
            profileContentBox.append(txt)
    elif pre == 'Education':
        if ((cleaned_text=="education") or ("gpa" in cleaned_text) or ("college" in cleaned_text)):
            educationContent.append(cleaned_text)
            educationContentBox.append(txt)
        elif (len(cleaned_text.split(" ")) > 5):
            educationContent.append(cleaned_text)
            educationContentBox.append(txt)
    elif pre == 'Technology/Technical skills':
        if ((cleaned_text=="language") or (cleaned_text=="personal")):
            continue
        technicalSkillsContent.append(cleaned_text)
        technicalSkillsContentBox.append(txt)
    elif pre == 'Personal Skills':
        personalSkillsContent.append(cleaned_text)
        personalSkillsContentBox.append(txt)
    elif pre == 'Interest':
        interestContent.append(cleaned_text)
        interestContentBox.append(txt)
    elif (pre == 'Extra Curricular/ Roles and Responsibilities') or (pre == 'Awards Achievements'):
        if (len(cleaned_text.split(" ")) > 3):
            awardsResponsibilitiesContent.append(cleaned_text)
    elif pre == 'Refrees':
        if (("referees" in cleaned_text) or ("referee" in cleaned_text)):
            tempRefereesContent.append(cleaned_text)
            tempRefereesContentBox.append(txt)
        elif (len(cleaned_text.split(" ")) > 4):
            tempRefereesContent.append(cleaned_text)
            tempRefereesContentBox.append(txt)
```

Figure 6.42: Rule-based approach for extracting content of each paragraph

Additionally, the system used a pre-trained custom Named Entity Recognition (NER) model to extract significant entities from the resume. In order to evaluate a candidate's qualities, this NER model had been particularly trained to recognize and extract crucial entities, such as person names, interests, personal abilities, and phone numbers. The system will be able to extract and annotate these significant things from the section-wise data by using our NER model (Figure 6.43).

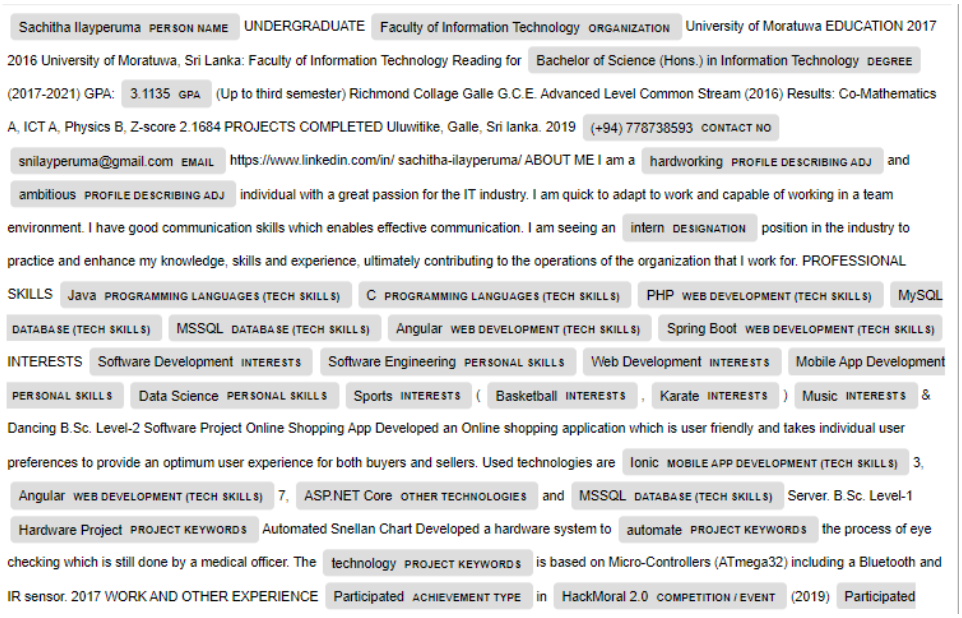


Figure 6.43: Entities annotated by trained NER

Finally, by merging the output from the multiclass classification model and the customized NER, our system produced section-wise data (Figure 6.44).

```
# Export content Json
import json

# some JSON:
result = {
"profileContent":profileContent,
"educationContent":educationContent,
"technicalSkillsContent":technicalSkillsContent,
"personalSkillsContent":personalSkillsContent,
"interestContent":interestContent,
"awardsResponsibilitiesContent":awardsResponsibilitiesContent,
"projectsContent":projectsContent,
"refreesContent":refreesContent,
"personName":personName,
"sectionOrder":sectionOrder
}

# parse x:
y = json.dumps(result)

# Writing to sample.json
with open("content.json", "w") as outfile:
    outfile.write(y)
```

Figure 6.44: Generate JSON file of section contents

At the final stage of the system, it goes beyond text extraction, and it generates images of the resume with relevant bounding boxes drawn around the sections pertinent to each category. For instance, in the case of the profile section, the system selectively draws bounding boxes around the sections that are specifically relevant to profile information (Figure 6.45 and 6.46). This approach ensures that only the visually relevant content is highlighted and displayed for each section. Then the generated images are displayed on the web application when the user clicks the relevant identified section.

```
def export_section_img(section_box_arr, section_name):
    current_page = 0
    box_to_draw = []
    for box in section_box_arr:
        if current_page == box.page:
            box_to_draw.append(box)
        else:
            plt.figure(figsize=(20,14))
            img = lp.draw_box(pdf_images[current_page], box_to_draw)
#             plt.imshow(img)

            img.save('Section_imgs/temp_'+section_name+'_'+str(current_page)+'.png', "PNG")

            box_to_draw = []
            current_page = box.page
            box_to_draw.append(box)

    plt.figure(figsize=(20,14))
    img = lp.draw_box(pdf_images[current_page], box_to_draw)
#     plt.imshow(img)

    img.save('Section_imgs/temp_'+section_name+'_'+str(current_page)+'.png', "PNG")

export_section_img(profileContentBox, 'profile')
export_section_img(projectsContentBox, 'projects')
export_section_img(educationContentBox, 'education')
export_section_img(technicalSkillsContentBox, 'technical_skills')
export_section_img(personalSkillsContentBox, 'personal_skills')
export_section_img(interestContentBox, 'interests')
export_section_img(awardsResponsibilitiesContentBox, 'awards_responsibilities')
export_section_img(refreesContentBox, 'referees')
```

Figure 6.45: Generate section wise images with identified paragraphs

Figure 6.46: Sample generated image for projects section

6.2.3 Module 3 - Company Recommendation System and Interview Questions Suggestion

6.2.3.1 Dataset analysis

As the first step the collected initial dataset was analyzed to capture the data distribution in order to determine the list of companies for which the recommendation system will give an output on. Then the list of companies with data available for at least more than 70 resume data was filtered out and amongst these, a set of 10 reputed companies were selected for the system. The data distribution of the selected data set is depicted in Figure 6.47 and Figure 6.48.

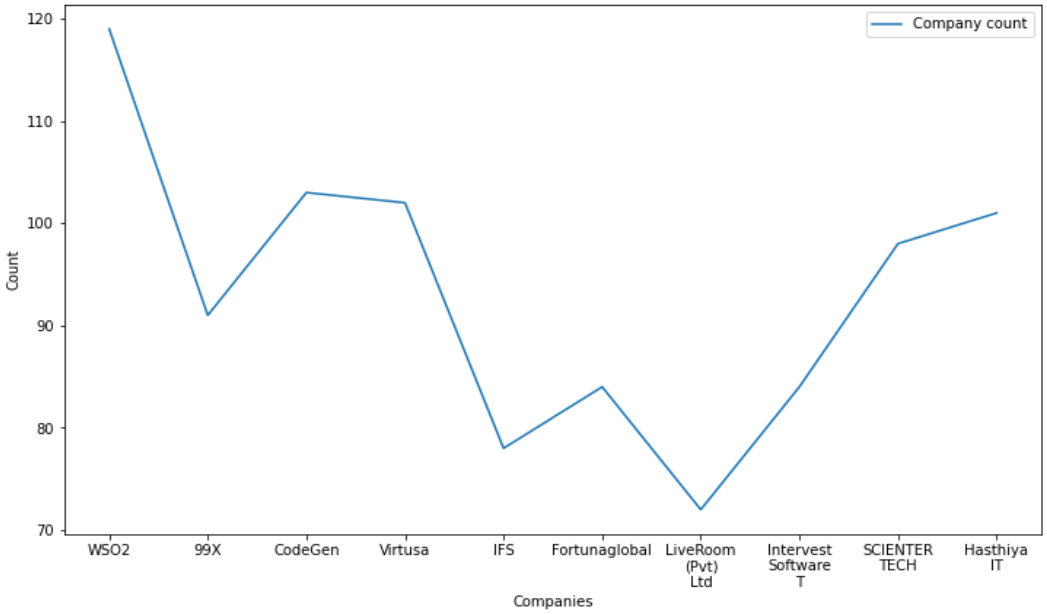


Figure 6.47: Data distribution graph of resume counts vs company

Virtusa	102
WSO2	115
Hasthiya IT	97
SCIENTER TECH	95
99X	88
Intervest Software T	81
LiveRoom (Pvt) Ltd	71
Fortunaglobal	83
CodeGen	98
IFS	77

Figure 6.48: Resume counts per selected companies

6.2.3.2 Dataset Preparation

6.2.3.2.1 Approach 01 - Analyzing whole text content of the resume

For approach 01, it was required to extract all the textual content from the collected previously selected resumes and prepare a dataset indicating for which companies the given resume had received first calls from the companies.

As the first step all the textual content of the relevant resumes were extracted into a new column named content in the above prepared dataset. Figure 6.49 shows the code for creation of the new column.

```
text_content = []
for i in range(len(company_dataset)):
    try:
        student = company_dataset.iloc[i]

        found = False

        pdf_doc = nlp(extract_text_from_pdf('./cv/'+ student["CV"]))
        final_text = str(pdf_doc)
        if len(final_text) > 800:
            text_content.append(final_text)
            found = True

        if not found:
            text_content.append('')

    except:
        text_content.append('')
        continue

    print(i+1, len(text_content))

company_dataset['content'] = text_content
company_dataset
```

Figure 6.49: Code for extracting resume content for a new column

Then a dataset was prepared such that it is suitable to be input into the selected models. A dataset with the selected company list as separate columns was created and 0's was inserted for each resume for companies for which they didn't receive first calls and 1's were inserted for companies they were selected. Figure 6.50 depicts how this was carried out using python and Figure 6.51 shows the new data set created.

```
for company in company_list:
    data.loc[data['Selected Org'].str.contains(company), 'Selected Org'] = company

company_dataset = data[['Reg No', 'Selected Org', 'CV']]

new_cols = []
new_cols.extend(company_dataset.columns.tolist())
new_cols.extend(company_list)

company_dataset = company_dataset.reindex(columns=new_cols, fill_value=0)
company_dataset = company_dataset.reset_index(drop=True)

for k in company_list:
    company_dataset.loc[company_dataset['Selected Org'].str.contains(k), k] = 1

company_dataset = company_dataset.drop(['Selected Org'], axis=1)

company_dataset.to_csv('multilabel_dataset.csv', index=False)
company_dataset
```

Figure 6.50: Code for adding 1's and 0's for selected companies

	Reg No	CV	Virtusa	WSO2	Hasthiya IT	SCIENTER TECH	99X	Intervest Software T	LiveRoom (Pvt) Ltd	Fortunaglobal	CodeGen	IFS
0	[redacted]	[redacted]	0	0	0	0	0	0	0	0	0	0
1	[redacted]		0	0	0	0	0	0	0	0	0	0
2	[redacted]		0	0	0	0	0	0	0	0	0	0
3	[redacted]		0	0	0	0	0	0	0	0	0	0
4	[redacted]		0	0	0	0	0	0	0	0	0	0
...	...		...	...	...	...	...	...	...	...	...	...
1325	[redacted]		0	0	0	0	0	0	0	0	0	0
1326	[redacted]		0	0	0	0	0	0	0	0	0	0
1327	[redacted]		0	0	0	0	0	0	0	0	0	0
1328	[redacted]		0	0	0	0	1	0	0	0	0	0
1329	[redacted]		0	0	0	0	0	0	0	0	0	0

1330 rows × 12 columns

Figure 6.51: Altered dataset with 1's and 0's

6.2.3.2.2 Approach 02 - Analyzing important features of resume

For the implementations of approach two it was required to prepare a dataset similar to approach 01 but instead of whole text content it was required to have section wise data. For that purpose, the pretrained NER model developed in module 4 was loaded using spaCy. Then a new data frame is created to accommodate extracted data. Then the previously extracted whole cv content was fed into the model and the output of the extraction was populated in the created data frame. The code for extracting section wise data is depicted in Figure 6.52 and the prepared dataset is shown in Figure 6.53.

```
for i in df.index:

  doc = trained_nlp(df['content'][i])

  df_new['Reg No'][i] = df['Reg No'][i]

  for ent in doc.ents:
    if str(df_new[ent.label_][i])!='nan':

      df_new[ent.label_][i] =  df_new[ent.label_][i] + ' ' + ent.text

    else:
       df_new[ent.label_][i] = ent.text
```

Figure 6.52: Code for Extracting Section-Wise Data

	Reg No	PROFILE DESCRIBING ADJ	DEGREE	GPA	GRADUATION YEAR	SCHOOL	AL STREAM	PROGRAMMING LANGUAGES (TECH SKILLS)	DATABASE (TECH SKILLS)	WEB DEVELOPMENT (TECH SKILLS)	...	ORGANIZATION
0	104001I	NaN	BSc. (Hons) Degree in Information Technology B...	NaN	NaN	St. Joseph's College Dalugama Parish -	NaN	Java Python Micro C JAVA JAVA C# C Python	MySQL	HTML Php	...	Old Boys Union Department of Computational Mat...
1	135060B	NaN	B.Sc. Degree in Information Technology and Man...	NaN	NaN	NaN	Arts Stream	C# Java C#	SQL server SQL server MSSQL SQL MySQL	JavaScript JQuery HTML CSS Bootstrap PHP HTML/...	...	Vembadi Girls' High School Vembadi Girls' High...
2	135104P	dedicated flexible undergraduate team working...	Bachelor's Degree in Information Technology. ...	1.28 2.19 1.13	NaN	J/Vembadi Girls Vaishnavy Vijayarathinam	Arts Stream	C# C# C# C C	MSSQL SQL server □ SQL Server □	JavaScript Bootstrap CSS HTML. □ □ HTML/CSS/ X...	...	XEEDIT Vembadi Girls' High School Vembadi Girl...

Figure 6.53: Sample from the Dataset Containing Section-wise Data

Then it was merged with the dataset containing selected company data to create the final dataset and a screenshot of that dataset is depicted in Figure 6.54.

Reg No	PROFILE DESCRIBING ADJ	DEGREE	GPA	GRADUATION YEAR	SCHOOL	AL STREAM	PROGRAMMING LANGUAGES (TECH SKILLS)	DATABASE (TECH SKILLS)	...	WSO2	Hasthiya IT	SCIENTER TECH	99X	Intervest Software T	LiveRoom (Pvt) Ltd	Fortunaglobal	CodeGen	IFS
104001I	NaN	BSc. (Hons) Degree in Information Technology B...	NaN	NaN	St. Joseph's College Dalugama Parish -	NaN	Java Python Micro C JAVA JAVA C# C Python	MySQL	...	0	0	0	1	0	0	0	0	0
135060B	NaN	B.Sc. Degree in Information Technology and Man...	NaN	NaN	NaN	Arts Stream	C# Java C#	SQL server SQL server MSSQL SQL MySQL	...	0	0	1	0	0	0	1	0	0
135104P	dedicated flexible undergraduate team working...	Bachelor's Degree in Information Technology. ...	1.28 2.19 1.13	NaN	J/Vembadi Girls Vaishnavy Vijayarathinam	Arts Stream	C# C# C# C C	MSSQL SQL server □ SQL Server □	...	0	0	0	0	0	0	0	0	0

Figure 6.54: Dataset with Company Selection Details

6.2.3.2.3 Approach 03 - Analyzing resume through different section details

For approach three, it was required to prepare a dataset such that it includes GPA and the frequencies of terms captured by NER (developed in module 4) under different sections such as GPA, profile describing adjectives, degree, graduation year etc. Then it was merged with the company selected detail table and one hot encoding for labels was put into a new column to prepare the final dataset. The prepared dataset after executing this task is depicted in Figure 6.55.

PROFILE DESCRIBING ADJ	DEGREE	GPA	GRADUATION YEAR	SCHOOL	AL STREAM	PROGRAMMING LANGUAGES (TECH SKILLS)	DATABASE (TECH SKILLS)	WEB DEVELOPMENT (TECH SKILLS)	BACKEND SERVICES	...	VERSION CONTROLLING (TECH SKILLS)	PERSONAL SKILLS	INTERESTS	ROLE (EXTRA CURRICULAR)	ORGANIZATION (EXTRA CURRICULAR)	ACHIEVEMENT TYPE	COMPETITION / EVENT	PROJECT KEYWORDS	OTHER TECHNOLOGIES	one_hot_labels
5	1	2.12	0	0	1	3	4	7	0	...	0	3	0	0	0	0	0	8	4	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
3	1	3.29	0	1	1	4	3	11	0	...	1	9	9	3	4	4	12	3	2	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
3	1	3.72	0	1	0	2	2	8	0	...	2	2	5	3	1	0	4	5	1	[0, 0, 0, 0, 0, 0, 1, 1, 0, 0]
4	1	3.48	0	1	1	6	3	7	0	...	1	3	3	4	7	6	6	5	2	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
6	1	0.00	0	1	0	3	2	7	0	...	1	2	0	2	0	2	3	3	0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Figure 6.55: Dataset with One Hot Encoding

6.2.3.3 Data preprocessing

Data preprocessing was required to be carried out for the following data.

1. Entire resume text content
2. GPA
3. Technical skills
4. Project keywords

Data preprocessing carried out for entire resume content, technical skills, and project keywords consisted of similar steps. First the empty rows were dropped and then data cleaning was performed. The provided code in Figure 6.56 performs text cleaning by tokenizing the text, lemmatizing, and stemming the tokens, removing short words and stop words, and finally, joining the cleaned tokens back into a string. This process helps in standardizing and reducing noise in the text data, making it more suitable for the models.

```
def tokenize_lemma_stopwords(text):
    text = text.replace("\n", " ")
    tokens = nltk.tokenize.word_tokenize(text.lower())
    tokens = [t for t in tokens if t.isalpha()]
    tokens = [wordnet_lemmatizer.lemmatize(t) for t in tokens]
    tokens = [stemmer.stem(t) for t in tokens]
    tokens = [t for t in tokens if len(t) > 2]
    tokens = [t for t in tokens if t not in stopwords]
    cleanedText = " ".join(tokens)
    return cleanedText
```

Figure 6.56: Data Preprocessing

6.2.3.3.1 GPA

As the initial step the section wise dataset was loaded, and a new data frame was created to contain only the company labels and GPA columns. Then the missing values were handled by dropping the rows where the value of GPA column was ‘nan’. After dropping the rows with missing data, the dataset was reduced from 823 entries to 436 entries.

The GPA values in some resumes were mentioned on a semester-wise basis rather than the cumulative GPA. To address this, further preprocessing steps were performed. Also, in the extracted GPAs, there existed some non-GPA numbers (such as years), non-numerical data, and combination of GPA and non-numerical data. If the GPA values

are valid was checked by considering if they fall in the range 0 to 4.2, and then non-numerical data were eliminated by checking if they are numerical. If found not to be numerical, they were split and taken to an array and each array element was checked for being a GPA value or not and stored in a new array. In cases where multiple GPAs were found within the same array, an averaging function was applied to obtain the average GPA value. The code segments corresponding to these preprocessing steps can be found in Figure 6.57.

