JOB RECOMMENDATION AND PROFILE MATCHING USING RESUME PARSING FOR HR ANALYTICS

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering degree in Computer Science and Engineering

By

UDAY VENKAT SARAN P (Reg no: 39110774)
SAI PRASAD P (Reg no: 39110781)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | 12B Status by UGC
Approved by AICTEC JEPPIAAR, RAJIV GANDHI SALAI,
CHENNAI – 600119

APRIL - 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Uday Venkat Saran P (Reg.No - 39110774) and Sai Prasad P (Reg.No - 39110781)** who carried out the Project Phase-2 entitled **"JOB RECOMMENDTION AND PROFILE MATCHING USING RESUME PARSING FOR HR ANALYTICS"** under my supervision from January 2023 to April 2023.

[illegible]

Internal Guide

Dr. R. AROUL CANESSANE, M.E., Ph.D.

[illegible]

Head of the Department
Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on______________________________

[illegible] [illegible]

Internal Examiner **External Examiner**

DECLARATION

I, **Uday venkat saran P(39110774) and Sai Prasad P(39110781),** here by declare that the Project Phase-2 Report entitled " **JOB RECOMMENDATION AND PROFILE MATCHING UAING RESUME PARSING FOR HR ANALYTICS "** done by me under the guidance of **Dr. R. Aroul Canessane, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

[illegible]

DATE:17/04/2023

PLACE: Chennai **SIGNATURE OF THECANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D**, **Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. R. Aroul Canessane M.E., Ph. D,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks totally Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTARCT

Agencies and various high-level firms must deal with a large number of new jobs seeking people with various resumes. However, managing large amounts of text data and selecting the best-fit candidate is more difficult and time-consuming. This paper provides an overview of an ongoing Information Extraction System project that helps recruiters in identifying the best candidate by extracting relevant information from the resume. This project presents a system that uses Natural Language Processing (NLP) techniques to extract minute data from a resume, such as education, experience, skills, and experience. The recruiting process is made easier and more efficient by parsing the resume. The proposed system is made up of three modules: an administration management system, File upload and parser system, and an information extraction system. The administrator will upload the applicant's resume into the system, and the relevant information will be extracted in a structured format. Using the parsed information from the Resume, HR can select the best candidate for the job based on the company's needs.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

Organizations develop and sustain by innovating new ideas to compete in the digital era. Innovation leads to decrease in manpower and increase in machine handling. Daily, corporate firms and recruiting agencies have to process a large number of resumes. Working with a large volume of text data is usually time consuming and stressful. Data gathered from different resumes can be in a various form, including .pdf, .docx, single column resumes, double-column resumes, free formats, and so on. And these formats might not be suitable for the particular application. So, questions may arise in our mind that, what is resume parsing? The process of converting the unstructured form (.pdf/ .docx / .jpeg etc.) of resume data into a structured format is known as resume parsing.

Subsequently, converting a resume into prepared text or structured information makes studying, analyzing, and comprehending easier. As a result, many organizations and institutions depend on Information Extraction, where unstructured data and vital information are extracted and converted to make information more readable and organized data forms. The completion of this task takes a long time for humans. So, it is necessary to develop an automated intelligent system that can extract all relevant information to determine whether an applicant is suitable for a particular job profile.

The foundation of this project is a resume automation system. Concerning the project, there will be an admin panel at first, into which the administrator must initially log in. Following that, there will be a section for uploading gathered CVs, which the admin will manage. After that, Regex, NLTK, and Spacy's phrase matcher will extract necessary information such as Name, Address, Email, phone number, Nationality, Skills [Hard Skills, Soft Skills], Education, Experience, Experience Year, Languages etc. in. json format. Finally, the extracted information or dump JSON file is saved in the database by admin if necessary. Below is the wireframe of Unstructured Data (Single-Column, Double-Column, Free Format). Structured Data

(Name, Address, Email, phone number, Skills [Hard Skills, Soft Skills], Education, Experience, Languages).

Name	
Address	
Email	
Phone number	
Skills	
Education	
Languages	

The structured data comes in. json format which makes the HR department easy to read the resume.