```
for index in range(len(new_df['GPA'])):
  i= new_df['GPA'].iloc[index]
  try:
    if check_gpa(i):
      continue
    elif i.isnumeric():
      if check_gpa(i):
        continue
    else:
      i_arr = i.split()
      sgpas = []
      print (i)
      for x in i_arr:
        if check_gpa(x):
          sgpas.append(float(x))
      avg = Average(sgpas)

      new_df['GPA'].iloc[index]= avg

  except Exception as e:
    print (e)
```

```
def check_gpa(gpa):
  try:
    gpa = float(gpa)
    if gpa<=4.2 and gpa>0:
      return True
    else:
      return False
  except:
    return False
```

```
def Average(lst):
    return sum(lst) / len(lst)
```

Figure 6.57: Code Segment for Preprocessing GPA

Figure 6.58 illustrates how GPA was cleaned using above mentioned steps.

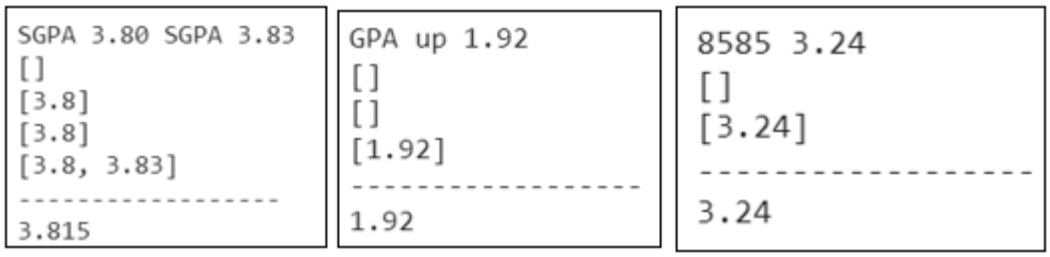


Figure 6.58: How GPA were Cleaned

6.2.3.3.2 Technical skills

From section wise dataset, new datasets were prepared by only selecting the relevant columns for technical skills dataset. For technical skills, the data of the columns programming skills, database skills, web development skills, backend services, project management tools, IDEs, and version controlling tools were combined into a new column.

First, the empty rows were dropped, causing the dataset to be reduced from 823 entries to 789 entries. Repetition of technologies and skills observed in the technical skills data are depicted in Figure 6.59. To handle the repetition, a function was used to iterate through each term in technical skills string and transferred to an array. Then set function was applied to obtain unique technological skills. Removal of repetitions is shown in Figure 6.59.

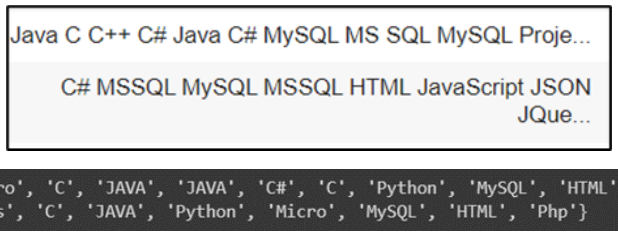


Figure 6.59: Removal of Duplication Data

Also, there existed some technologies which were captured through NER as depicted in the latter part of figure 6.59. They were handled inside the same function used to remove repetitions and a cleaned version of the data sample in Figure 6.60.

Figure 6.60: Separation of the Technologies as Separate Records

Extracted data which were not relevant for technical skills were then removed by selecting through features produced by the vectorizer. Figure 6.61 contains such features removed.

```
not_features=['05', '0714329357', '0719968294', '0777767027', '09', '10', '1085',
       '11', '111', '14', '18', '1994', '2014', '2016', '2021', '21',
       '21000', '2647', '289', '415', '423447141', '54', '5dd6d21e36d33',
       '5dd7846732984', '62a9a8165', '659', '66', '702690280', '77', '81',
       '8620', '94', '940702171495', '9471', '94779621431', '96064114b',
       '9a','aanaiccottai', 'accumulative', 'achivements', 'algorithm', 'and','anishka', 'application', 'askme',
       'atapattu', 'atheletic', 'attuneco', 'author','balasingam',
       'based', 'boossa','captain', 'chathurangi', 'chemistry', 'cid',
       'cipant', 'cipated', 'colombo', 'columnist',
       'com', 'complcompleted', 'complete', 'computa','cutter',
        'dean', 'dedicated', 'dep', 'development', 'developmenthtml',
       'devolopmet', 'dhanushika','effec', 'farmwatta', 'feb','globalwavenet',
       'hasinthak',  'leadership','madhubahashini','manually', 'mariyathas',
       'matale', 'milinda', 'moksha22', 'nagavillu', 'nasrina', 'national',
```

Figure 6.61: Removal of Non-relevant Data from Features

6.2.3.3.3 Tokenization, Vectorization and Encoding

The textual inputs needed to be converted into numerical representations so that machine learning algorithms could understand. For approach 01, where contextual nature was important since whole text content was analyzed, it required preserving the context information to experiment its significance on the classification task. Hence, in BERT model, the BERT tokenizer was used to tokenize the sentences and to encode them. Here the Byte Pair Encoding is used. Encoding is the process of transforming text data into a format that the BERT model can comprehend. The sequential structure of the text is preserved through the encoding process. Tokenization, special token addition, padding, and the creation of additional inputs like attention masks and token type IDs are a few of the processes involved here. For approach 02, no contextual relationships were required as they contain the keywords only. Thereby, vectorization was found to be adequate for the transformation of text data into numerical representations. Unlike encoding, the vectorization process does not preserve sequential nature. TF-IDF vectorizer is one of the commonly used vectorizers in text classification tasks. Here the TF-IDF vectorizer from scikit learn is used to generate vectors for technical skills and project keywords.

6.2.3.4 Implementation

6.2.3.4.1 BERT model for analyzing whole text content of resume (Approach 01)

The prepared dataset was first loaded into the notebook using the pandas library. From the prepared dataset a set of 30 random tuples were removed for the final test set data. To better understand the dataset the average sentence length in data in the content column and the standard deviation was calculated. The average sentence length of a resume was found to be approximately 416 words.

After studying the dataset distribution, as the next step the dataset was modified so that it is suitable to be input into the model. For the multi-label classification task using BERT, it mainly requires two inputs. For this task the two inputs would be the content along with one-hot encodings that correspond to that content have to be inputted to the model. What one-hot encoding does is that it transforms the categorical data into a format that the algorithm can understand. Figure 6.62 illustrates a sample of the finalized dataset with the new column.

	Reg No	content	Virtusa	WSO2	Hasthiya IT	SCIENTER TECH	99X	Intervest Software T	LiveRoom (Pvt) Ltd	Fortunaglobal	CodeGen	IFS	one_hot_labels
0	185066U	LEESHA SAMADHI A B O U T M E An energetic, a...	0	0	0	0	0	0	0	0	1	0	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
1	185067A	FAMITHA ROSNI UNDERGRADUATE (UNIVERSITY OF MO...	0	0	0	0	0	0	0	0	1	0	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
2	185068D	EDUCATION 2018 - Present 2018 - Present 2017 2...	0	0	0	0	0	1	0	1	0	0	[0, 0, 0, 0, 0, 1, 0, 1, 0, 0]
3	185069G	EDUCATION Undergraduate at University of M...	0	1	0	0	0	0	0	0	0	0	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
4	185070C	HASHANI SATKUNALINGAM UNDERGRADUATE ABOUT ME...	0	0	0	0	0	0	0	0	0	0	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Figure 6.62: Dataset with one-hot encodings column

Also, in this implementation the raw text as whole which is extracted from resume is inputted to tokenizer without any preprocessing as BERT is a pretrained model and delicate preprocessing is not required. Traditional data preprocessing techniques stemming and stopword removal will remove important information that may sometimes be important in understanding context.

As the next step the content was tokenized and encoded using the specific tokenizer for the selected transformer model. In this approach we are using the BERT transformer model and hence the BERT tokenizer was used to encode the contents.

Previously, it was found that the average content length of a resume is around 416 words. Therefore, initially in creating the model a limit of 300 tokens was used to deal

with the memory issues faced. But the 300 token limit was found to be insufficient as mostly the initial tokens would include basic details like email, contact details etc. whereas other important data with context were lost. Therefore, maximum token length was increased to 400 by also taking into consideration the average content length found before.

In this approach, the BERT base model (uncased) is used, and its basic configurations are shown in Figure 6.63. In this model it does not differentiate words based on lowercase or uppercase in the sense that all text is lowercased before tokenizing. Also, this model has been developed with 12 encoder layers and has a vocabulary of 30522.


```
Model config BertConfig {
  "_name_or_path": "bert-base-uncased",
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "transformers_version": "4.30.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}
```


Figure 6.63: Basic configurations of BERT model used

As the next step all the data entries with only one occurrence of one-hot encodings are identified and they are inserted into the train dataset to make sure that no unseen data is being inserted to the validation dataset. Then the dataset is split into train and validation datasets using the train_test_split function. Here the dataset was split in such a way that the validation dataset is ten percent. Then both the train and validation datasets are transformed into torch tensors as it is the required datatype of selected model.

Next, the required model is loaded. Here, for the task at hand the BERT for sequence classification was required since the series of content observations needed to be given

appropriate labels. As the next step, the custom optimizing parameters were set and Adam optimizer was applied.

Figure 6.64 illustrates the code for training the model and Binary Cross Entropy With Logits (BCEWithLogits) was used as the loss function. In this implementation the model was trained under 30 epochs.

```
for _ in trange(epochs, desc="Epoch"):

  model.train()

  tr_loss = 0 #running loss
  nb_tr_examples, nb_tr_steps = 0, 0

  for step, batch in enumerate(train_dataloader):

    batch = tuple(t.to(device) for t in batch)
    b_input_ids, b_input_mask, b_labels, b_token_types = batch
    optimizer.zero_grad()

    # Forward pass for multilabel classification
    outputs = model(b_input_ids, token_type_ids=None, attention_mask=b_input_mask)
    logits = outputs[0]
    loss_func = BCEWithLogitsLoss()
    loss = loss_func(logits.view(-1,num_labels),b_labels.type_as(logits).view(-1,num_labels))
    train_loss_set.append(loss.item())

    # Backward pass
    loss.backward()
    optimizer.step()
    tr_loss += loss.item()
    nb_tr_examples += b_input_ids.size(0)
    nb_tr_steps += 1

  print("Train loss: {}".format(tr_loss/nb_tr_steps))
```

Figure 6.64: Code for training model

Training loss plotted against the epoch number is shown in Figure 6.65.

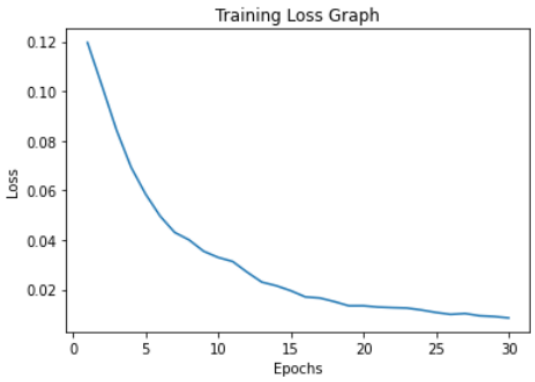


Figure 6.65: Training loss graph

After training the model, the model which showed best performance was saved.

6.2.3.4.2 Problem Transformation Methods for approach 01 and 02

In the problem transformation approaches, it addresses the multilabel problem by transforming the problem into multiple binary classification tasks. This strategy was implemented to experiment the relationship of getting selected for a specific company by considering whole text content, technical skills, GPA, and project keywords (Approach 01 and approach 02). The selected problem transformation methods (Binary relevance, Classifier chains, and Label Powersets) were imported from 'skmultilearn.problem_transform' module.

Under each of these problem transformation models 04 different models were trained as their base models. The selected base models are Linear SVC, Gaussian Naïve Bayes, Decision Tree, and Logistic Regression. All of these models have been successfully used in research for similar multilabel text classification tasks.

Implemented models:

- Binary Relevance
    - Linear SVC
    - Gaussian Naive Bayes
    - Decision Tree
    - Logistic Regression
- Classifier Chains
    - Linear SVC
    - Gaussian Naive Bayes
    - Decision Tree
- Logistic Regression
    - Linear SVC
    - Gaussian Naive Bayes
    - Decision Tree
    - Logistic Regression
- Label Powersets
    - Linear SVC
    - Gaussian Naive Bayes

- Decision Tree
- Logistic Regression

Figure 6.66 contains a sample of the code segment for implementing one of the models.

```
pip install scikit-multilearn
from skmultilearn.problem_transform import BinaryRelevance

from sklearn.naive_bayes import GaussianNB

BinaryRelNB = BinaryRelevance(GaussianNB())
BinaryRelNB.fit(vectorised_train_documents, train_labels)
```

Figure 6.66: Code Segment for Implementing Binary Relevance Gaussian NB

6.2.3.4.3 Algorithm adaptation methods for approach 01 and 02

In the context of multilabel classification, algorithm adaptations involve modifying or developing new algorithms to specifically handle multiple labels associated with each occurrence. Inherently, Decision Trees and k-Nearest Neighbors have the ability to handle multi-label classification problems. Hence, for whole text content, technical skills, GPA, and project keywords (approach 01 and approach 02) these two algorithms were applied (Linear SVC, Gaussian Naïve Bayes, Decision Tree, and Logistic Regression).

Implemented models:

- Decision Tree
- k-Nearest Neighbors

Figure 6.67 shows the code for implementing one of algorithm adaptation methods.

```
from sklearn.tree import DecisionTreeClassifier

dtClassifier = DecisionTreeClassifier()
dtClassifier.fit(vectorised_train_documents, train_labels)
```

Figure 6.67: Code Segment for Implementing Decision Tree Classifier

6.2.3.4.4 Ensemble methods for approach 01 and 02

By combining the predictions of various base models, ensemble methods like bagging, chaining, and gradient boosting offer methods to enhance multilabel classification

performance. Under the bagging classifier, 04 different base models were trained (Linear SVC, Gaussian Naïve Bayes, Decision Tree, and Logistic Regression).

Implemented models:

- · Bagging Classifier
  - Linear SVC
  - Gaussian Naive Bayes
  - Decision Tree
  - Logistic Regression
- Gradient Boosting Classifier
- Random Forests Classifier

The code snippet for implementing one of the ensembling methods for multilabel classification, is depicted in Figure 6.68.

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.multiclass import OneVsRestClassifier

boostClassifier = OneVsRestClassifier(GradientBoostingClassifier())
boostClassifier.fit(vectorised_train_documents, train_labels)
```

Figure 6.68: Code Segment for Gradient Boosting Classifier

6.2.3.4.5 Combination of different models for approach 02

The models which showed highest micro-f1 scores for each of the key features, GPA, technical skills, and project keywords were saved. Then a function was written to combine the individual predictions of the three models and the code snippet for the function can be found in Figure 6.69.

```
def combined_output(gpa, TS , KW):
  gpa_output = np.array (gpa_model.predict(gpa).toarray())
  TS_output = np.array(TS_model.predict(TS))
  KW_output = np.array(KW_model.predict(KW))
  output = gpa_output.astype(int) & TS_output & KW_output
  return output
```

Figure 6.69: Code Segment for Combining GPA, Technical Skills, and Project Keyword Models

Then the loaded datasets were transformed appropriately to be fed into the models. Then the transformed data are fed into the function to get the final combined prediction. The code segment for this process is shown in Figure 6.70.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import metrics
import pickle

# loading vectorizers
kw_vectorizer= pickle.load(open("//content/drive/MyDrive/FYP-mod3/Approach 2/PROJECT KEYWORDS/KW_vector.pickel", "rb"))
TS_vectorizer= pickle.load(open("/content/drive/MyDrive/FYP-mod3/Approach 2/Technical skills/TS_vector.pickel", "rb"))

# transforming technical skills and project keywords
vectorised_kw_test_documents = kw_vectorizer.transform(arr_KW)
vectorised_ts_test_documents = TS_vectorizer.transform(arr_TS)
vectorised_ts_test_documents=vectorised_ts_test_documents.toarray()
vectorised_kw_test_documents=vectorised_kw_test_documents.toarray()

# reshaping gpa data
arr_gpa = arr_gpa.to_numpy().reshape(-1, 1)

# calling combining function
pred = combined_output(arr_gpa, vectorised_ts_test_documents ,vectorised_kw_test_documents)
```

Figure 6.70: Code for Transforming Data and Predicting the Company List using Combined Model

6.2.3.4.6 Neural Network model implementation for approach 03

Under this approach, to make the prediction on which companies an applicant has the possibility of getting shortlisted, multiple of significant features of a resume such as GPA and frequencies of word counts appearing under selected criteria which are depicted in Figure 6.71 were used. These features were extracted by aid of NER developed in module 4.

```
['PROFILE DESCRIBING ADJ', 'DEGREE', 'GPA', 'GRADUATION YEAR', 'SCHOOL',
 'AL STREAM', 'PROGRAMMING LANGUAGES (TECH SKILLS)',
 'DATABASE (TECH SKILLS)', 'WEB DEVELOPMENT (TECH SKILLS)',
 'BACKEND SERVICES', 'MOBILE APP DEVELOPMENT (TECH SKILLS)',
 'PROJECT MANAGEMENT TOOL (TECH SKILLS)', 'IDE (TECH SKILLS)',
 'VERSION CONTROLLING (TECH SKILLS)', 'PERSONAL SKILLS', 'INTERESTS',
 'ROLE (EXTRA CURRICULAR)', 'ORGANIZATION (EXTRA CURRICULAR)',
 'ACHIEVEMENT TYPE', 'COMPETITION / EVENT', 'PROJECT KEYWORDS',
 'OTHER TECHNOLOGIES', 'one_hot_labels'],
```

Figure 6.71: Features Considered for Approach 03

First the dataset prepared was split into train and test sets and data batches were generated. Then the model is defined. The model used for this approach is a neural network and model assembling code can be found in Figure 6.72. The input layer of the model is defined as a fully connected (Dense) layer of 200 layers. As model assembling is defined as a procedure, varying number of input features can be fed to the model.

This essentially means the number of input features considered among the selected feature lists can be changed to carry out different experiments. In the output layer, it is defined as a 200 layer fully connected layer which outputs 10 features (corresponding to 10 company labels). In between the input layer and output layer there exists two dense layers of 200 layers each and in between each dense layer, Rectified Linear Unit (ReLU) activation function, batch normalization, and a dropout rate of 0.2 is applied. After the final output dense layer sigmoid activation is applied.

```
def assemble_model(input_size, output_size):
  inputs = Input(shape=(input_size,))

  nn = Dense(200)(inputs)
  nn = Activation('relu')(nn)
  nn = BatchNormalization()(nn)
  nn = Dropout(0.2)(nn)
  nn = Dense(200)(nn)
  nn = Activation('relu')(nn)
  nn = BatchNormalization()(nn)
  nn = Dropout(0.2)(nn)
  nn = Dense(200)(nn)
  nn = Activation('relu')(nn)
  nn = BatchNormalization()(nn)
  nn = Dropout(0.2)(nn)
  nn = Dense(output_size)(nn)
  nn = Activation('sigmoid', name='company_probability_output')(nn)

  model = Model(inputs=inputs, outputs=[nn], name='company_probability_model')
  return model
```

Figure 6.72: Code Segment for Assembling the Model

After assembling the model, it was compiled by defining the optimizing algorithm as the Adams optimizer. Thereafter it was trained under 100 epochs and given batch size as 16. Within 100 epochs, the model had converged, and the binary cross entropy graph and overall loss graph plotted against the number of epochs are illustrated in Figure 6.73 and Figure 6.74.

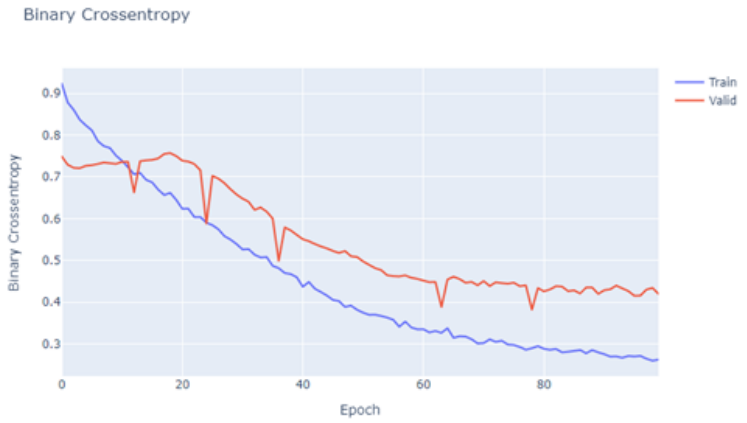


Figure 6.73: Binary Cross entropy Graph for Approach 03 Model

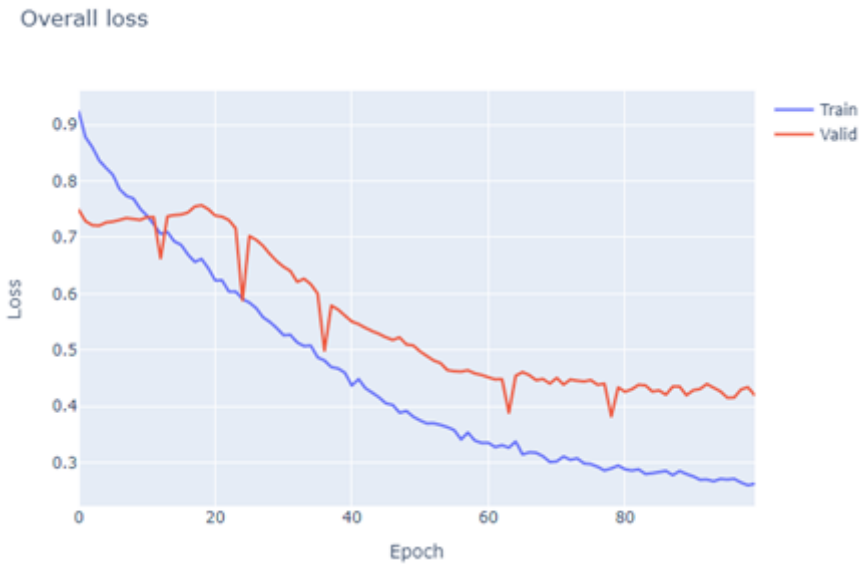


Figure 6.74: Overall Loss Graph for Approach 03

Then the compiled model is trained with the train dataset and the model giving the highest scores is saved. Then the model is evaluated with the test set prepared.

6.2.3.4.7 Question suggestion system

By utilizing the NER developed under module 04, the technical skills mentioned in a resume are extracted. From the user inputs the number of required questions and the difficulty level of the questions are taken and then based on these inputs and captured skills, rules are written to create prompts to be sent for the generative model. Figure 6.75 contains a sample code depicting how the prompt will be generated and sent to the generative model.

```
Tech = "react"
No_of_ques = "10"
Difficulty_level = "intermediate level"
response = openai.ChatCompletion.create(
  model="gpt-3.5-turbo",
  messages=[
        {"role": "system", "content": "You are a helpful assistant for IT students."},
        {"role": "user", "content": "Give me " + No_of_ques + Difficulty_level + " interview sample questions for " + Tech},
    ]
)
```

Figure 6.75: Sample Code Segment for Generating Prompt

Then the generated response is processed to display the list of questions and a sample of such generated questions can be found in Figure 6.76.

```
1. What's the difference between state and props in React?
2. Explain the virtual DOM in React.
3. How would you debug and solve unexpected behavior in a React component?
4. Can you explain the concept of higher-order components (HOC) in React?
5. What is the importance of keys in React?
6. Can you compare and contrast React's setState() and forceUpdate() methods? When would you use each?
7. How do you use Redux with React? What are the advantages and disadvantages of using Redux?
8. Explain what a controlled component is in React and give an example.
9. Explain the concept of React Hooks and how they differ from using class components.
```

Figure 6.76: Sample of Generated Questions

6.2.4 Module 4 - Analyze Resume Content and Suggest Content Improvements

This module mainly focuses on developing a model for giving a score for the content. To implement a better model for that, separate models were implemented for each section and two separate approaches were considered.

6.2.4.1 Preparing Dataset

Section-wise resume content was provided by module two. Also, there is a separate dataset which contains section-wise scores based on design and content quality. The first step of preparing the dataset was creating a single dataset by mapping the score data and content data. As Figure 6.77 shows, first imported both datasets.

```
import pandas as pd

data = pd.read_csv('Annotated_dataset.csv')
score_data = pd.read_csv('scores.csv')
```

Figure 6.77: Importing datasets

```
profile = []
education = []
achievements = []
extra = []
tech_skills = []
other_skills = []
projects = []
refrees = []
work = []
interests = []

for row_index, row in score_data.iterrows():

    profile_txt = ''
    education_txt = ''
    achievements_txt = ''
    extra_txt = ''
    tech_skills_txt = ''
    other_skills_txt = ''
    projects_txt = ''
    refrees_txt = ''
    work_txt = ''
    interests_txt = ''

    for row_index,data_row in data.iterrows():
        if data_row['Folder'] == row['folder'] and data_row['Id'] == row['ID']:
            if data_row['Section'] == 'Profile':
                profile_txt = profile_txt + ' ' + str(data_row['Text'])
            elif data_row['Section'] == 'Education':
                education_txt = education_txt + ' ' + str(data_row['Text'])
            elif data_row['Section'] == 'Awards Achievements':
                achievements_txt = achievements_txt + ' ' + str(data_row['Text'])
            elif data_row['Section'] == 'Extra Curricular/ Roles and Responsibilities':
                extra_txt = extra_txt + ' ' + str(data_row['Text'])
            elif data_row['Section'] == 'Technology/Technical skills':
                tech_skills_txt = tech_skills_txt + ' ' + str(data_row['Text'])
            elif data_row['Section'] == 'Personal Skills':
                other_skills_txt = other_skills_txt + ' ' + str(data_row['Text'])
            elif data_row['Section'] == 'Projects':
                projects_txt = projects_txt + ' ' + str(data_row['Text'])
            elif data_row['Section'] == 'Refrees':
                refrees_txt = refrees_txt + ' ' + str(data_row['Text'])
            elif data_row['Section'] == 'Work Experience':
                work_txt = work_txt + ' ' + str(data_row['Text'])
            elif data_row['Section'] == 'Interest':
                interests_txt = interests_txt + ' ' + str(data_row['Text'])

    profile.append(profile_txt)
    education.append(education_txt)
    achievements.append(achievements_txt)
    extra.append(extra_txt)
    tech_skills.append(tech_skills_txt)
    other_skills.append(other_skills_txt)
    projects.append(projects_txt)
    refrees.append(refrees_txt)
    work.append(work_txt)
    interests.append(interests_txt)


score_data['profile'] = profile
score_data['education'] = education
score_data['achievements'] = achievements
score_data['extra'] = extra
score_data['tech_skills'] = tech_skills
score_data['other_skills'] = other_skills
score_data['projects'] = projects
score_data['refrees'] = refrees
score_data['work'] = work
score_data['interests'] = interests
```

Figure 6.78: Preparing final dataset by merging two datasets

After that, created a single dataset by combining relevant data rows from both datasets by using the code shown in Figure 6.78.

```
score_data.drop(score_data.columns[[2,3,4,5,6,7,8,9]], axis=1, inplace=True)

score_data.to_csv('annotated_dataset_with_section_scores.csv', index=False)
score_data
```

Figure 6.79: Removing unwanted columns in the dataset

And then, using the code shown in Figure 6.79, the unwanted data columns were removed and finalized the dataset required to train this module and saved it as a csv file for future usage. Figure 6.80 shows the sample of the final dataset.

ID	content_rel_profile	content_rel_edu	content_rel_expe	content_rel_projects	content_rel_technol	section_order	overall_content	comment	profile	education	achievements	extra	tech_skills	other_skills	projects	refrees	work	interests
1	7	8	8	7	8	Okay	7	Well organized layout and design. Needs improvement in content	Dedicated third year undergraduate with strong interpersonal skills and extensive knowledge in the field of Information	EDUCATION B.SC. (HONS.) IN INFORMATION TECHNOLOGY UNIVERSITY OF MORATUWA Level 1 Semester 1 GPA 3.78 Level 1 Semester 2 GPA 3.88	AWARDS & ACHIEVEMENTS Dean's list in Level 1 Semester 2	EXTRA CURRICULAR ACTIVITIES University of Moratuwa Rotaractor - Rotaract Club (2016-Present) Holy Family	TECHNICAL SKILLS Programming Languages - C, Java Databases - MySQL, Oracle Web designing - HTML, JSP,		PROJECTS FILEX-SAMPATH BANK PLC (LEVEL 2) Description: A web based application	Mr. B. H. Sudantha Senior Lecturer Department of Information		
2	7	6	7	8	5	Not Okay	5	Needs improvement in both design and content	ABOUT ME am an enthusiastic , self-motivated, hard working individ- ual and a team player who is seeking for an opportunity to obtain an excellent learning experience as an intern.	READING FOR B.SC. (HONS.) DEGREE IN INFORMATION TECHNOLOGY (Expected 2018) Current GPA: 3.20 (up to semester 3) EDUCATION PORAMADULLA NATIONAL COLLEGE, RIKILLAGASKADA G.C.E (A/L - 2013) - PHYSICAL	FIT CodeRush 2016, HackerRank Competition Participant. AWARDS & ACHIEVEMENTS	EXTRA CURRICULAR ACTIVITIES Committee Member - Dancing society 2015 - 2016 National Regional Dancing Competition First place - 2008	TECHNICAL SKILLS Java, Android JavaScript, HTML/CSS ,PHP Bootstrap MySQL,MSSQL Eclipse, NetBeans		PROJECTS HEALTH CARE ANDROID APPLICATION (Ongoing project) This application is developing for IEEEmadC competi- tion. This is for	NON-RELATED REFEREES Dr. Lochandaka Ranathunga Head Department of Information		
3	7	5	7	7	5	Okay	6	Needs improvement in both content and layout	ABOUT ME Highly motivated responsible and a hard working dedicated undergraduate who wants to gain more experience to become a qualified professional in	EDUCATION READING FOR B.SC. (HONS.) IN INFORMATION TECHNOLOGY UNIVERSITY OF MORATUWA (Expected 2018) DHARMASOKA COLLEGE, AMBALANGODA G.C.E (A/L - 2011) - BIOLOGY STREAM	AWARDS & ACHIEVEMENTS FIT CodeRush 2016, HackerRank Competition Awarded by certificate Colurs	EXTRA CURRICULAR ACTIVITIES Committee Member - AIESEC Colombo South 2015-present University of Moratuwa Hockey Team 2014 - Present Member	TECHNICAL SKILLS Java, C, C# JavaScript, AngularJS, Angular2 XHTML, HTML5, CSS Bootstrap MySQL, MongoDB Git GIMP, Photoshop Eclipse,	NON-TECHNICAL SKILLS Creative Thinking Leadership Skills	PROJECTS REMOTE MARINER An under water camera that can be controlled by a remote controller and get the visual feedbacks.	NON-RELATED REFEREES Mr. B. H. Sudantha Senior Lecturer Department of Information		

Figure 6.80: Sample of prepared dataset

6.2.4.2 Score Distribution of the Dataset

Figure 6.81, 6.82, 6.83 and 6.84 show the score distribution for each sections.

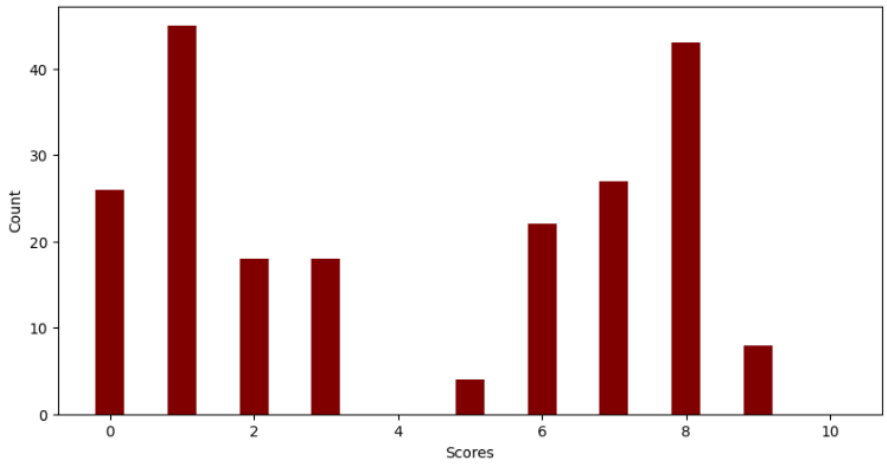


Figure 6.81: Profile Section Score Distribution

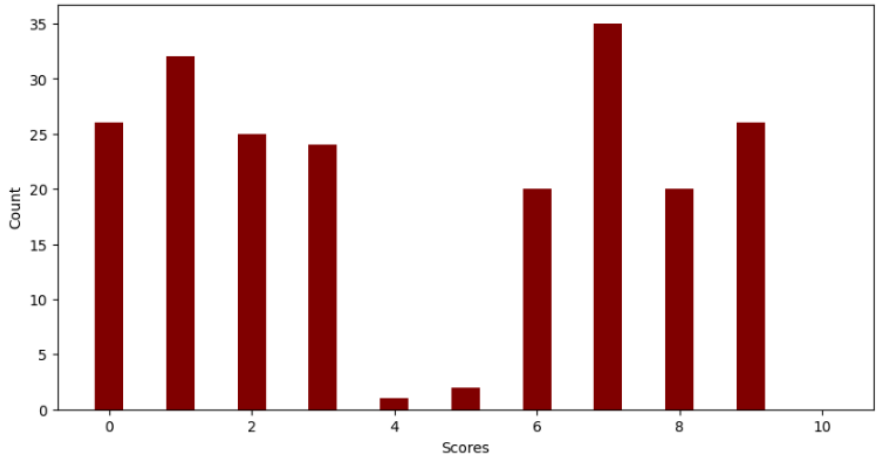


Figure 6.82: Education Section Score Distribution

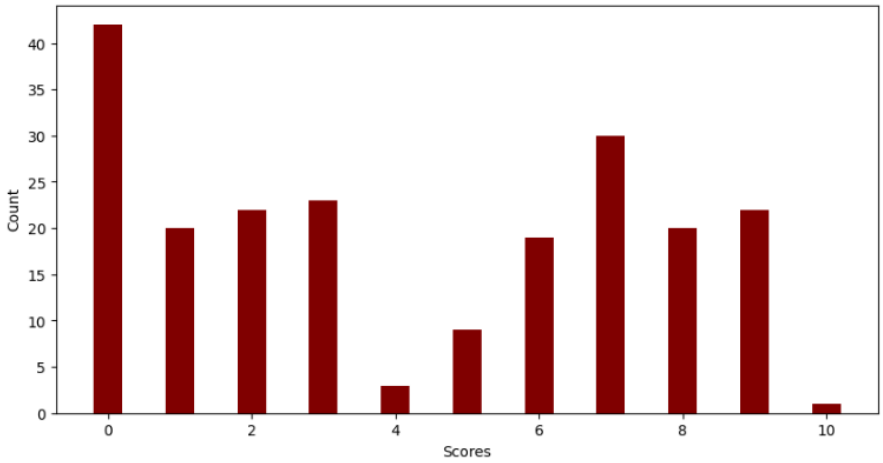


Figure 6.83: Projects Section Score Distribution

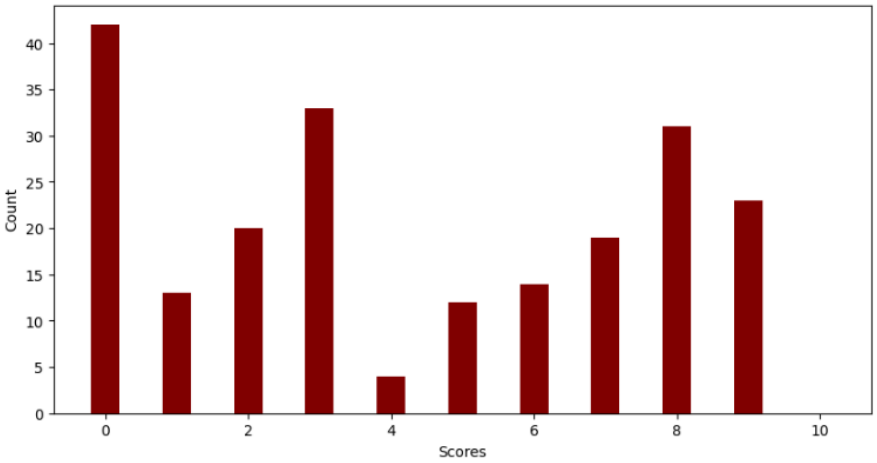


Figure 6.84: Technical Skills Section Score Distribution

6.2.4.3 Approach 01 for Content Scoring

This approach analyzes and provides scores for major sections based on the overall content of each section.

6.2.4.3.1 Data Preprocessing

When implementing the model, as a first step it is required to preprocess the created dataset for further usage. To implement the model Google Colaboratory was utilized.

	score	text
0	7	Dedicated third year undergraduate with stron...
1	7	ABOUT ME am an enthusiastic , self-motivated...
2	7	ABOUT ME Highly motivated responsible and a ...
3	8	PROFILE An enthusiastic undergraduate intent...
4	7	A highly skilled Software Engineer with five ...

Figure 6.85: Sample dataset for profile content along with score

As the initial step, all the required libraries were imported, and the finalized dataset was imported. Then removed all the unnecessary columns except the column containing the profile section content and the column containing the score for that content. Figure 6.85 shows the head of that dataset used for the profile section.