1.1 PROBLEM STATEMENT

People with diverse personalities come from a variety of fields and backgrounds. In the same way, their resume writing style varies. They've worked on a variety of projects, and each of them has unique writing style. As a result, each resume is unique.

Some people work in the human resources department. They will have to go through hundreds of resumes on the internet. Executives summarize the resume, enter specific information into their database, and then call the applicant for job counselling after obtaining the resume. An executive spent around 10-15 minutes on each resume, summarizing it and entering the information into the database. This project will help in the automation of this procedure.

1.2 IMPLEMENTATION OF MODULE

By collecting the relevant information from the resume, a natural language processing technique is employed to create a Hire ability system. Different NLP libraries, such as NLTK and Spacy, are utilized for extraction. Natural language networks are usually taught as unsupervised techniques, which implies that no previous tagging or labeling is done before the model is trained.

1.2.1 REASON BEHIND IMPLEMENTATION OF NLP

The key reason I chose NLP for Hire ability is that it can handle massive volumes of data in seconds or minutes that would take days or weeks to analyze manually. NLP technologies can instantly scale up or down to match demand, allowing us to have as much or as little processing capacity as per requirement. Aside from that, humans are prone to errors or may have personal biases that might distort the findings while conducting repeated jobs like examining resumes one by one and other textual data. In this case, NLP-powered solutions can be taught to understand company's need and requirements in only a few steps. So, once they're up and going, they perform better in terms of accuracy.

1.2.2 WORK FLOW OF NATURAL LANGUAGE PROCESSING

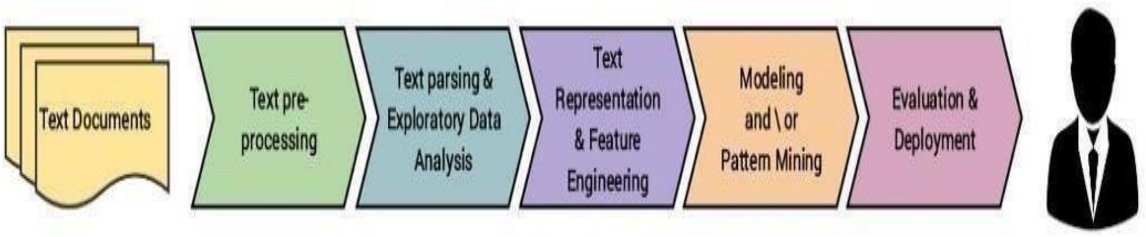

Text Documents
Text pre-processing
Text parsing & Exploratory Data Analysis
Text Representation & Feature Engineering
Modeling and \ or Pattern Mining
Evaluation & Deployment

Figure 1: Generic Workflow of NLP

NLP is an interdisciplinary subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. The goal is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.

The procedures given above represent the standard workflow for an NLP task. The initial stage is generally text wrangling and pre-processing on the collection of documents. Then there's parsing and some basic exploratory data analysis. The next stage is to represent text using word embeddings and then do feature engineering.

Challenges in natural language processing frequently involve speech recognition, natural-language understanding, and natural-language generation.

In the early days, many language-processing systems were designed by symbolic methods, i.e., the hand-coding of a set of rules, coupled with a dictionary look up such as by writing grammars or devising heuristic rules for stemming.

More recent systems based on machine-learning algorithms have many advantages over hand-produced rules:

- The learning procedures used during machine learning automatically focus on the most common cases, whereas when writing rules by hand it is often not at all obvious where the effort should be directed.

- Automatic learning procedures can make use of statistical inference algorithms to produce models that are robust to unfamiliar input (e.g., containing words or structures that have not been seen before) and to erroneous input (e.g., with misspelled words or words accidentally omitted). Generally, handling such input gracefully with handwritten rules, or, more generally, creating systems of handwritten rules that make soft decisions, is extremely difficult, error-prone and time-consuming.

- Systems based on automatically learning the rules can be made more accurate simply by supplying more input data. However, systems based on handwritten rules can only be made more accurate by increasing the complexity of the rules, which is a much more difficult task. In particular, there is a limit to the complexity of systems based on handwritten rules, beyond which the systems become more and more unmanageable. However, creating more data to input to machine-learning systems simply requires a corresponding increase in the number of man-hours worked, generally without significant increases in the complexity of the annotation process.

Despite the popularity of machine learning in NLP research, symbolic methods are still (2020) commonly used:

- when the amount of training data is insufficient to successfully apply machine learning methods, e.g., for the machine translation of low-resource languages such as provided by the Apterium system,
- for pre-processing in NLP pipelines, e.g., tokenization, or
- for postprocessing and transforming the output of NLP pipelines, e.g., for knowledge extraction from syntactic parses.

1.3 AIMS

- To take the help of the cutting-edge and latest NLP technology to enhance their business processes.
- To extract the required information about candidates without having to go through each resume manually.
- To replace slow and expensive human processing of resumes with extremely fast and cost-effective software.

1.4 OBJECTIVES

- For information extraction, NLP model will be configured and reconfigured.
- A system for uploading resumes will be developed.
- The unstructured data will be transformed into structured form.

1.5 SCOPE AND LIMITATIONS OF THE PROJECT

A purpose system can help in resolving the challenge of obtaining useful information from a resume in a structured format. By resolving this issue, recruiters will be able to save hours each day by eliminating manual resume screening. Bias in hiring is still prevalent, thus this method may also address the bias hiring process and strengthen a non-bias policy.

Purpose system is not able to solve the complex network issue such as:

- Excessive web traffic can significantly slow down or restrict access to a website entirely. This occurs when the server receives more file requests than it can handled.
- Latency issue.

1.6 ARTEFACTS

A subsystem is a fundamental component of a larger system. Dividing the system into subsystems will aid in the development of a more advanced version. The following are the three key artefacts that will be developed for this project in order to create a functional model.

1. Admin management system

This sub-system is one of the most secretive aspects of the project. As it is a carefully guarded part of the project, only the administrator can access using the admin email ID and password. After the admin logged in, there'll be given the option to upload the resume. Following the submission of the resume, there will have two sub-models: resume parsing and information extraction. The following is a brief list of how this artefact will be created and the tools required to do it.

IDE: VsCode/Sublime text.
Frontend: Tkinter

Backend: Python
Build Tools: Pip

Browser: Google Chrome or Mozilla
Control: git

2. File upload and Parsing system

This system is at the center of the overall project. In this sub system the admin will upload the resume and go for further processing. The parser subsystem will use different libraries to transform the submitted unstructured resume to a structured format. This will make it much easier to examine, analyze, and grasp the data. The following is a brief list of how this artefact will be created and the tools required to do it.

Data Requirement: dataset collected from Kaggle
Data analysis and evaluation: Jupyter Notebook/VsCode

IDE: VsCode/Sublime text
Version Control: Git

Browser: Google Chrome or Mozilla
Build Tools: Pip

3. Information extraction system

After parsing the data, we employ the NLTK, Spacy phrase matchers, regex to extract the essential information. The required extracted data will be saved in JSON format. Finally, the extracted data or dump JSON file is stored in a MySQL database for further usage if it is needed. The following is a brief list of how this artefact will be created and the tools required to do it.

Programming Language: Python 3 Technologies
Libraries: NLTK, Spacy

Data analysis and evaluation: Jupyter Notebook/VsCode
IDE: VsCode

Version Control: Git **Build Tools:** Pip

CHAPTER-2

LITERATURE SURVEY

Agencies and different high-level companies have to deal with an extreme number of new jobs seeking employees with different resumes. However, looking after those large numbers of text data and filtering out the needed candidates is a burden on the brain and more time consuming. Therefore, the essence of this literature review is on studying resumes in different formats such as single-column resumes, double-column resumes with extension.pdf,.docx, and how the suggested Information Extraction System converts that unstructured information into structured layout through Parsing.

This technique stated parsing of the resumes with least limit and the parser works the utilization of two or three rules which train the call and address. Scout bundles use the CV parser system for the determination of resumes. As resumes are in amazing arrangements and it has different sorts of real factors like set up and unstructured estimations, meta experiences, etc. The proposed CV parser approach gives the component extraction method from the moved CV's.