```
df = df.replace({r'(?i)(\||1)\s?am': 'I am'}, regex=True)
df = df.replace({r'(?i)(ABOUT)?(ME)?(PROFILE)?': ''}, regex=True)
df['text'] = df['text'].str.strip()
df['score'] = df.score.astype(int)
```

```
def clean_text(text):
    text = str(text).lower()
    text = text.replace(u'\xa0', u' ')
    return text

df['text'] = df.text.apply(clean_text)
```

Figure 6.86: Cleaning dataset

After that, as shown in Figure 6.86, the dataset was cleaned further by removing the section title, and additional spaces contained in the dataset. In the current implementation, BERT transformer was utilized, hence, commonly used preprocessing steps like stemming and lemmatization were not applied to the dataset. In addition to that, the score was converted to an integer.

6.2.4.3.2 Feature Extraction

As a first approach, CountVectorizer was used which transforms text content into vectors by considering the frequency of each word occurring in the entire dataset. Since it considers the frequency of the words, it couldn't capture the contextual meaning of the words. When it comes to giving a score for text content, having contextual meaning could increase the accuracy of the score. For example, when it comes to the profile section, someone can introduce themself in a more unique way which is eligible to have a higher score. But when only considering the word frequencies, that content could get a low score due to the fact that the words used in that content have less frequencies than the words used in the contents which got more score. Therefore, it is identified that the contextual meaning also plays a major part when giving a score for text content.

Through the research, it was identified that Bidirectional Encoder Representations from Transformers (BERT) is more suitable for the implementation of this model. It is a pre-trained neural model which follows transfer learning mechanism and produces word embeddings which could be used as features. BERT uses Transformer, an attention mechanism that recognizes the relationships between words in a text based on contextual meaning.

```
from transformers import BertTokenizer, DataCollatorWithPadding
from torch.utils.data import DataLoader

BASE_MODEL = "bert-base-uncased"
LEARNING_RATE = 2e-5
MAX_LENGTH = 300
BATCH_SIZE = 8
EPOCHS = 40

tokenizer = BertTokenizer.from_pretrained(BASE_MODEL)
encoded_corpus = tokenizer(text=df.text.tolist(),
                           add_special_tokens=True,
                           padding='max_length',
                           truncation='longest_first',
                           max_length=MAX_LENGTH,
                           return_attention_mask=True)
input_ids = encoded_corpus['input_ids']
attention_mask = encoded_corpus['attention_mask']
```

Figure 6.87: Implementation of BERT tokenizer

The model is implemented based on BERT regression. Therefore, at the feature extraction step, BertTokenizer was used to produce word embeddings. Figure 6.87 shows the implementation of the BertTokenizer. In this model the 'bert-base-uncased' model is used as the base model.

After getting word embeddings, the next step was to split the dataset into three separate datasets as train, validation, and test. Figure 6.88 shows the implementation of splitting the dataset.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

test_size = 0.2
seed = 42

scaler = StandardScaler()
labels = scaler.fit_transform(np.array(df.score.to_list()).reshape(-1, 1).astype(float))
test_labels = scaler.transform(np.array(df_test.score.to_list()).reshape(-1, 1).astype(float))

train_inputs, validation_inputs, train_labels, validation_labels, train_masks, validation_masks, = train_test_split(input_ids, labels, attention_mask, test_size=test_size, random_state=seed)

print(len(train_labels), len(validation_labels), len(test_labels))
```

Figure 6.88: Splitting dataset into train, test and evaluation datasets

6.2.4.3.3 Regression Model for section content scoring

After the feature extraction, the next step was to implement the regression model. For this, BertRegressionModel was implemented with an additional sequential layer. Figure 6.89 shows the code for BertRegressionModel.

```
import torch.nn as nn
from transformers import BertModel

class BertRegressionModel(nn.Module):
  def __init__(self, drop_rate=0.2, freeze_camembert=False):
    super(BertRegressionModel, self).__init__()
    D_in, D_out = 768, 1

    self.bert = BertModel.from_pretrained(BASE_MODEL)
    self.regressor = nn.Sequential(
        nn.Dropout(drop_rate),
        nn.Linear(D_in, D_out)
    )

  def forward(self, input_ids, attention_masks):
    outputs = self.bert(input_ids, attention_masks)
    class_label_output = outputs[1]

    outputs = self.regressor(class_label_output)

    return outputs

model = BertRegressionModel(drop_rate=0.2)
```

Figure 6.89: BertRegressionModel

In this model a Dropout layer was added to prevent overfitting the model for training dataset. Then a regression layer was added to predict the score by identifying the relationship between features and scores in the training dataset. Figure 6.90 shows the architecture of the Bert Regression Model that was created.

```
BertRegressionModel(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (regressor): Sequential(
    (0): Dropout(p=0.2, inplace=False)
    (1): Linear(in_features=768, out_features=1, bias=True)
  )
)
```

Figure 6.90: Architecture of the BERT Regression Model

The next step was to train the model using the dataset. Code for the model training is shown in Figure 6.91. The model was trained using 40 epochs. Mean Squared Error Loss function (MSELoss) was used as the loss function which measures the mean squared error between predicted score and actual score.

```
def train(model, optimizer, scheduler, loss_function, epochs, train_dataloader, device, clip_value=2):
    current_epoch = 1
    for epoch in trange(epochs):
        best_loss = 1e10
        running_loss = 0.0
        running_count = 0
        model.train()
        for step, batch in enumerate(train_dataloader):
            batch_inputs, batch_masks, batch_labels = tuple(b.to(device) for b in batch)
            model.zero_grad()

            outputs = model(batch_inputs, batch_masks)
            loss = loss_function(outputs.squeeze(), batch_labels.squeeze().to(outputs.dtype))

            loss.backward()
            clip_grad_norm(model.parameters(), clip_value)

            optimizer.step()
            scheduler.step()

            running_loss += loss.item()
            running_count += 1

        training_loss.append(running_loss/running_count)
        print(f"Training Loss: {running_loss/running_count}")

        valid_loss = 0.0
        valid_count = 0
        model.eval()
        for step, batch in enumerate(validation_dataloader):
            batch_inputs, batch_masks, batch_labels = tuple(b.to(device) for b in batch)
            model.zero_grad()

            outputs = model(batch_inputs, batch_masks)
            loss = loss_function(outputs.squeeze(), batch_labels.squeeze().to(outputs.dtype))

            valid_loss += loss.item()
            valid_count += 1
```

```
        validation_loss.append(valid_loss/valid_count)
        print(f"Validation Loss: {valid_loss/valid_count}")

        epoch_nos.append(current_epoch)
        current_epoch += 1

    print(epoch_nos)
    print(training_loss)
    print(validation_loss)
    return model

total_steps = len(train_dataloader) * EPOCHS
scheduler = get_linear_schedule_with_warmup(optimizer, num_warmup_steps=0, num_training_steps=total_steps)

model = train(model, optimizer, scheduler, loss_function, EPOCHS, train_dataloader, device, clip_value=2)
```

Figure 6.91: Model training function

Figure 6.92, 6.93, 6.94 and 6.95 show how training and validation loss changed through the training of the model for each section. It clearly shows that the training loss was reduced by each epoch.

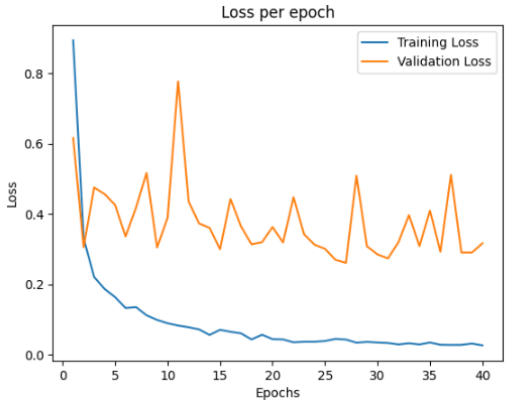


Figure 6.92: Change of the training and validation losses for Profile section

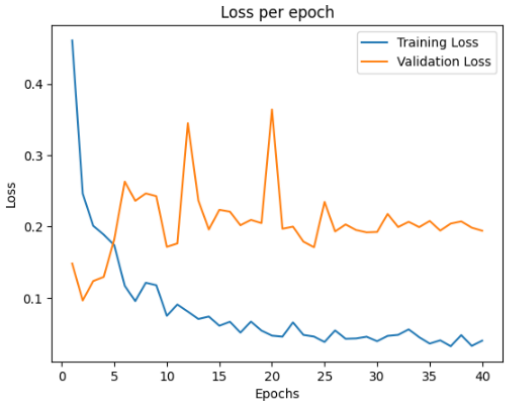


Figure 6.93: Change of the training and validation losses for Education section

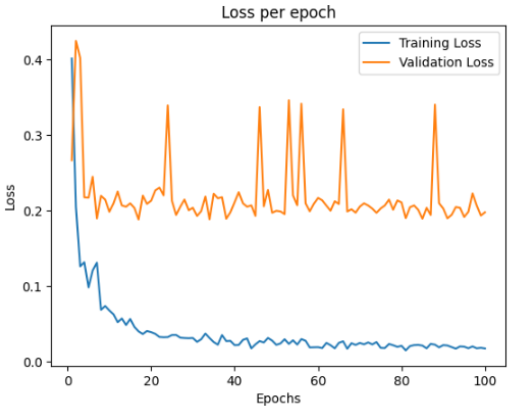


Figure 6.94: Change of the training and validation losses for Projects section

Figure 6.95: Change of the training and validation losses for Technical Skills section

Figure 6.96 shows the code for the score prediction of sample contents.

```
test_sentence_arr = ['I\'ma persevering, adaptable, outgoing IT undergraduate who is seeking for an int

input_ids, attention_mask = tokenize_sentence(test_sentence_arr)
dataloader = create_dataloaders(input_ids, attention_mask)

y_pred_scaled = predict(model, dataloader, device)

[y_pred] = scaler.inverse_transform(np.array(y_pred_scaled).reshape(-1, 1)).reshape(1, -1).tolist()
y_pred = list(map(round, y_pred))

print('\nPredicted Score =', y_pred)
```

Figure 6.96: Score prediction for unseen data

6.2.4.4 Named Entity Recognizer (NER) for Feature Extraction

When analyzing resume contents, considering the specific information in the resume is very important. A Named Entity Recognizer (NER) was created in order to identify and extract particular features from the resume content. The NER's goal is to locate and extract pertinent information for each section of the resume.

The resume information was labeled using “NER Annotator for spaCy” with predefined labels that matched the various features specified to each section in resumes in order to train the NER. The annotation process involved manually labeling the resume content with the labels to create the labeled dataset for training the NER to extract the relevant features. The labels used to annotate the resume content are as follows.

PROFILE DESCRIBING ADJ
DEGREE
GPA
GRADUATION YEAR
SCHOOL
AL STREAM
PROGRAMMING LANGUAGES (TECH SKILLS)
DATABASE (TECH SKILLS)
WEB DEVELOPMENT (TECH SKILLS)
BACKEND SERVICES
MOBILE APP DEVELOPMENT (TECH SKILLS)
PROJECT MANAGEMENT TOOL (TECH SKILLS)
IDE (TECH SKILLS)
VERSION CONTROLLING (TECH SKILLS)
OTHER TECHNOLOGIES
PERSONAL SKILLS
INTERESTS
PERSON NAME
DESIGNATION
ORGANIZATION
CONTACT NO
ADDRESS
EMAIL

ROLE (EXTRA CURRICULAR)

ORGANIZATION (EXTRA CURRICULAR)

ACHIEVEMENT TYPE

COMPETITION / EVENT

PROJECT KEYWORDS

Figure 6.97 shows the sample of data annotation using the “NER Annotator for spaCy”.

Figure 6.97: Sample Data Annotation Using "NER Annotator for spaCy"

After creating the labeled dataset by manually annotating the resumes, the step was to train the NER to extract the features using unseen resume contents. To train the Named Entity Recognizer (NER) model, the popular natural language processing library, spaCy, was utilized. spaCy provides a convenient and efficient framework for training custom NER models.

```
from tqdm import tqdm
import spacy
from spacy.tokens import DocBin

# #nlp = spacy.blank("en") # load a new spacy model
nlp = spacy.load("en_core_web_sm") # load other spacy model

db = DocBin() # create a DocBin object

for text, annot in tqdm(train_data): # data in previous format
    doc = nlp.make_doc(text) # create doc object from text
    ents = []
    for start, end, label in annot["entities"]: # add character indexes
        span = doc.char_span(start, end, label=label, alignment_mode="contract")
        if span is None:
            print("Skipping entity")
        else:
            ents.append(span)
    doc.ents = ents # label the text with the ents
    db.add(doc)

os.chdir(r'D:\FIT Academic\L4\FYP\NER\New folder')
db.to_disk("./train.spacy") # save the docbin object
```

Figure 6.98: Data Conversion to .spacy Format

Prepared dataset should be converted into spaCy format prior to training the model. A DocBin object must be built in order to store our data before converting it to spaCy format. After adding the example and the entity label to the DocBin object, then iterate through the data and save the object to .spacy file. Figure 6.98 shows the code for data conversion to .spacy format.

```
!python -m spacy download en_core_web_lg
!python -m spacy train "/content/drive/MyDrive/FYP - Clay Squad/CV Reviewer/Dataset/Dataset for module 4/NER/config.cfg"
  --output "/content/drive/MyDrive/FYP - Clay Squad/CV Reviewer/Dataset/Dataset for module 4/NER/output"
  --paths.train "/content/drive/MyDrive/FYP - Clay Squad/CV Reviewer/Dataset/Dataset for module 4/NER/train.spacy"
  --paths.dev "/content/drive/MyDrive/FYP - Clay Squad/CV Reviewer/Dataset/Dataset for module 4/NER/train.spacy" --gpu-id 0
```

Figure 6.99: Code Segment for Train the Custom Pipeline

Figure 6.99 shows the code used for training the custom pipeline. Once the pipeline had been trained, the best model was saved in the output directory. The final model scored 0.99 as the score. This NER model could identify the section specified features successfully and Figure 6.100 shows how successfully this model identify and labels the given sample data content.

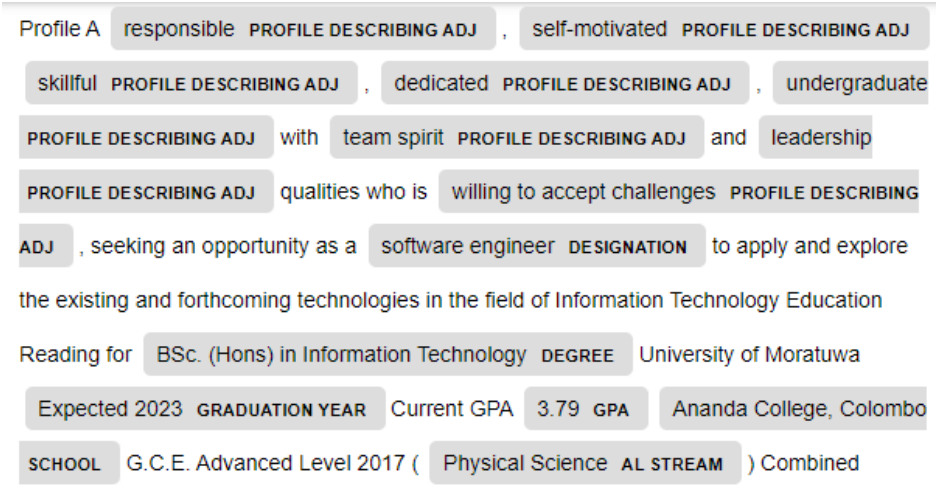


Figure 6.100: Identifying and Labeling the Sample Data Using NER Model

6.2.4.5 Approach 02 for Content Scoring

This approach analyzes and provides scores for major sections based on the section specific features.

6.2.4.5.1 Section Specified Feature Extraction using a Named Entity Recognizer (NER)

The next step involved developing individual models for each section to predict scores for the section content based on the section specified features. The initial phase of this approach focused on feature extraction of each section, utilizing the previously developed NER model. Table 6.1 outlines the specific features considered for each section.

Table 6.1: Specific features considered for each section.

Section	Feature	Value Type
Profile Section	PROFILE DESCRIBING ADJ	No of person describing adjectives present in the section
	Content length	Length of the profile section
	DESIGNATION	Whether the job position that the candidate going to apply is included in the section or not

Education Section	DGREE	Whether the degrees that the candidate currently reading is included in the section or not
	GPA	Value of the GPA if included in the section
	GRADUATION YEAR	Whether the graduation year of the candidate is included in the section or not
	SCHOOL	Whether the school of the candidate is included in the section or not
	AL STREAM	Whether the A/L Stream of the candidate is included in the section or not
Projects Section	PROGRAMMING LANGUAGES (TECH SKILLS)	No of programming languages present in the section
	DATABASE (TECH SKILLS)	No of database technologies present in the section
	WEB DEVELOPMENT (TECH SKILLS)	No of web development technologies present in the section
	BACKEND SERVICES	No of backend services present in the section
	MOBILE APP DEVELOPMENT (TECH SKILLS)	No of mobile app development technologies present in the section
	OTHER TECHNOLOGIES	No of any other technologies present in the section
	PROJECT KEYWORDS	No of project key words present in the section

Technical Skills Section	PROGRAMMING LANGUAGES (TECH SKILLS)	No of programming languages present in the section
	DATABASE (TECH SKILLS)	No of database technologies present in the section
	WEB DEVELOPMENT (TECH SKILLS)	No of web development technologies present in the section
	BACKEND SERVICES	No of backend services present in the section
	MOBILE APP DEVELOPMENT (TECH SKILLS)	No of mobile app development technologies present in the section
	PROJECT MANAGEMENT TOOL (TECH SKILLS)	No of project management tools present in the section
	IDE (TECH SKILLS)	No of IDEs present in the section
	VERSION CONTROLLING (TECH SKILLS)	No of version controlling technologies present in the section
	OTHER TECHNOLOGIES	No of any other technologies present in the section

The features mentioned in above table are some of the major details that should include in each section according to the research findings. Hence, it is important to pay attention to that information.

The feature extraction code segments for each section vary depending on the specific features considered within that section. As an example, the following code segment illustrates feature extraction for the profile section. In this code, the NER model is imported and referred to as "trained_nlp" (see Figure 6.101).

```
df['score'] = df.loc[:, 'content_rel_profile']
df = df.drop(df.columns[[0]], axis=1)
```

```
df['text'] = ''
df['content_len'] = 0
df['adjectives_len'] = 0
df['position'] = 0
```

```
for i in df.index:
  adjectives_len = 0
  if str(df['profile'][i]) == 'nan':
    continue

  doc = trained_nlp(df['profile'][i])

  for ent in doc.ents:
    if ent.label_ == 'DESIGNATION':
      df['position'][i] = 1
    elif ent.label_ == 'PROFILE DESCRIBING ADJ':
      adjectives_len = adjectives_len + 1

      if str(df['text'][i])!='nan':
        df['text'][i] =  df['text'][i] + ' ' + ent.text
      else:
        df['text'][i] = ent.text

  df['content_len'][i] = len(df['profile'][i])
  df['adjectives_len'][i] = adjectives_len
```

Figure 6.101: Feature Extraction Using NER

Similar code segments were developed to extract features from each section by considering the features applying to each section.