It follows an approach of 4 stages, the first stage was to get the data (resume) and convert them into structured format and then perform the analysis using deep learning techniques. Second step includes the psychometric test where the text mining is used to generate scores for each candidate. In the third step they perform web scraping on various social media sites to get the additional information about the candidates and recommend suitable jobs to them. In the fourth step, the system will recommend the skills and requirements in which the students are lacking and also help them to get recruited in the desired company.

Here the problem definition was based on designing an automated resume parser

system, which will parse the uploaded resume according to the job profile. And transform the unstructured resumes into structured format. It will also maintain a ranking system on the resumes. Ranking will depend on the basis of information extracted i.e.,
according to technical skills, education etc. Here the CV parser is used.CV parsing is such a technique for collecting CV’s. CV parser supports multiple languages, Semantic mapping for skills, job boards, recruiter, ease of customization. Parsing with hire ability provides us accurate results. Its integration makes users API key for integration efforts. The parser operates using some rules which instructs the name and address. Recruiter companies use CV parser technique for selection of resumes. As resumes are in different formats and it has different types of data like structured and unstructured data, metadata etc. The proposed CV parser technique provides the entity extraction method from the uploaded CV’s.

In this work, a qualitative assessment of resumes on the basis of different quality parameters using a simple text analytic based approach for a resume collection was described. The resume collection was processed for two qualitative coverage, comprehensibility and the aspects; and extracted ratings are modified into a quality rating which is comprehensive. All the parameters were collectively uniformed into a combined 1 to 5 rating scale for associating a quality metric for resumes. The qualitative evaluation results obtained through the algorithmic approach were congruent to and were hence validated through the wisdom of crowds.

Accordingly, this review also helps to understand and apply several in-use and well recognized algorithms currently being used in industries to reduce human labor. Depending upon the Company's preference to hire employees, the Extraction System will manage the gathered information with more readability and organized data forms. Furthermore, the analysis of various machine learning algorithms and natural language processing techniques would be equally carried out along with their proper implementation and evaluation. The reviews from multiple research publications and journals are included below.

2.1 End-to-End Resume Parsing and Finding Candidates for a Job Description.

For evaluating the suitable candidates for various job openings in consideration to their compatibility, the use of deep learning-based system has provided the end-to-end solution for people seeking employment They may be: first, by developing a resume parser that extracts all necessary information from candidate resumes, and second, by conducting ranking using. In this research, they explored the possibility of building a standard parser for all types of resumes and determined that it was impossible to do so without losing information in all situations, resulting in the unfair rejection of specific candidates' resumes.