Figure 6.102, 6.103, 6.104, 6.105 show the correlation between the features extracted for each section with the score of the sections.

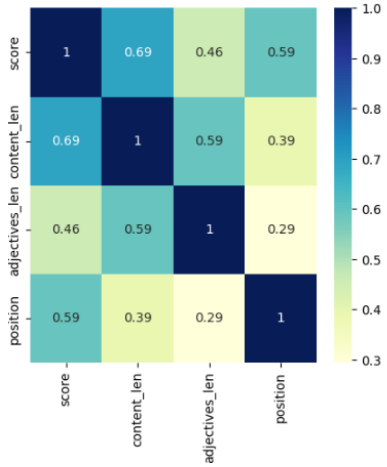


Figure 6.102: Correlation between Score and Features in Profile Section

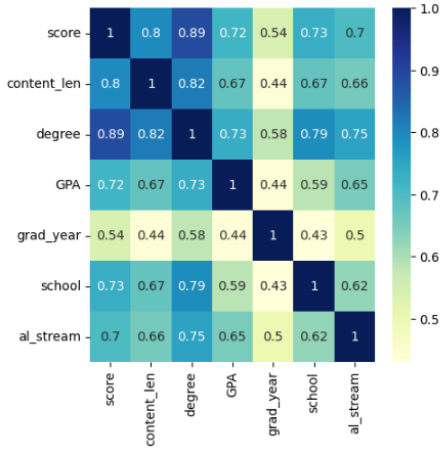


Figure 6.103: Correlation between Score and Features in Education Section

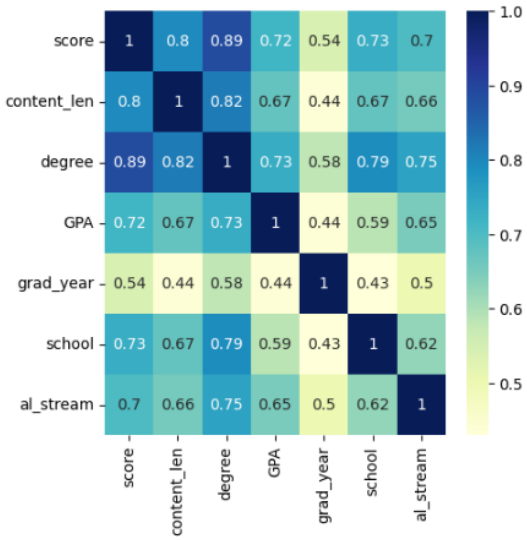


Figure 6.104: Correlation between Score and Features in Projects Section

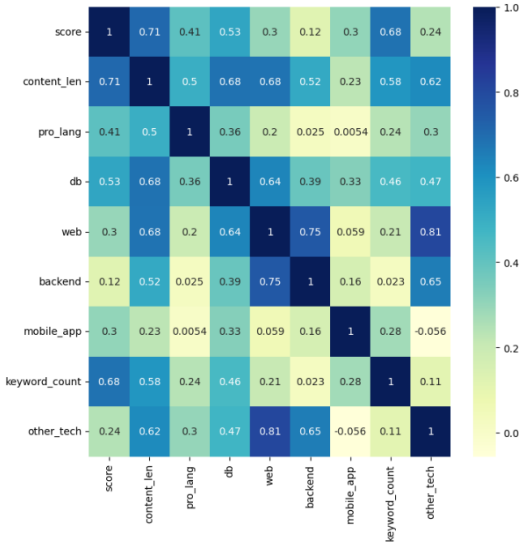


Figure 6.105: Correlation between Score and Features in Technical Skills Section

6.2.4.5.2 Regression Models for section content scoring

Following the feature extraction process, the subsequent step involved training regression models for each section to predict scores for the corresponding section contents. To accomplish this, several regression models were implemented, namely Linear Regression, Decision Tree Regression, Support Vector Regression (SVR), and Random Forest Regression. The objective was to identify the most suitable model for each section. Figure 6.106 showcases the code segment detailing the implementation of these models.

```
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor

models = [LinearRegression(), DecisionTreeRegressor(), SVR(), RandomForestRegressor()]

for m in models:
  print(m)
  m.fit(X_train, y_train)

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from math import sqrt
import pickle
from keras.models import load_model

for m in models:
  y_pred = m.predict(X_valid)
  y_pred = [round(j) for j in y_pred]

  print("")
  print("============================================================")

  print("")
  print("Model\t\t", m)
  print("MAE:\t\t", mean_absolute_error(y_valid.tolist(), y_pred))
  print("RMSE:\t\t", sqrt(mean_squared_error(y_valid.tolist(), y_pred)))
  print("R2 score:\t", r2_score(y_valid.tolist(), y_pred))
```

Figure 6.106: Implementation of the Regression Models

In the provided code segment, the regression models, namely Linear Regression, Decision Tree Regression, SVR, and Random Forest Regression, are instantiated and stored in the models list. Each model is then trained using the training data, X_train and y_train, by calling the fit() method.

Following the training phase, the models are evaluated using several metrics, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared score (R2 score). The predicted values, y_pred, are obtained using the predict() method on the validation data, X_valid. The comparison of each model will be done in the evaluation chapter.

6.2.4.6 Missing Fields and Content Improvement Suggestions

In addition to providing scores for the profile, education, projects and technical skills sections, this solution will provide some suggestions to improve the resume by considering the missing contents or sections and possible content improvements. This was implemented using a rule-based approach based on the features extracted from the NER model.

Figure 6.107, 6.108 show some examples for suggestions provided by the system.

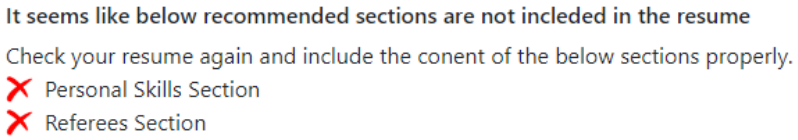


Figure 6.107: Missing Section Suggestions

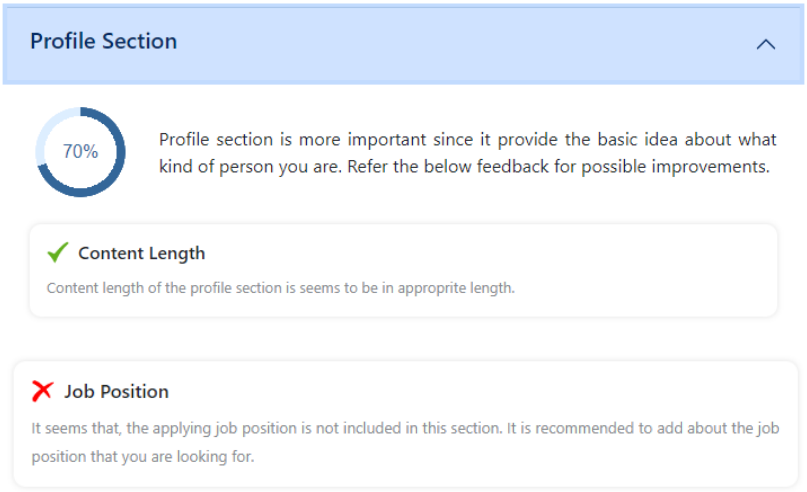


Figure 6.108: Section Content Improvement Suggestions

6.2.4.7 Grammar and Spelling Mistake Checker

It is important to prepare a resume without any grammatical errors. Since there are many existing models for grammar checking purposes, one of them was selected to integrate with the platform. Gramformer is the solution chosen to implement with the platform due to its ability to detect, highlight and correct grammatical mistakes.

6.3 Summary

In summary, the implementations chapter includes detailed information of different experimentations and approaches tried out to implement the proposed resume review system. It elaborates module wise dataset analysis, data preprocessing steps, approaches, algorithm implementations with the aid of relevant code snippets. Under model 01, five subcomponents were implemented to give proper feedback about the graphical layout of the resume design. As implementations under module 02, development of a resume parser was developed to feed module 03 and 04. were carried out. For the module 03, three approaches were implemented to find the most appropriate method to predict the company list, finally scoring and rule-based content feedback module were implemented for module 04.

Chapter 7 – Evaluation

7.1 Introduction

The evaluation chapter aims to assess the performance and effectiveness of the implemented resume review system. This chapter focuses on evaluating the system's ability to analyze and provide feedback on various aspects of resume design, including graphical layout, content relevance, and overall quality. The evaluation process involves applying established evaluation metrics and techniques to measure the system's accuracy, and robustness.

7.2 Module 1 - Analyzing Resume Design

7.2.1 Evaluation of Text contrast ratio identification algorithm

Since the method (a) is ineffective the evaluation has been done only for the method (b). Evaluation has been done using 10 PDFs. From those pdfs, identifiable 99 text styles were extracted as images. Then the background colour and text colour of the text styles were manually annotated. Then using method (b) (colour clustering algorithm) the same task is carried out. Actual contrast ratio of those text styles and predicted contrast ratio of those text styles were used to measure the RMSE and MAE.

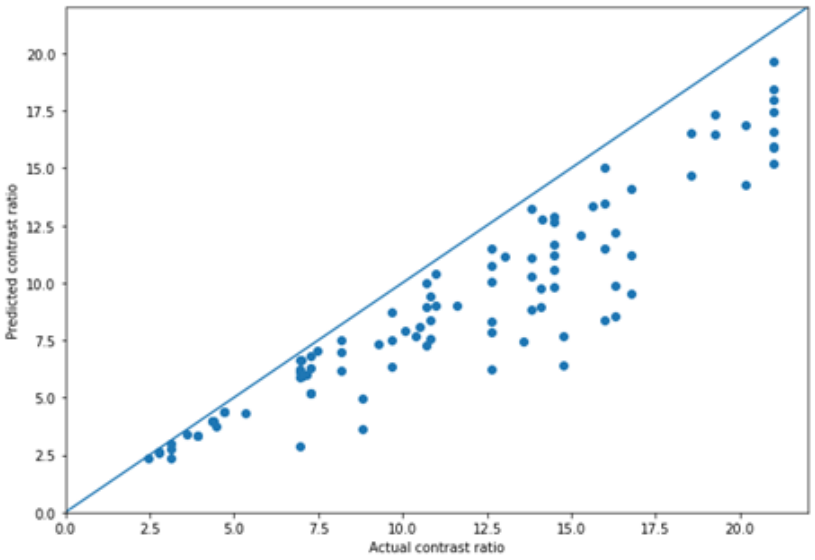


Figure 7.1: Actual contrast ratio vs Predicted contrast ratio graph for method (b) (colors clustering)

As shown in the Figure 7.1 when using colour clustering, higher the contrast ratio it gets more error, contrast ratio is under-predicted. Yet because of the effectiveness of the algorithm and the efficiency, it has been used for the colour contrast ratio identification.

7.2.2 Evaluation of Color combination scoring model

In the evaluation of the colour combination scoring model, the model was assessed using various metrics suitable for regression tasks, including Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R-squared (R2) score. These metrics were employed to measure the performance and accuracy of the model in predicting the colour combination scores.

Mean Absolute Error, as a common evaluation metric, calculates the average absolute difference between the predicted and actual scores. It provides insights into the magnitude of the prediction errors, disregarding their direction. Lower MAE values indicate better accuracy and a smaller average discrepancy between predicted and actual scores.

Root Mean Squared Error, another widely used metric, computes the square root of the average squared differences between predicted and actual scores. It penalizes larger errors more severely than MAE and provides a measure of the average magnitude of the prediction errors. Similar to MAE, lower RMSE values indicate improved accuracy and a reduced average prediction error.

R2 score, also known as the coefficient of determination, measures the proportion of the variance in the target variable that is explained by the model. It ranges between 0 and 1, with higher values indicating a better fit of the model to the data. R2 score provides an indication of how well the model captures the variability in the colour combination scores.

By utilizing these evaluation metrics, the performance of the colour combination scoring model was thoroughly analysed and quantitatively assessed, providing a comprehensive understanding of its predictive capabilities and accuracy in the context of predicting the colour combination score. The figure 7.2 shows the quantitative comparison between the evaluated models with their evaluation metric values.

	10 pallets			25 palletes			39 pallets			Average		
Algorithm	MAE	RMSE	R2	MAE	RMSE	R2	MAE	RMSE	R2	MAE	RMSE	R2
Keras NN (500 Epochs, L. Rate – 1e-4)	1.8	2.24	0.02	1.36	1.897	0.45	1.38	1.85	0.49	1.51	2	0.32
Linear Regression	1.4	1.73	0.41	1.16	1.562	0.63	1.18	1.52	0.66	1.25	1.6	0.57
Decision Tree Regressor	1.9	2.12	0.12	1.76	2.263	0.22	1.64	2.12	0.34	1.77	2.17	0.22
Support Vector Regressor	1.8	2.19	0.06	1.28	1.789	0.51	1.26	1.66	0.6	1.45	1.88	0.39
Random Forest Regressor	1.2	1.55	0.53	1.04	1.442	0.68	1	1.41	0.71	1.08	1.47	0.64

Figure 7.2: Quantitative comparison between the evaluated models with their evaluation metric values

According to the aforementioned comparison, since Random Forest Regressor shows the better results in all MAE, RMSE and R2 score, it has been selected for the designated task.

7.2.3 Evaluation of Layout extraction and scoring model

7.2.3.1 Evaluation of Layout classification model

In the evaluation of the layout classification model, the model underwent a comprehensive assessment considering its nature as an imbalanced multiclass classification task. To address the challenges posed by class imbalance, the evaluation employed the micro-average F1 score as a suitable metric.

The micro-average F1 score calculates the harmonic mean of precision and recall, taking into account the overall performance across all classes while considering the imbalanced nature of the dataset. It treats all instances and predictions as a single collective, providing a measure of the model's ability to classify instances correctly regardless of class distribution.

The classification report for the model is illustrated in figure 7.3. Based on the evaluation results in the report, the layout classification model achieved a **micro-average F1 score of 0.71** for the designated classification task. This score indicates that the model performs favourably in accurately classifying instances, considering the imbalanced nature of the dataset. Hence, the model is considered as a good model for predicting the layout of the resume.

	precision	recall	f1-score	support
0	0.67	0.67	0.67	3
1	0.33	1.00	0.50	1
2	0.00	0.00	0.00	1
3	1.00	0.50	0.67	2
4	0.00	0.00	0.00	0
5	0.89	0.89	0.89	9
6	0.00	0.00	0.00	1
micro avg	0.71	0.71	0.71	17
macro avg	0.41	0.44	0.39	17
weighted avg	0.73	0.71	0.70	17
samples avg	0.71	0.71	0.71	17

Figure 7.3: The classification for the layout classification model

7.2.3.2 Evaluation of Layout scoring model

Similar to section 7.1.2, this model is also evaluated using MAE, RMSE and R2 Score for its suitability for the stated task. The evaluation metric values achieved for three different test sets and the average of all values are illustrated in figure 7.4. According to the illustration Random Forest Regressor is selected for the designated regression task.

	1st prediction			2nd prediction			3rd prediction			Average		
Algorithm	MAE	RMSE	R2	MAE	RMSE	R2	MAE	RMSE	R2	MAE	RMSE	R2
Keras NN (1000 Epochs, L. rate – 1e-4)	1.18	1.73	0.08	1.29	2.09	0.174	1.36	1.76	0.44	1.27	1.86	0.23
Linear Regression	1.38	1.86	-0.07	1.33	2.23	0.056	1.62	2.08	0.21	1.44	2.06	0.07
Decision Tree Regressor	1.4	2.2	-0.48	1.33	2.54	-0.22	1.29	1.71	0.47	1.34	2.15	-0.08
Support Vector Regressor	1.2	1.69	0.13	1.4	2.08	0.178	1.36	1.78	0.42	1.32	1.85	0.24
Random Forest Regressor	1.13	1.71	0.11	1.13	1.73	0.431	1.31	1.72	0.46	1.19	1.72	0.33

Figure 7.4: Quantitative comparison between the evaluated models with their evaluation metric values

7.2.4 Evaluation of First Impression Classification model

Similar to section 7.1.3.1, the first impression classification model is designated for classifying an imbalanced class prediction. Hence, the harmonic mean between precision and recall, which is F1 score, is used for measuring the performance of the model. According to the classification report, the model exhibits **0.83 accuracy** while achieving **0.78 macro averaged F1 score**. The macro-averaged F1 score takes into account the individual F1 scores for each class and provides an unbiased measure of the model's overall effectiveness in capturing both precision and recall. With a macro-

averaged F1 score of 0.78, the model demonstrates a notable capability to handle the challenges posed by class imbalance and achieve commendable performance across all classes.

The classification report and the confusion matrix for the first impression classification model is illustrated in figure 7.5 and figure 7.6 respectively.

	precision	recall	f1-score	support
0	1.00	0.50	0.67	2
1	0.78	0.78	0.78	9
2	0.86	0.92	0.89	13
accuracy			0.83	24
macro avg	0.88	0.73	0.78	24
weighted avg	0.84	0.83	0.83	24

Figure 7.5: The classification report for the first impression classification model

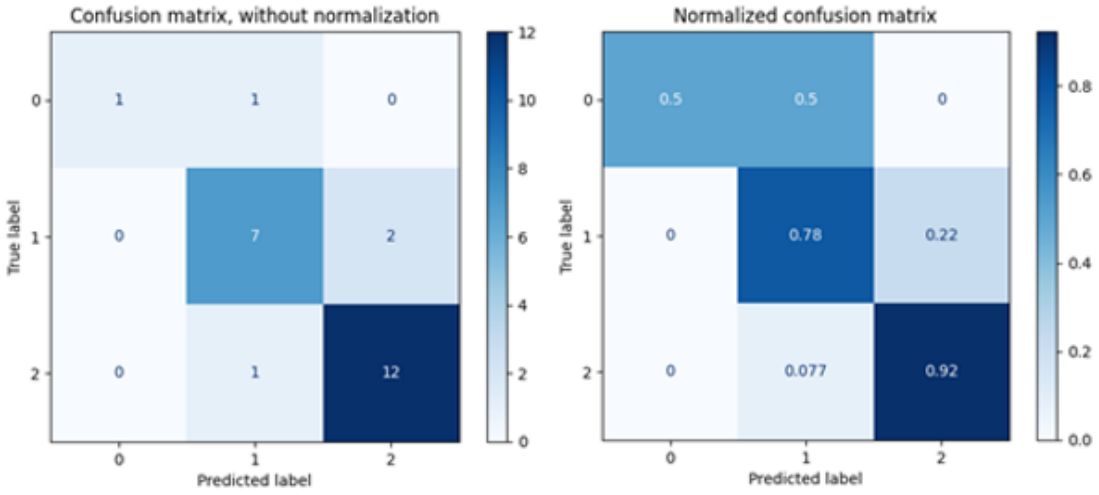


Figure 7.6: The confusion matrix for the first impression classification model (with and without normalization)

7.2.5 Evaluation of Overall design scoring model

In this model, considering its nature as a regression task, previously mentioned error metrics and the coefficient of determination were employed to compare the trained models. While certain models exhibited favourable performance, others yielded less satisfactory results. Among the evaluated models, linear regression emerged as the most promising choice due to its ability to minimize errors and maximize the overall fit. Consequently, linear regression was selected as the preferred approach for predicting

the score associated with the overall design of the resume. The all-evaluation results values are included in the Table 7.1.

Table 7.1: Quantitative comparison between linear regression, decision tree regressor, SVR and RandomForestRegressor for predicting overall design score

Model	MAE	RMSE	R2 score
Linear Regression	0.7652	0.8973	0.7071
Decision Tree Regressor	1.2917	1.5943	0.0752
SVR	1.2342	1.4043	0.2824
Random Forest Regressor	1.0292	1.2585	0.4237

7.3 Module 2 - Resume Parser

7.3.1 Evaluation of Multiclass Classification Model for Resume Section Prediction: F1-Score Analysis

The system used the F1-Score assessment metric to evaluate the accuracy with which its multiclass classification model performed in predicting each section of a resume. The F1-Score provides an accurate evaluation of the model's classification accuracy for each section by taking into consideration both precision and recall. The system will be able to evaluate the model's performance across many areas, such as profile, education, projects, referees, awards & responsibilities, and interests by calculating the F1-Score for each section separately. This approach allowed us to gain insights into the strengths and weaknesses of the model in accurately predicting each section. The F1-Score evaluation provides a quantitative measure to evaluate the overall effectiveness of the multiclass classification model.

The system has been experimented with a few different classifiers such as Multinomial Naive Bayes, Gaussian Naive Bayes and Support Vector Machine (SVM) with SGDClassifier. Then the F1-scores are calculated for each section predicted by both models. Upon analysing the results, it became evident that the SVM with SGDClassifier consistently outperformed the Multinomial Naive Bayes classifier in terms of F1-scores for each section. As shown in Table 7.2, the SVM with SGDClassifier exhibited higher

accuracy, precision, and recall values, resulting in superior overall F1-scores across the different resume sections.

Table 7.2: F1-score values for each section using SVM and NaiveBayes models

Section	SVM (F1-Score)	NaiveBayes (F1-Score)
Profile	0.90	0.80
Education	0.91	0.82
Projects	0.92	0.90
Referees	0.97	0.98
Extra Curricular/ Roles and Responsibilities	0.79	0.76
Awards Achievements	0.85	0.85
Interest	0.88	0.78
Personal Skills	0.88	0.73
Technology/Technical skills	0.94	0.91
Work Experience	0.39	0.00

In addition to that, the accuracy of both models as shown in the table 7.3.

Table 7.3: Accuracy of SVM and NaiveBayes models

Model	Accuracy obtained
SVM with SGDClassifier	0.8881431767337807
Multinomial NaiveBayes	0.8400447427293065

7.3.2 Cosine Similarity Analysis for Section Content Comparison

The cosine similarity metric is used to evaluate how closely the extracted relevant section to the actual section content in order to evaluate the system's accuracy and efficacy. As depicted in figure 7.7 and 7.8, the system determined the cosine similarity score for a set of resumes by contrasting the textual representation of the extracted section content with the actual section content. The system then calculates the average values of cosine for each section. The cosine similarity takes into account the direction and magnitude of the vectors representing the two texts, providing a measure of their similarity.

Section	Actual	Predicted
Profile	An enthusiastic and passionate individual v	an enthusiastic and passionate i
Education	B.Sc(Hons.) in Information Technology Univ	no 230 1st step nawamalkaduwa
Technical Skills	C Java HTML JavaScript ReactJS NodeJS SQL	c java html javascript reactjs noc
Personal Skills	Good communication Team work Leadershi	good communication leadershi
Interests	Animal Welfare Hand crafting Volunteering	travelling gaming interest
Awards and Responsibilities	HackX Inter-University Startup Challenge 2(	www.linkedin.com in dini thi ar
Project	INSFRA SMART OFFICE A software system w	http github.com dinzie95 http t
Referees	Dr. (Mrs.) G. U Ganegoda Senior Lecturer, F	94 71 380 6807 +94 77 728 6807 d
Profile	A committed hard-working third year unde	objective a committed hard wo
Education	UNIVERSITY OF MORATUWA (2017-PRESENT	education overall gpa 3.46 univ
Technical Skills	Programming Languages Java, C Databases	technical skill programming lan
Personal Skills	Project management Complex problem sol	interpersonal skill project mana
Interests	Volunteering Blogging Reading Swimming	interest volunteering blogging
Awards and Responsibilities	Student Representative (2017-2018) Memb	linkedin linkedin.com in nethm
Project	A SOFTWARE SYSTEM FOR BUSINESS ANALY	project project a web based app
Referees	Dr. G. Upeksha Ganegoda Senior Lecturer D	phone +94 76 993 2442 +94 11 28

Figure 7.7: Content used for calculating cosine similarity

```
average = sum(cosinesArr)/len(cosinesArr)
print("\nAverage Cosine Value for Profile Section: "+ str(average))
```

```
similarity:  0.9333333333333333
similarity:  0.9574271077563381
similarity:  0.9230769230769231
similarity:  0.8295150620062532
similarity:  0.8888888888888888
similarity:  0.9181561700975341
similarity:  0.8236877675803729
similarity:  0.8467803948114511
similarity:  0.9146591207600472
similarity:  0.8125

Average Cosine Value for Profile Section: 0.8848024768311141
```

Figure 7.8: Obtained average cosine value for profile section

Table 7.4 shows the average cosine value for all the sections which is calculated using the actual and predicted section content from the system.

Table 7.4: Average cosine value for each section

Section	Average Cosine Value
Profile	0.8848024768311141
Education	0.7301637245094814
Projects	0.7836110811709401
Technical Skills	0.7353279814496185
Personal Skills	0.8700204069920932
Interests	0.740757026121876
Awards and Responsibilities	0.7716714248985228
Referees	0.8270679024187793

7.4 Module 3 - Company Recommendation System and Interview Questions Suggestion

Multi-label classification differs from single-label classification in that it allows instances to be partially correct rather than simply classified as correct or incorrect. Consequently, multi-label classification necessitates the use of distinct metrics compared to traditional single-label classification. Further they should have the capability to handle the label imbalance and give a thorough grasp of the model's predicting capabilities (multi label problem transformation methods).

Micro measures are better suited over macro measures in multilabel classification evaluations as macro measures are heavily affected by minority classes (Addressing imbalance in multilabel classification: Measures and random resampling algorithms). Hence, micro-f1 score can be utilized as the most suitable evaluation metric for muti label classification in this module. The importance of the Micro f1-score in multi-label classification resides in its capacity to handle imbalanced label distributions effectively

and to assess the classifier's overall capacity to capture relevant labels across all instances. It is especially helpful when individual labels are not the primary focus, but rather the classifier's overall performance.

Another metric that can be used to get an idea on overall performance of multi-label classification tasks is subset accuracy. In this metric, to achieve a correct prediction for a sample in multi-label classification, the set of predicted labels must precisely match the set of labels in true labels (Ensemble Methods for Multi-label Classification).

This section analyzes the obtained results with the aid of the selected metrics and compares them across the different models implemented. The obtained results are discussed and analyzed using accuracy and micro F1 score as the chosen evaluation metrics. The accuracy metric is employed to measure the proportion of correctly classified instances out of the total number of instances. By utilizing accuracy, we evaluate the overall correctness of the model's predictions in terms of classifying instances correctly. The micro-f1 score is utilized as a performance metric in this evaluation as accuracy is a very strict measure and accuracy alone will not correctly represent a model's performance. The micro-f1 score takes into account both precision and recall at the micro-average level, aggregating the counts across all classes. This metric is particularly useful when dealing with imbalanced datasets or when equal importance is assigned to each class. Essentially, the micro-f1 score emphasizes the model's ability to correctly identify the positive labels.

Since, by nature, multi-label classification models include a high number of ‘0’ labels compared to 1 label, measuring the model’s performance through micro-f1 score gives a better understanding on the prediction capability of the model.

7.4.1 Evaluations for approach 01 - analyzing whole text content of resumes

7.4.1.1 Evaluations for BERT model

The trained model was evaluated against an unseen dataset and the accuracy score and micro-f1 score obtained can be found in Table 7.5.

Table 7.5: Micro-F1 Score and Accuracy Score for BERT Model

Micro - f1 score	0.0444
Accuracy score	0.1935

These results indicate that the model's performance is relatively poor in terms of accurately predicting the labels for the given dataset.

7.4.1.2 Evaluations for problem transformation methods, algorithm adaptation methods, and ensemble methods

The table 7.6 show the different micro-f1 scores and accuracies obtained for the models implemented under different traditional multi-label classification approaches (problem transformation methods, algorithm adaptation methods, and ensemble methods), for whole text content of resumes.

Table 7.6: Comparison of results obtained for whole text content through problem transformation methods, algorithm, adaption methods, and ensemble methods

		Micro-f1 score	**Accuracy score**
Problem transformation Methods			
Binary relevance	**Linear SVC**	0	0.3548
	GaussianNB	0	0.3548
	Logistic regression	0	0.3548
	Decision Tree	0.0833	0.1612
Classifier chain	**Linear SVC**	0	0.3226
	GaussianNB	0	0.3548
	Logistic regression	0	0.3548
	Decision Tree	0.1276	0.2258

label powersets	**Linear SVC**	0.1081	0.3226
	GaussianNB	0	0.3226
	Logistic regression	0	0.3548
	Decision Tree	0.0667	0.0645
Algorithm adaptation Methods			
k-Nearest Neighbor		0	0.3548
Decision tree		0.0357	0.1612
Ensemble Methods			
Bagging	**Decision Tree**	0.0741	0.3226
	Linear SVC	0	0.3548
	Logistic Regression	0	0.3548
	GaussianNB	0	0.3548
Gradient boosting		0.1379	0.3548
Random forest		0	0.3548

The majority of the implemented models demonstrated micro-f1 scores of zero even though they showed a certain level of accuracy. The accuracy scores here have been generated due to the existence of label sets of all zeros in true labels and having prediction labels of having all zeros in predicted labels. Thereby, these accuracy scores do not provide accurate representation of predicting capability of the model. In contrast, micro-f1 scores tell us about the capability of the model in predicting positive labels, which is more meaningful for our problem. The majority of the models had an micro-f1 score of '0', meaning that they haven't predicted any of the positive labels correctly. Decision trees have shown a micro-f1 score for under all implemented methods, still it is a very low value.

7.4.1.3 Comparison of different models implemented for approach 01

Upon evaluation, the BERT model achieved an accuracy score of 0.1935 and a micro-f1 score of 0.0444. From the traditional multi-label classification models, the maximum micro-f1 score of 0.1379 was shown by the gradient boosting model and its' accuracy was 0.3548. The traditional model, gradient boosting model, outperformed the BERT model in terms of both accuracy and micro-F1 score. Further it is observed that compared to other models, decision trees have shown a notable micro-f1 score. Still, all these obtained micro-f1 scores and accuracy scores for different models under approach 01 are very low, indicating poor prediction capabilities.

7.4.2 Evaluations for approach 02

7.4.2.1 Evaluations for problem transformation methods, algorithm adaptation methods, and ensemble methods for GPA

The table 7.7 show the different micro-f1 scores and accuracies obtained for the models implemented under different traditional multi-label classification approaches (problem transformation methods, algorithm adaptation methods, and ensemble methods), for GPA.

Table 7.7: Comparison of results obtained for GPA through problem transformation methods, algorithm, adaption methods, and ensemble methods

		Micro-f1 score	**Accuracy score**
Problem transformation			
Binary relevance	**Linear SVC**	0	0.25
	GaussianNB	0	0.2375
	Logistic regression	0	0.25
	Decision Tree	0.1184	0.1125
Classifier chain	**Linear SVC**	0	0.25

	GaussianNB	0.1398	0
	Logistic Regression	0	0.25
	Decision Tree	0.1429	0.1
Label powersets	**Linear SVC**	0	0.25
	GaussianNB	0	0.225
	Logistic regression	0	0.25
	Decision Tree	0.1734	0.1
Algorithm adaptation			
k-Nearest Neighbor		0.0721	0.2
Decision tree		0.1184	0.1125
Ensemble			
Bagging	**Decision Tree**	0.068	0.075
	Linear SVC	0	0.25
	Logistic regression	0	0.25
	GaussianNB	0	0.2375
Gradient boosting		0.1029	0.1375
Random forest		0.1358	0.1

The evaluation results obtained when models were trained by feeding only the GPA feature showed similar results as approach 01. The majority of models depicted a micro-f1 score of 0 while decision trees provided some level of micro-f1 score. Further, Gaussian Naive Bayes model implemented through classifier chain methodology has indicated a value for micro-f1 score yet shows an accuracy of 0. This indicates that for individual labels, positive labels have been predicted but when comparing the entirety

of the label set, the predictions are false. Additionally, k-nearest neighbors, gradient boosting, and random forest algorithms have shown some level of accuracy. Similar to approach 01, all the results obtained for classification by GPA have shown very low predicting powers.

7.4.2.2 Evaluations for problem transformation methods, algorithm adaptation methods, and ensemble methods for technical skills

The table 7.8 show the different micro-f1 scores and accuracies obtained for the models implemented under different traditional multi-label classification approaches (problem transformation methods, algorithm adaptation methods, and ensemble methods), for technical skills.

Table 7.8: Comparison of results obtained for technical skills through problem transformation methods, algorithm, adaption methods, and ensemble methods

		Micro-f1 score	**Accuracy score**
Problem transformation			
Binary relevance	**Linear SVC**	0.0588	0.265
	GaussianNB	0.1869	0.0125
	Logistic Regression	0	0.25
	Decision Tree	0.1333	0.1375
Classifier chain	**Linear svc**	0.0545	0.225
	GaussianNB	0.189	0.0125
	Logistic regression	0	0.25
	Decision Tree	0.1451	0.125
Label powersets	**Linear svc**	0.0719	0.1625
	GaussianNB	0.1162	0.0625
	Logistic Regression	0	0.25

	Decision Tree	0.1209	0.0875
Algorithm adaptation			
k-Nearest Neighbor		**0.08**	**0.25**
Decision tree		**0.1031**	**0.075**
Ensemble			
Bagging	**Decision Tree**	**0.0741**	**0.175**
	Linear SVC	**0.0204**	**0.225**
	Logistic Regression	**0**	**0.25**
	GaussianNB	**0.1805**	**0.0375**
Gradient boosting		**0.1037**	**0.1875**
Random forest		**0.0217**	**0.25**

When analyzing the scores observed for classification on technical skills, it can be observed a significant increase in results for the majority of the models. Except for logistic regression models, all the other models have shown some level of scores under the micro-f1 metric. The highest micro-f1 scores were observed for gaussian models under different implementation methods. Also, decision tree models showed considerable level compared to other models. Still, the score levels remain at a very low level of between 5% - 20%.

7.4.2.3 Evaluations for problem transformation methods, algorithm adaptation methods, and ensemble methods for project keywords

The table 7.9 show the different micro-f1 scores and accuracies obtained for the models implemented under different traditional multi-label classification approaches (problem transformation methods, algorithm adaptation methods, and ensemble methods), for project keywords.

Table 7.9: Comparison of results obtained for project keywords through problem transformation methods, algorithm, adaption methods, and ensemble methods

		Micro-f1 score	Accuracy score
Problem transformation			
Binary relevance	**Linear SVC**	0.0385	0.2125
	GaussianNB	0.2062	0.0375
	Logistic regression	0	0.25
	Decision Tree	0.1302	0.0875
Classifier chain	**Linear SVC**	0.0367	0.1875
	GaussianNB	0.2054	0.0375
	Logistic regression	0	0.25
	Decision Tree	0.113	0.0625
Label powersets	**Linear SVC**	0.1039	0.1625
	GaussianNB	0.1182	0.0625
	Logistic regression	0	0.25
	Decision Tree	0.1111	0.075
Algorithm adaptation			
k-Nearest Neighbor		0.0577	0.2375
Decision tree		0.1272	0.0625
Ensemble			
Bagging	**Decision Tree**	0.0781	0.1875
	Linear SVC	0.0204	0.2125

	Logistic regression	0	0.25
	GaussianNB	0.1993	0.0375
Gradient boosting		0.1	0.2115
Random forest		0.0187	0.1875

The results obtained for model evaluation for project keywords show similar behavior as models for the technical skills but show slightly low micro-f1 scores and accuracy scores. The highest micro-f1 scores are gained for the gaussian naive bayes model whereas decision tree models depicted a certain level of micro-f1 scores as well. The observed maximum micro-f1 levels ranged between 18% - 21%.

7.4.2.4 Evaluation for the final combined model for approach 02

The models which gave the highest micro-f1 scores under different features GPA, Technical skill, and project keywords are summarized in table 7.10.

Table 7.10: Summary of models with highest micro f1-score

	Micro-f1 score	**Accuracy score**
GPA: Label Powerset - Decision Tree	0.1734	0.1
Technical skills: Classifier Chain - Gaussian Naive Bayes	0.189	0.0125
Project keywords: Classifier Chain - Gaussian Naive Bayes	0.2054	0.0375

The combination of the predictions made by the models depicted in Table 7.9 produced a micro-f1 score of 0.125 and accuracy score of 0.1625. The obtained results can be found in Figure 7.9.

```
from sklearn.metrics import accuracy_score, f1_score

micro_f1 = f1_score(actual, pred, average='micro')
accuracy = accuracy_score(actual, pred)

print('micro-f1 score:',micro_f1)
print('accuracy score', accuracy)
```

```
micro-f1 score: 0.125
accuracy score 0.1625
```

Figure 7.9: Micro f1-score and accuracy score for the combination of the predictions made by models

Even Though the micro-f1 score produced by combined prediction is lower than that of the three predictions made for single features, it has shown a better accuracy score. Hence, performance is improved overall but due to the values being still low, the model's prediction capabilities still remain at a poor level.

7.4.3 Evaluations for approach 03

The trained model for approach 03 was evaluated against an unseen dataset and the accuracy score and micro-f1 score obtained can be found in table 7.11.

Table 7.11: Accuracy score and micro-f1 score obtained for unseen data

Micro - f1 score	0.4593
Accuracy score	0.2395

When the limitations of the dataset are taken into account the results obtained through approach 03 indicate that the model's performance is relatively fair. The model is capable of predicting the positive labels correctly nearly half of the time. When looking at the accuracy score, it still indicates that model's capability in predicting entire label sets correctly remains at a low level.

7.4.4 Comparison of approach 01, approach 02, and approach 03

Table 7.12, compare the evaluation results of the three approaches implemented.

Table 7.12: Evaluation results of the three approaches implemented

	Micro-f1 score	Accuracy score
Approach 01 : Gradient Boosting	0.1379	0.3548
Approach 02	0.125	0.1265
Approach 03	0.4593	0.2395

When the three approaches are compared based on the micro-f1 scores, it is observed that approach 01 and approach 02 produce similar results whereas comparatively high results are produced through approach 03. Though the micro-f1 score of 0.4593 cannot be interpreted as a very good score, given the dataset limitations and other model's performances, it can be interpreted as a fairly reasonable score.

By examining the accuracy scores of three different approaches, it can be seen that approach 01 shows the highest score and approach 02 shows the lowest. Hence, the accuracy score of approach 03 lies in the middle of the accuracy scores of the other two approaches.