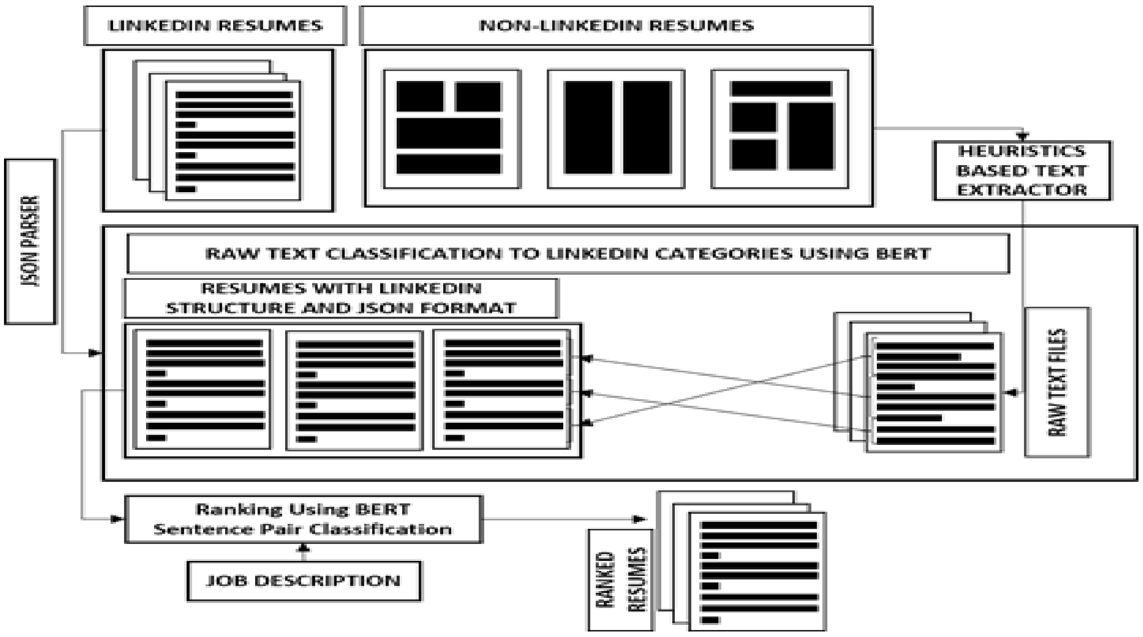

LINKEDIN RESUMES
NON-LINKEDIN RESUMES
JSON PARSER
HEURISTICS BASED TEXT EXTRACTOR
RAW TEXT CLASSIFICATION TO LINKEDIN CATEGORIES USING BERT
RESUMES WITH LINKEDIN STRUCTURE AND JSON FORMAT
RAW TEXT FILES
Ranking Using BERT Sentence Pair Classification
JOB DESCRIPTION
RANKED RESUMES

Figure 2: A system diagram showing data transmission and task completion.

Instead, they used LinkedIn-style resumes to scan without losing any information. In addition, the study also creates a firm foundation and a feasibility study that can lead to advancements in deep learning and language representation being used in the hiring process. The system diagram below represents the data flow and the completed task.

2.2 Information Extraction from Free-Form CV Documents in Multiple Languages.

The use of two natural language processing algorithms to extract important data from an unstructured multilingual CV has provided a solution for selecting relevant document parts and the similar particular information at the low hierarchy level (VUKADIN, et al., 2021). It uses a machine learning method in NLP to obtain a high level of extraction accuracy. In their practice, authors used the transformer architecture and its application of the encoder part.

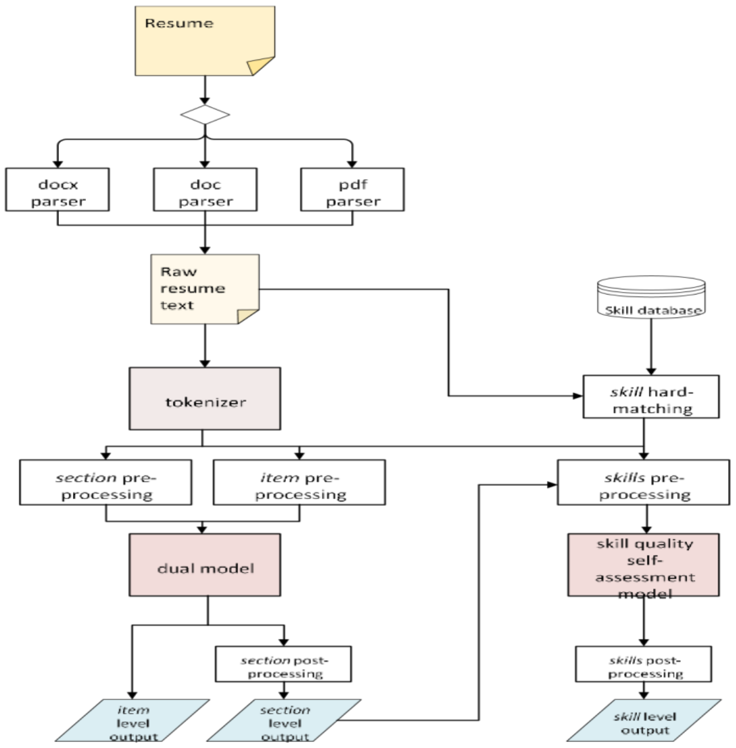


Figure 3: The Information Extraction System from Free-form CVs High-Level Overview

A dual model was developed to extract both section and item level information from a CV document. A self-assessment model of skill proficiency categorizes the retrieved Skills section from the dual model. The authors claim that they have solved the CV parsing challenge by building an NLP system. The recently introduced tokens [NEW LINE] and [SKILL] are shown to have trained to perform as expected. In the future, adding additional CVs in other languages to the dataset would improve performance in other languages.