Based on the comprehensive analysis of both accuracy scores and micro-F1 scores, it can be concluded that approach 03 emerges as the optimal choice among the conducted experiments.

7.5 Module 4 - Analyze Resume Content and Suggest Content Improvements

7.5.1 Evaluation of Feature Extraction Model (NER Model)

In this section performance of the feature extraction model (NER model) is evaluated using the log of the NER model training. During the training process, the NER model's performance was monitored over several epochs. The recorded metrics at different stages of training provide valuable insights into the model's learning progress and its ability to accurately identify named entities. Here are the metrics obtained during the training:

- **Epochs**: The number of training epochs is indicated in the first column.
- **Loss**: The loss values for the "tok2vec" and "ner" components are provided.

- **ENTS_F**: The F1 score for named entities is reported.
- **ENTS_P**: The precision score for named entities is reported.
- **ENTS_R**: The recall score for named entities is reported.
- **SCORE**: The overall score is provided, which is a combined measure of the model's performance.

The figure 7.10 shows the performance metrics obtained during training.

```
============================ Training pipeline ========================
ℹ Pipeline: ['tok2vec', 'ner']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ------  ------------  --------  ------  ------  ------  ------
  0       0          0.00    217.54    0.00    0.00    0.00    0.00
  1     200       3932.36  28082.89   37.83   58.80   27.89    0.38
  3     400       6540.78  17608.96   58.73   64.37   53.99    0.59
  5     600       1969.98  12592.27   72.85   73.72   72.00    0.73
  7     800        916.82   9640.55   79.92   83.74   76.43    0.80
  9    1000       1034.64   7870.56   85.33   84.90   85.75    0.85
 11    1200        769.66   6201.32   87.50   86.60   88.42    0.88
 12    1400        708.90   5228.95   91.00   90.08   91.95    0.91
 14    1600        938.73   4749.39   91.82   93.20   90.47    0.92
 16    1800        687.48   3991.02   92.10   91.16   93.07    0.92
 18    2000        714.92   3650.43   94.35   94.38   94.31    0.94
 20    2200        822.21   3366.25   94.85   93.82   95.90    0.95
 22    2400        986.80   2958.19   95.19   95.94   94.45    0.95
 23    2600        579.78   2652.43   96.01   96.53   95.50    0.96
 25    2800        624.94   2442.13   95.76   95.17   96.35    0.96
 27    3000        597.95   2217.73   96.43   96.14   96.72    0.96
 29    3200        536.97   2115.36   96.54   96.54   96.54    0.97
 31    3400        620.25   2033.81   96.61   96.74   96.47    0.97
 33    3600        645.59   1918.07   97.82   97.74   97.90    0.98
 34    3800        812.15   1684.32   96.82   96.16   97.49    0.97
 36    4000        724.32   1755.95   96.83   96.70   96.96    0.97
 38    4200        946.76   1809.26   96.77   97.09   96.44    0.97
 40    4400        784.09   1646.11   96.97   97.09   96.85    0.97
 42    4600        667.64   1451.56   97.51   97.24   97.79    0.98
 44    4800        629.99   1456.53   97.57   97.18   97.96    0.98
 45    5000        637.74   1314.85   97.97   97.99   97.94    0.98
 47    5200        766.41   1243.41   98.07   97.57   98.58    0.98
 49    5400        572.47   1211.45   98.57   98.53   98.61    0.99
 51    5600       1091.00   1347.53   98.21   98.46   97.96    0.98
 53    5800        657.55   1071.88   97.78   97.79   97.76    0.98
 55    6000        697.26   1214.82   98.24   98.21   98.26    0.98
 56    6200        627.29   1042.60   98.25   98.55   97.96    0.98
 58    6400        713.97   1052.62   98.10   97.82   98.38    0.98
 60    6600        645.87   1000.15   98.96   98.86   99.06    0.99
 62    6800        830.72    988.74   98.44   98.11   98.76    0.98
 64    7000        745.79   1001.50   98.42   98.16   98.68    0.98
 66    7200        997.16   1048.31   98.56   98.94   98.18    0.99
 67    7400        995.58   1091.16   98.46   98.55   98.36    0.98
 69    7600       1095.21   1015.06   98.37   98.00   98.74    0.98
 71    7800       1140.02    969.17   98.74   98.55   98.93    0.99
 73    8000        837.59    914.40   98.42   98.24   98.60    0.98
 75    8200        999.06    949.15   98.83   98.77   98.89    0.99
✓ Saved pipeline to output directory
```

Figure 7.10: Log of the NER Training

The model starts with low performance and exhibits notable improvements in performance metrics as training progressed, as observed through increasing F1 scores, precision, and recall. At the end of the training process, the NER model demonstrated

promising results. The overall score achieved was 0.99, indicating a high level of accuracy in identifying named entities.

7.5.2 Evaluation of Approach 01 and Approach 02

A total of five models were developed for each section: one model from Approach 01 and four models from Approach 02. The evaluation of these models will help determine the best-performing model for each section. The main difference between the two approaches is that Approach 01 takes into account the entire content of the section, while Approach 02 focuses on specific features extracted through the NER model. Both approaches utilize regression models, with Approach 01 using the BERT regression model, and models in Approach 02 employing linear regression, decision tree regression, SVR, and random forest regression.

Since all five scoring models predict scores between 0 and 10, it is more appropriate to use Mean Absolute Error (MAE) rather than R-squared (R2) score for comparing models. Here's why:

- Interpretability: MAE provides a direct interpretation of the average prediction error in the original units of the target variable. In this case, the scores range from 0 to 10, and the MAE would represent the average absolute difference between the predicted and actual scores. This gives you a clear understanding of the average prediction error in the context of the scoring scale.
- Magnitude of Errors: MAE is sensitive to the magnitude of errors, as it calculates the average absolute difference between predictions and actual values. It will directly reflect the size of the errors in the score predictions. This is important when the absolute difference between predicted and actual scores is more critical than the direction of errors.
- R2 score interpretation: R2 score measures the proportion of variance in the target variable that is explained by the model. However, the interpretation of R2 score is less intuitive when the target variable has a limited range, like scores between 0 and 10. R2 score may not provide a meaningful representation of the model's performance in this specific context.

As explained, for a scoring model where the predicted scores range from 0 to 10, using MAE is more appropriate for comparing models. It provides a straightforward interpretation of the average prediction error in the score scale and is more suitable

when the absolute difference between predicted and actual scores is of primary importance. Hence MAE was utilized to evaluate the models developed through approach 01 and approach 02. The table 7.13 shows the MAEs for each model trained for each section.

Table 7.13: Mean Absolute Errors for each model trained for each section

Section	**MAE**				
	Approach 01	**Approach 02**			
	BERT Regression	**Linear Regression**	**Decision Tree**	**SVR**	**Random Forest**
Profile Section	0.6667	1.7209	1.5581	1.5581	1.4186
Education Section	0.7333	1.8837	0.74418	1.0	0.6279
Projects Section	0.5333	1.5349	0.9767	1.1860	1.1163
Technical Skills Section	1.2667	1.7714	2.2286	1.8286	1.7714

Based on the MAEs shown in the above table, for the profile, projects and technical skills sections, the BERT regression model developed in the approach 01 is more suitable where it considers entire content of the section and the context of the content when predicting the score for the section. For the education section, random forest regression model which was developed in the approach 02 where it considers the section specified information is more suitable.

Table 7.14 shows the scores predicted for the given sample content for each section form the models developed through both approach 01 and approach 02.

Table 7.14: scores predicted for the given sample content for each section

Section	Content	score					
		Actual	Approach 01	Approach 02			
			BERT Regression	Linear Regression	Decision Tree	SVR	Random Forest
Profile	Ability to handle highly stressed situations and unexpected errors in an efficient manner	02	02	02	02	02	02
	Dedicated third year undergraduate with strong interpersonal skills and extensive knowledge in the field of Information Technology, seeking for an internship in an organization that indulges personal growth while allowing me to utilize my knowledge and skills.	07	08	07	07	07	07
Education	BSc. (Hons) in Information Technology — University of Moratuwa Reading: second year - CGPA: 3.12 (Level 1) G.C.E. Advanced Level 2017 Z-Score: 1.5013 Central College, Anuradhapura, Sri Lanka	06	06	07	06	04	06
	B.SC. (HONS.) IN INFORMATION TECHNOLOGY UNIVERSITY OF MORATUWA Level 1 Semester 1 GPA 3.78 Level 1 Semester 2 GPA 3.88 Level 2	08	08	07	08	08	08

	Semester 1 GPA 3.56 Overall GPA 3.72 HOLY FAMILY CONVENT, COLOMBO 04 G.CE (A/L - 2013) - PHYSICAL SCIENCE STREAM Results - Physics A, Combined Mathematics B, Chemistry B GCE (O/L - 2009) Results - 6As 2Bs 1C PROFESSIONAL QUALIFICATIONS Successfully completed a Certificate course in Computer Science at National Institute of Business Management (NIBM), 2010						
Projects	class manager system for addressing a website and a mobile application to tuition classes for ease of their activities	02	01	03	02	02	02
	Remote mariner an under water camera that can be controlled by a remote controller and get the visual feedbacks. technologies: pic programming, wireless networking weather - now mentored by virtusa polaris(pvt)ltd. a crowdsourcing web application and android application which can be used to know real time weather updates and also it can be updated according to current weather satus and get the personal ranks of reliable updates. project link:	07	07	06	07	06	06

	matrix.projects.mrt.ac.ik:3000 technologies: angularjs, nodejs, html5, mongodb google maps api, android, mysql						
Technical Skills	Web Development HTML, CSS, JavaScript, Bootstrap, jQuery #NAME? Version Control Systems Git	05	06	04	05	03	04
	Programming Languages - C, Java Databases - MySQL, Oracle Web designing - HTML, JSP, JavaScript, php, Bootstrap, Servlets Multimedia - Sketchup IDE - Netbeans, Eclipse, Atmel Studio Version Control System - GitHub Other - Rational Rose, Star UML	08	09	06	06	08	08

7.6 Summary

The evaluation of different modules in the resume analysis system provides valuable insights into the performance and effectiveness of various models and approaches. In Module 1, the color contrast ratio identification was evaluated using two methods - (a) manual annotation and (b) a color clustering algorithm. The color clustering algorithm showed higher efficiency and effectiveness, making it the preferred choice for this task. Further for layout extraction, and first impression classification different evaluation metrics were used to measure the performance of these models, with the Random Forest Regressor showing the best results for certain tasks.

Module 2 involved the evaluation of multiple models for tasks such as resume section prediction and finding the cosine similarity between the predicted section content and actual section content. Module 3 focused on multi-label classification for company recommendation and interview question suggestions. Various traditional multi-label classification approaches were evaluated, and the results showed limited success in accurately predicting positive labels. In Module 4, the NER model showed promising

performance in feature extraction, particularly in identifying named entities in the resume content.

Overall, the evaluation revealed the strengths and limitations of each module and provided valuable information for further improvements. The system's performance was satisfactory for some tasks, but there is still room for enhancement, especially in multi-label classification and content improvement suggestions. Further research and model refinement can lead to a more comprehensive and accurate resume analysis system.

Chapter 8 - Discussion

8.1 Introduction

The discussions section aims to conduct an in-depth analysis, interpretation, and discussion of the implications of the research findings. This section aims to provide a thorough understanding of the research outcomes and contribute to the development of machine learning applications in the field of resume evaluation by critically evaluating the findings, making connections to pertinent theories and previous studies, and highlighting practical and theoretical insights.

8.2 Module 1 - Analyzing Resume Design

The analysis of resume design through graphical examination has provided valuable insights into the factors that influence the perception and evaluation of resumes by HR professionals. This discussion section presents and examines the key findings obtained from the graphical analysis of resumes.

The first finding of the study highlights the significance of contrast ratios in resume design. Resumes with good contrast ratios were observed to be more easily readable, enabling HR professionals to quickly grasp important facts. This finding emphasizes the importance of ensuring the usage of appropriate contrast ratios in resume design to enhance readability and facilitate efficient information processing. Furthermore, the use of light colors for the background was found to be more attractive compared to darker color backgrounds. This finding suggests that selecting lighter color schemes can contribute to the overall visual appeal of a resume, potentially capturing the attention of HR professionals and creating a positive impression. Regarding the layout of resumes, the study revealed that HR professionals tend to favor previously experienced and familiar layouts. This preference for familiar layouts indicates that there is a certain level of consistency and standardization in resume designs that is appreciated by HR professionals. It implies that while innovation and creativity are important, it is also crucial to adhere to established conventions to maintain a sense of organization and attractiveness in resume layouts. The analysis of color combinations in resume design revealed several noteworthy findings. The intensity level of the background color, the highest contrast ratio used, and the lowest contrast ratio used were identified as factors that significantly influence color combinations. These findings emphasize the

importance of carefully selecting and balancing color combinations to achieve visually pleasing and harmonious resume designs.

The impact of color combinations and layout on the first impression of a resume was also established. The study found that these two aspects play a crucial role in forming the initial perception of a resume. This suggests that HR professionals may quickly form judgments based on the color combinations and layout, highlighting the need for attention to detail in these areas to create a positive and memorable first impression. Finally, when evaluating the overall design of a resume, factors such as color combinations, layout, and the first impression level of the resume were identified as significant contributors. These findings indicate that a holistic approach to resume design, considering these factors collectively, is crucial for achieving a high overall design score.

Despite the valuable findings and insights obtained from the graphical analysis of resume design, there are certain limitations that should be acknowledged. These limitations are essential for a comprehensive understanding of the module's capabilities and potential areas of improvement. Firstly, one of the primary limitations of the module is the reliance on a limited dataset during the training phase. The accuracy and performance of the module's output are inherently constrained by the resumes available for training. Therefore, the generalizability of the module's findings may be limited, particularly when applied to a wider range of resumes that were not included in the training dataset. Secondly, the module's approach is based on supervised learning, which implies that it relies on pre-existing design examples for evaluation. Consequently, the module may struggle to assess completely novel or innovative resume designs that deviate significantly from the training dataset. This limitation indicates that the module's effectiveness in evaluating unconventional or unique resume designs may be compromised. Lastly, it should be noted that the module has primarily been trained using skill-based resumes. While the findings and evaluations from this dataset are informative for skill-based resumes, the module's performance and reliability may vary when applied to other resume types, such as chronological or experience-based resumes. The different structures and content organization in these resume formats may require specific adaptations or additional training to ensure accurate evaluations.

To address the aforementioned limitations, future research endeavors can focus on two key areas. Firstly, incorporating larger and more diverse datasets would significantly enhance the robustness and representativeness of the module's output. By including a wider range of resume types within the dataset, such as chronological or experience-based resumes, the module's versatility and applicability would be greatly improved. Furthermore, to overcome the limitation of the module's inability to explain novel or innovative resume designs and layouts, alternative approaches should be explored. One potential avenue is to incorporate unsupervised or semi-supervised learning techniques, which can enable the module to identify and assess patterns and structures in previously unseen designs. Additionally, exploring other design evaluation methods, beyond the current supervised learning approach, could provide a more comprehensive understanding and analysis of unconventional resume designs.

8.3 Module 2 – Resume Parser

The main objective of this module was to find a layout-aware content extraction technique to accurately extract the section wise text content from the resumes and grab the important entities from the resumes. Through the experiments and the implementations, an effective resume parsing system has been found as the main output of this module. The problems of the existing systems were the accuracy and inability to extract section wise data. Those systems only extract the important entities from resumes, not the whole content. The accuracy of the implemented parsing system showcased remarkable performance in accurately extracting and categorizing different sections of resumes. Through the implementation of advanced classification models and algorithms, the system demonstrated its ability to handle a diverse range of resume formats and layouts, ultimately yielding reliable results. The accuracy of the section predicting multiclass classification model which was implemented using SVM is around 0.8 which has a higher value. When considering the average cosine similarity value for each section it gives more than around 0.75 for every section. The system’s exceptional accuracy in extracting section-wise information can be attributed to the successful integration of machine learning techniques, such as the multiclass classification model and custom NER.

When considering the limitations of the implemented model, the variety of resume formats makes it difficult to effectively parse and extract information. Because different

people have different preferences, there are many different forms and layouts for resumes. The system may have issues when dealing with unusual or highly customized layouts, despite being built to properly handle typical resume formats. In addition to that, even when the algorithm successfully extracts section wise content, it could not have a complete knowledge about the context of that content.

As further work of this module, deep learning methods can be implemented to improve the effectiveness of our resume parsing system. Then the system will be able to learn more about the semantic and contextual information contained in resumes by investigating the use of deep learning models like recurrent neural networks (RNNs) or transformer-based models. Further, researching the application of layout-aware object detection and image processing techniques has the potential to significantly enhance the accuracy of paragraph division in the implemented resume parsing system. Then the system will be able to identify the visual layout components in resumes, including headings, subheadings, and text blocks, by utilizing advanced computer vision techniques and object detection models.

8.4 Module 3 – Company Recommendation System and Interview Questions Suggestion

The research focus of this module was to explore the existence of a relationship between resume text content and the possibility of getting shortlisted by a certain company. For our best understanding, there does not exist any research conducted to explore this problem. Through the experimentations done under approach one, it was found that the BERT model underperformed compared to the traditional classification algorithms. This implies that the context of the resume text content has minimal impact on the event of being shortlisted by a company. Majority of the models tried out under approach one demonstrated zero micro-f1 scores and amongst the models that showed a score, they too lied in a low range of 0% - 10%. Thereby, the approach 01 was found ineffective for addressing the problem at hand and the effect of entire text content of resume on being selected for a company is found to be not relevant. Since it was found entire text content is giving no insight into the task, through the next approach selected key features were given focus. Approach two tries to address the task by predicting the companies for which candidate may get shortlisted by individually making predictions for 03 key features identified (GPA, technical skills, and project keywords). Then the

combination of the 03 outputs is taken as the final prediction. In this approach it showed some level of micro-f1 scores compared to zero micro-f1 scores observed when entire content was considered. Also, the combined prediction produced a micro-f1 score and accuracy score, which despite being low values infers the possible existence of a relationship between shortlisting possibility and different significant features in resume content. Also, the consistent zero micro-f1 scores obtained for logistic regression models further support that there doesn't exist any linear relationship between the features analyzed under approach 01 and 02 against the shortlisting possibility. From approach 03 a frequency-based analysis was done such that it represents the entire content of the resume. The existence or absence of a certain section and in instances where the section data existed, the frequency of existence of key text features under each section was taken as the features for experimentation under the approach 03. This approach showed the best results among all the approaches tried out implying the possible existence of a relationship between a resume content with the possibility of the candidates being shortlisted. Still the obtained performances remain at a low level, emphasizing the requirement for further research on the matter.

This research was conducted with limited data resources owing to the difficulty in obtaining the data. The main limitation identified in the research is the practical difficulty in finding precise data. This is due to the fact that in the real-world scenario, it is nearly impossible to find the data on candidates such that they have applied for all the selected 10 companies. So, this can create a bias in the dataset, as the majority of the candidates may have applied to a few of the most popular companies. Had they applied for other companies as well, they might or might not have got shortlisted. Thereby the complexity of this problem is very high.

Within the scope of this study, only a few key features were taken into account when implementing approach 02. Based on the findings under approach 02, it can be anticipated that the performance could possibly be improved by experimenting with this approach by incorporating more key features. Further, in a multilabel classification problem there can exist label sets which may repeat many times, label sets which only occur a few numbers of times, and label sets which may never occur at all in the training dataset. For when addressing such kinds of scenarios, this research could be further extended to be experimented with few-shot and zero-shot learning approaches which

are not covered under the scope of this study. Also, since the scope of this study is limited to fresh graduates of Faculty of Information Technology, University of Moratuwa, the dataset remains limited. The same approaches can be applied to a broader context to further confirm and improve the findings of this study.

8.5 Module 4 - Analyze Resume Content and Suggest Content Improvements

The main objective of this module is to find appropriate approaches to give scores for sections of the resume based on the quality of the content and to give content improvement suggestions. Most of the existing approaches focus on analyzing resume content for recruiters and did not provide content improvement suggestions for the candidates. Even though there are some similar solutions, they did not provide feedback accurately based on the section-wise content.

When analyzing the content of the resume, it is important to identify what are the important sections and details that should be included in the resume. Through the initial research done, it is identified that, when studying resumes by a recruiter, 26.1% of recruiters spend only 30-60 seconds and around 30% of the recruiters spend 1-2 minutes while 27.5% spend 2-3 minutes. Hence, it is important to mention content in a way to gain attention in such a small time. Moreover, it is identified that personal information, job objective/profile, education, experience, skills, hobbies, interests, and extracurricular activities and referees are some major sections that should be included in the resume. As customization this for the proposed solution, since this is for the undergraduates of the Faculty of Information Technology, project section included instead of the experience section and also technical skills and non-technical skills sections included instead of skills section. As per the initial research findings it is also identified some important contents that should be included in above mentioned sections. Applicant's name, phone number and address are some mandatory information that should be included as personal information. As per the education qualifications it is important to include degree or the designation and the major, minor and if applicable, the anticipated graduation date and GPA. Also, as the result of the survey done with the recruiters, it is identified that including the technologies utilized in each project is also very important. Moreover, it is identified that details such as height, weight, race, religion, birth date, marital state, number of dependents,

physical/health status, and social security number should not be included in the resumes.

When it comes to the scoring model, when considering the overall contents of a sections, for the feature extraction phase vectorization methods like CounterVectorizer in not much suitable. Because it only considers the frequencies of the words presented. It is also important to consider the context of the information to provide more accurate results. Through the research it is identified that BERT (Bidirectional Encoder Representations from Transformers) is more suitable for scoring model since it also takes context of the content into consideration. Moreover, as research findings it is also identified that for sections like profile and projects of the resume to gain more accurate score it is needed to consider the whole content of the section where sections like education provide more accurate results when considering only the section specified features.

This research study encountered limitations due to the scarcity of available data resources, which posed challenges in obtaining a sufficiently large and precise dataset. The primary limitation identified in this research pertains to the practical difficulty in acquiring highly accurate and specific data. The scores assigned to different sections of the resumes tended to exhibit a bias towards scores higher than 5. To mitigate this bias and improve the dataset's representativeness, additional data was incorporated with the assistance of recruiters. It is important to acknowledge that the models employed in this research were trained using limited data resources. Consequently, the outcomes and performance of these models are contingent upon the dataset used for training. Furthermore, the results derived from this particular module heavily rely on the outputs generated by module 2, because the main input for this module is the outputs generated from it. Therefore, it is imperative to ensure the accuracy of the outputs from module 2 to obtain precise and reliable results from this module.

As for further research studies based on this research, for the scoring model, another approach where it analyzes the context of the section specified features could be tried. Also, an approach with the combination of both overall content and section specified features could be studied to find whether those approaches would have any impact on the accuracy of the results. Moreover, the performance of this module could be enhanced by training the models with larger datasets.

References

[1] P. G. Roos, “Development and evaluation of a competence-based curriculum vitae-writing programme for new graduates,” North-West University, 2018.

[2] “7 Simple But Effective Ways to Make Your CV Stand Out | Top Universities.” https://www.topuniversities.com/blog/7-simple-effective-ways-make-your-cv-stand-out (accessed Jul. 22, 2022).

[3] J. Rout, S. Bagade, P. Yede, and N. Patil, “Personality Evaluation and CV Analysis using Machine Learning Algorithm,” *Int. J. Comput. Sci. Eng.*, vol. 7, pp. 1852–1857, 2019, doi: 10.26438/ijcse/v7i5.18521857.

[4] E. Furtmueller, C. Wilderom, and R. M. Mueller, “Online resumes: Optimizing design to service recruiters,” 2010.

[5] R. M. Schramm and R. Neil Dortch, “An analysis of effective resume content, format, and appearance based on college recruiter perceptions,” *Bull. Assoc. Bus. Commun.*, vol. 54, no. 3, pp. 18–23, 1991.

[6] J. K. Arnulf, L. Tegner, and Ø. Larssen, “Impression making by résumé layout: Its impact on the probability of being shortlisted,” *Eur. J. Work Organ. Psychol.*, vol. 19, no. 2, pp. 221–230, 2010.

[7] A. M. Mithun, Z. A. Bakar, and W. M. S. Yafooz, “The impact of web contents color contrast on human psychology in the lens of HCI,” *Int. J. Inf. Technol. Comput. Sci.*, vol. 11, no. 10, pp. 27–33, 2019.

[8] R. H. Hall and P. Hanna, “The impact of web page text-background colour combinations on readability, retention, aesthetics and behavioural intention,” *Behav. \& Inf. Technol.*, vol. 23, no. 3, pp. 183–195, 2004.

[9] G. N. Burns, N. D. Christiansen, M. B. Morris, D. Periard, J. A. Coaster, and others, “Effects of applicant personality on resume evaluations,” *J. Bus. Psychol.*, vol. 29, no. 4, pp. 573–591, 2014.

[10] V. Bhatia, P. Rawat, A. Kumar, and R. R. Shah, “End-to-end resume parsing and finding candidates for a job description using bert,” *arXiv Prepr.*

arXiv1910.03089, 2019.

[11] S. Zu and X. Wang, “Resume information extraction with a novel text block segmentation algorithm,” *Int J Nat Lang Comput*, vol. 8, pp. 29–48, 2019.

[12] J. Chen, L. Gao, and Z. Tang, “Information extraction from resume documents in pdf format,” *Electron. Imaging*, vol. 2016, no. 17, pp. 1–8, 2016.

[13] C. Daryani, G. S. Chhabra, H. Patel, I. K. Chhabra, and R. Patel, “An automated resume screening system using natural language processing and similarity,” *ETHICS Inf. Technol. [Internet]. VOLKSON Press*, pp. 99–103, 2020.

[14] S. Sanyal, S. Hazra, S. Adhikary, and N. Ghosh, “Resume parser with natural language processing,” *Int. J. Eng. Sci.*, vol. 4484, 2017.

[15] D. Chandola, A. Garg, A. Maurya, and A. Kushwaha, “Online resume parsing system using text analytics,” *J. Multi Discip. Eng. Technol.*, vol. 9, no. 1, pp. 1–5, 2015.

[16] A. Anand and M. S. Dubey, “CV Analysis Using Machine Learning,” *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 5, pp. 1316–1322, May 2022, doi: 10.22214/IJRASET.2022.42295.

[17] L. Mathew, N. C. George, N. Linet, and N. K. Thomas, “‘ATS BREAKER’-A System for Comparing Candidate Resume and Company Requirements.”

[18] A. N. Tarekegn, M. Giacobini, and K. Michalak, “A review of methods for imbalanced multi-label classification,” *Pattern Recognit.*, vol. 118, p. 107965, 2021.

[19] N. H. Anderson and A. A. Barrios, “Primacy effects in personality impression formation.,” *J. Abnorm. Soc. Psychol.*, vol. 63, no. 2, p. 346, 1961.

[20] S. D. Risavy and others, “The resume research literature: Where have we been and where should we go next,” *J. Educ. Dev. Psychol.*, vol. 7, no. 1, pp. 169–187, 2017.

[21] H. Hewage, K. U. Hettiarachchi, K. Jayarathna, K. P. C. Hasintha, A. N.

Senarathne, and J. Wijekoon, “Smart human resource management system to maximize productivity,” in *2020 International Computer Symposium (ICS)*, 2020, pp. 479–484.

[22] T. Zimmermann, L. Kotschenreuther, and K. Schmidt, “Data-driven HR-R$\backslash$’esum$\backslash$’e Analysis Based on Natural Language Processing and Machine Learning,” *arXiv Prepr. arXiv1606.05611*.

[23] A. Shestakova and A. Corradini, “Exploring the Use of Machine Learning for Resume Recommendations,” in *International Conference on Speech and Computer*, 2022, pp. 626–640.

[24] Peter Szemraj, “flan-t5-large-grammar-synthesis (Revision d0b5ae2).” Hugging Face, 2022, doi: 10.57967/hf/0138.

[25] P. Damodaran, “A framework for detecting, highlighting and correcting grammatical errors on natural language text.” https://github.com/PrithivirajDamodaran/Gramformer/ (accessed Jun. 13, 2023).

[26] E. A. Cherman, M. C. Monard, and J. Metz, “Multi-label problem transformation methods: a case study,” *CLEI Electron. J.*, vol. 14, no. 1, p. 4, 2011.

[27] J. Read, B. Pfahringer, G. Holmes, and E. Frank, “Classifier chains for multi-label classification,” *Mach. Learn.*, vol. 85, pp. 333–359, 2011.

[28] P. Pant, A. Sai Sabitha, T. Choudhury, and P. Dhingra, “Multi-label classification trending challenges and approaches,” *Emerg. Trends Expert Appl. Secur. Proc. ICETEAS 2018*, pp. 433–444, 2019.

[29] S. González, S. Garc\’\ia, J. Del Ser, L. Rokach, and F. Herrera, “A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities,” *Inf. Fusion*, vol. 64, pp. 205–237, 2020.

[30] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[31] R. Smith, “An overview of the Tesseract OCR engine,” in *Ninth international*

conference on document analysis and recognition (ICDAR 2007), 2007, vol. 2, pp. 629–633.

[32] I. Pavan Kumar, V. P. Hara Gopal, S. Ramasubbareddy, S. Nalluri, and K. Govinda, “Dominant color palette extraction by K-means clustering algorithm and reconstruction of image,” in *data engineering and communication technology*, Springer, 2020, pp. 921–929.

[33] L. Moreno, R. Alarcon, and P. Mart\'\inez, “Accessibility and readability compliance in Spanish public hospital websites,” in *Proceedings of the 10th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion*, 2022, pp. 185–194.

[34] J. Sweller, “The role of evolutionary psychology in our understanding of human cognition: Consequences for cognitive load theory and instructional procedures,” *Educ. Psychol. Rev.*, vol. 34, no. 4, pp. 2229–2241, 2022.

[35] J. Canny, “A computational approach to edge detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 6, pp. 679–698, 1986.

[36] R. O. Duda and P. E. Hart, “Use of the Hough transformation to detect lines and curves in pictures,” *Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.

Appendixes

Appendix A

Individuals Contribution to the Project

Name of student: Hindakaraldeniya T.M.
Index Number: 184055D

The module I am responsible for is the company recommendation and interview question suggestion system.

What the company recommendation system does is that it considers the textual content of a resume and outputs a list of companies for which there is a high possibility of getting shortlisted in the first round of resume screening. By doing background research it was realized that this problem should be addressed as a multi-label classification problem. For preparing the dataset, the resumes of previously selected candidates and the companies from which they got first calls were collected.

After preparing and studying the dataset, I researched and tested different machine learning and deep learning approaches to apply for the problem at hand. Because the resume content is a large text content, to experiment the importance of context of resume content for the classification task, transfer learning mechanism was chosen as the approach to experiment. Also, entire text content was analyzed through conventional classification algorithms through various methodologies for implementing multilabel classification. On studying different transfer learning models, Bidirectional Encoder Representations from Transformers (BERT) was chosen as research proved that it performed well in classification tasks. Then the appropriate loss functions, optimizers, and evaluation metrics were studied to choose the most appropriate ones for the selected model. After building and training the model it was tested against test data. Similarly, models were implemented and evaluated under different traditional classification algorithms and results were compared with the BERT model.

Since, findings through approach 01 indicated poor prediction capability in considering entire text content, an approach 02 where key features were given importance was

implemented. For this approach, I prepared datasets using NER in module 04 and appropriately cleaned and preprocessed the extracted data. Then different models were implemented to carry out classification and evaluated by comparing results. This approach showed a slight improvement against approach 01 but still didn't provide clear proof of the existence of the researched relationship. Thereby, I tried out an approach three where the GPA value along with the frequency of occurrence of keywords under different sections of the profile which also check the existence of a specific section in a resume. Through this approach a comparatively better result was obtained when evaluated against a test data set. Then I explored how to implement the practice interview suggestion system. First, I tried to implement this by manually creating a question bank but the limitation of the number of questions that can be provided through a question bank and the complexity in creating a question bank were identified as shortcomings of this approach. Then I found that it was better to be implemented using a reliably tested existing generative model to build this sub module through which user is provided the facility of requesting their preferred number of questions as well as preferred complexity level of questions that they wish to practice upon.

Finally, as a team, we all worked together integrating the modules to build the final system and the website.

Name of student: Perera N.N.
Index Number: 184126X

The first module of the project, analyzing resume design using image processing techniques was my assigned module in this project.

As the first task of the project, it was needed to check the feasibility of the module and limiting the scope. After having a look around in the related works, it was clear that, there was not any particular research study which was focused on building machine learning based resume graphical design analyzing system. The existing systems were mainly focused as development projects, not the projects which were driven by a research study. When considering the scope of the module, according to earlier studies, it stated that layout and color combination play huge roles in the resume graphical design which will leads for the first impression of the resume. Therefore, after studying some background details, it was decided to limit the scope to color combinations, resume layout, first impression classification and overall design scoring.

Since there was not any research for each of the mentioned tasks, it made a hard time finding related works among past studies. As the first subcomponent of the module, for calculating text-color contrast ratio WCGA web page made a good reference for the task.

For measuring color combinations, first impression level and overall score for a resume design it was needed to prepare a dataset which is a score sheet. And to share with others to score, it was needed to find many resumes from the undergraduates/fresh graduates in IT field. To fulfil that requirement, our team was able to collect all internship resumes from the industrial training platform administrator. After getting the required number of resumes we had several meetings with HR personnel from various IT companies to create the guideline for scoring sheet. Then the scoring sheets were distributed among HR personnel and collected after marking scores for each allocated resume.

For the implementation of the modules various techniques were tried out, which some of them are newly built algorithms while others were existing algorithms. Further, for implementing neural network-based regression algorithms, previous studies were taken

as references. After thoroughly going through all those studies, finally, I was able to come up with solutions for addressing each of the problems identified in each subcomponent. Accordingly, implementation of the proposed approaches was carried out using the solutions that were designed.

Finally, the all modules were compiled to single website using Django framework and made available for the users as a collective effort.

Name of student: Warusawithana S.P.
Index Number: 184186E

As per the initial project proposal, the main objective of module 2 is to implement a method to extract all the text content from the resumes while considering the section wise content. The initial idea was to extract the text content directly from the resumes. But there would be a new approach since module 4 needs section wise content to analyze the data.

There were several ways to extract text data from pdf documents such as using OCR techniques and using the text extraction libraries. With the help of the research papers, I found an approach to extract whole content from the resumes with higher. As the first step, the system used a library to extract whole text content from the resume without considering the layouts or sections. After extracting those then the system cleans the text content by removing unnecessary new line characters and output it to both module 3 and 4.

In addition to that, I had to consider the section wise text extraction. As described in many research papers, there were several ways to extract section wise entities. But for the proposed system we needed the whole content which includes inside those sections. Therefore, I had to do research about layout aware text extraction.

Moreover, we needed a dataset which includes the text content and the section label. First, we collected past undergraduate resumes of our previous batches from the industrial training platform under the consent of protecting privacy. To create that dataset, we used a software called Label Studio to annotate the resumes. Then I exported a file which includes the annotated data with the x, y coordinates and wrote a code block for reading the exported file and I used an OCR technique to extract the text content from each section. Then the dataset was created after extracting text from each section.

Further, after referring research papers a multiclass classification model which can predict the resume section for a given content was implemented using machine learning algorithms. By referring to research papers I used CountVectorizer and TF-IDF for the vectorization part and then I used both Naive Bayes algorithm and Support Vector

Machine with SGDClassifer to train the model and got the best one which has higher accuracy.

Finally, when the user uploads their resume to the system, an effective algorithm will be used to divide sections and paragraphs of the resume by considering the spaces between word boxes. After identifying the different sections from the resume, then the multiclass classification model will be applied to predict the relevant section for each divided section. Then all the section wise extracted data will be output using JSON files

Name of student: Weerasinghe R.L.
Index Number: 184188L

In this research project my primary focus is to do the research and implementation about the fourth module which is analyzing resume content and content improvement suggestions. This module contains scoring resume content according to the continent quality of each section, suggesting important missing fields, and suggest grammar and spelling mistakes.

As the starting point, I went through the existing studies related to my part. I was able to find some studies regarding the contents of the resume. By referring those studies, I was able to get clear idea about what should be there in a resume and what shouldn't. Moreover, gathered details about the order of the content and importance of each content. Apart from that we had several meetings with the Sri Lankan IT companies, and we got inputs regarding the contents of a resume from experts. Those details include what details are important, order of the sections, length of a resume, etc. Then I also followed some existing approaches for scoring the resume contents. Most of the existing approaches focus on analyzing the resume content from a recruiter's perspective.

One of our main challenges was to prepare the dataset. To create a collection of resumes we obtain resumes of undergraduates past five batches of the IT faculty from the platform administrator of the industrial training platform under the consent of protecting the privacy details. Through the research done, criteria for scoring resumes were finalized and the resumes required for the dataset preparation were scored by taking assistance from industry experts. Moreover, we annotated resumes contents section wise for creating a dataset of resumes for the training purpose of the models.

When it comes to implementation of module four, I was focus on implementing the content scoring model. As a beginning I tried to implement that model with using CountVectorizer and linear regression. Even though this method gives a score for the content, it does not consider the context of the content. Hence, the accuracy of this approach was very low and not suitable for the required task. Therefore, I started to study about transformers and decided to use BERT for the implementation. At the moment I implemented resume content scoring for profile section using BERT

transformer and it resulted considerable accuracy. Then as the second approach I focused on the section specific features. To extract the section specific features first I developed a spaCy NER model. For that, I Had to annotate the resumes of the dataset using the NER Annotator for spaCy. After annotating, I have successfully trained and developed the NER model for feature extraction. Then using the features extracted for each section using the NER model, Linea regression, SVR, Decision Tree Regression, and Random Forest regression models were trained. Finally, after analyzing all five models together, the best model for each section was chosen. The content improvement suggestion feature was implemented using rule-based techniques by considering the outputs of the NER model.

Moreover, as the part of content improvement suggestions, checking grammar and spelling is important. For that, several models are there that could be utilized by us. After studying several famous existing models I chose Gramformer model to implement the grammatical error checking feature. The reason to choose this model is because of the features provided by the model and the easiness of the model.

After successfully implementing the models related to my module, we collaboratively developed the platform for users and integrated the modules that we developed.