2.3 Information Extraction from Resume Documents in PDF Formats

It focuses on the problem of extracting data from PDF-format resumes and proposes a hierarchical extraction technique. Resume papers break a page into blocks using heuristic criteria, categorize each block using a Conditional Random Field (CRF) model, and approach the detailed information extraction problem as a sequence labeling issue. According to the authors, the layout-based features have shown to be especially beneficial for semi structured information extraction challenges, increasing the average F1score by more than 20% in testing. In comparison to HTML resumes, PDF resumes usually include more detailed information. They want to experiment with various page segmentation techniques in the future to improve their understanding of the document's layout and content. The report's method is explained below.

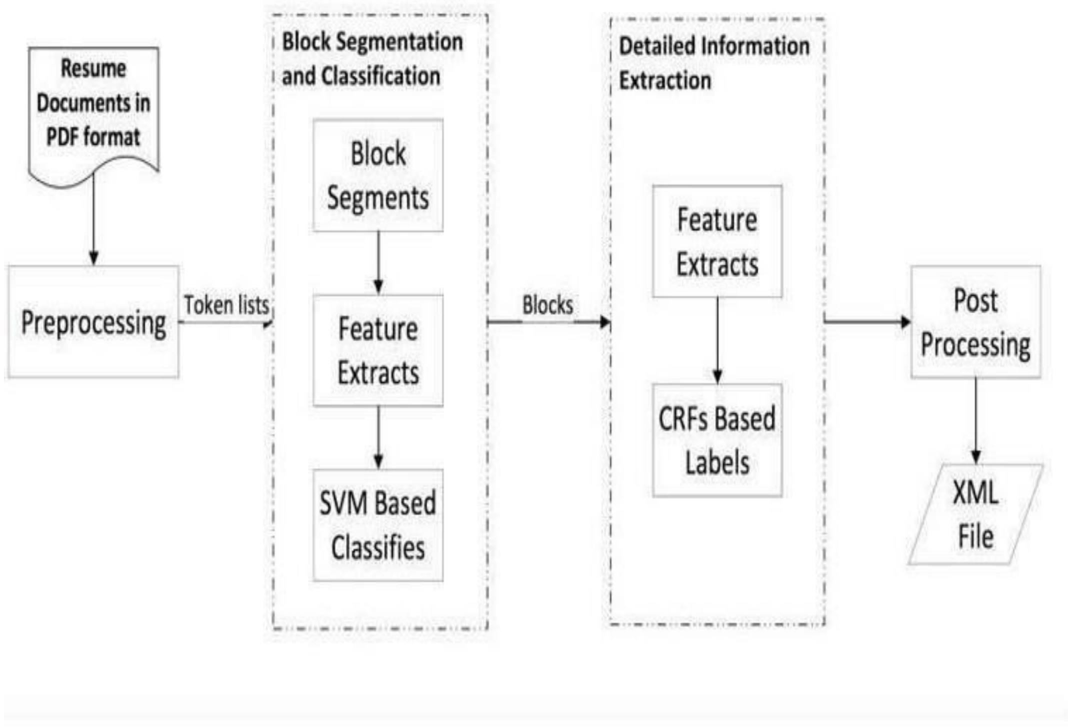


Figure 4: Workflow of Information Extraction from Resume

2.4 Study of Information Extraction in Resume

The proposes Text Segmentation, Rule-based Named Entity Recognition, Deep Neural Network Find Name Entities, and Text Normalization approaches that automatically retrieve and process multiple resumes formats. To label the sequence in the resume and then extract Name Entities from the labeled line, the author defined the Deep Learning model as a mix of Convolutional Neural Networks, Bidirectional Long Short-Term Memory, and Conditional Random Field. The developer collected promising findings with over 81 percent F1 for NER when experimenting on a medium-sized collection of CV information and compared this model to other systems. However, there are some flaws in this system that might be addressed. The quantity of data to train the model is insufficient. Thus, the more data, the more accurate the model will be. In addition, the model's calculations necessitate the use of sophisticated computers. The author wishes to improve the hardware systems to improve performance in the future.

2.5 Automatic Extraction of Usable Information from Unstructured Resumes to Aid Search

The proposes a natural language processing (NLP) system that focuses on automated information extraction from resume to facilitate speedy resume search and management for structured and unstructured resumes. According to them, it is a two-pass method in the first pass, the resume is divided into a group of successive labeled blocks that indicate the broad data that the block includes. Then, in the second pass, information is retrieved in detail. They applied a variety of heuristics and pattern matching algorithms for the extraction procedure. According to the results of experiments conducted on many resumes, the suggested system can grip a wide range of resumes in various document formats with an accuracy of 91% and a recall of 88%. In the future, they expect to enhance the accuracy with which a CV is picked for high-skilled employment. The data flow and completed tasks are represented in the system diagram below.

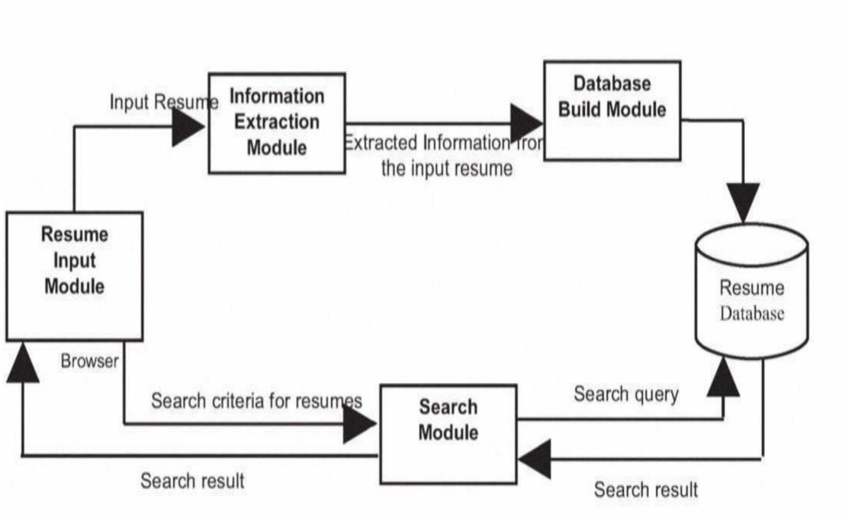


Figure 5: Workflow of the System

2.6 conclusion of literature review

Modern science has developed an excellent learning and working environment in the twenty-first century. Likewise, AI-based solutions are gaining worldwide recognition and implementation for solving the challenge of filtering necessary applicants for a company. As a result, different information extraction systems are created in the present time, replacing the traditional and time-consuming one-by-one approach with a simple filtering out of what is required for the Company's post.

The reviewed literature addressed a wide range of information extraction concepts and research-based procedures and their main approaches. The majority of the research material uses a combination of ML and DL-based methods. After analyzing all of the review findings, it is easier to conclude that extracting meaningful information from resumes using NLP and its different techniques like Regex and Spacy simplifies the recruiting process significantly minimizes time complexity in a larger way.

CHAPTER-3

EXISTING AND PROPOSED METHODOLOGY

3.1 EXISTING METHODOLOGY:

The existing system is a traditional Machine Learning based system. It gives lower rates of accuracy. It has lower efficiency. It gives lower inaccurate results. This system may lead to loss of human potential.

DISADVANTAGES OF EXISTING SYSTEM

- Low retention
- Low recruitment

3.2 PROPOSED METHODOLOGY:

Our system is a resume ranking that use natural language processing (NLP) and regular expression libraries to analyses the resume. Every organization has a site that all companies can access, where they can also see the resumes of the applicants for the position that have been gathered. Every resume submitted via this website will be screened and examined in line with the unique criteria of the company. Additionally, the project aims to gather crucial data from social media wireless networks like the professional networking website for potential employment applications. After gathering outstanding candidates throughout many regions, they attempt to reduce unfair and prejudiced behaviors to streamline the recruiting procedure.

Scores can then be given to the resumes and they can be ranked from highest match to lowest match. This ranking is made visible only to the company recruiter who is interested to select the best candidates from a large pool of candidates. This is done with the aim to aid the recruiters of any company from the long and tedious task of viewing and analyzing thousands of candidates resumes. The calculated ranking scores can then be utilized to determine best-fitting candidates for that particular job opening Since the dynamic model leverages NLP, it gives the output instantly. While going through all these pipelines, it will score each resume and give out accurate output with higher efficiency.

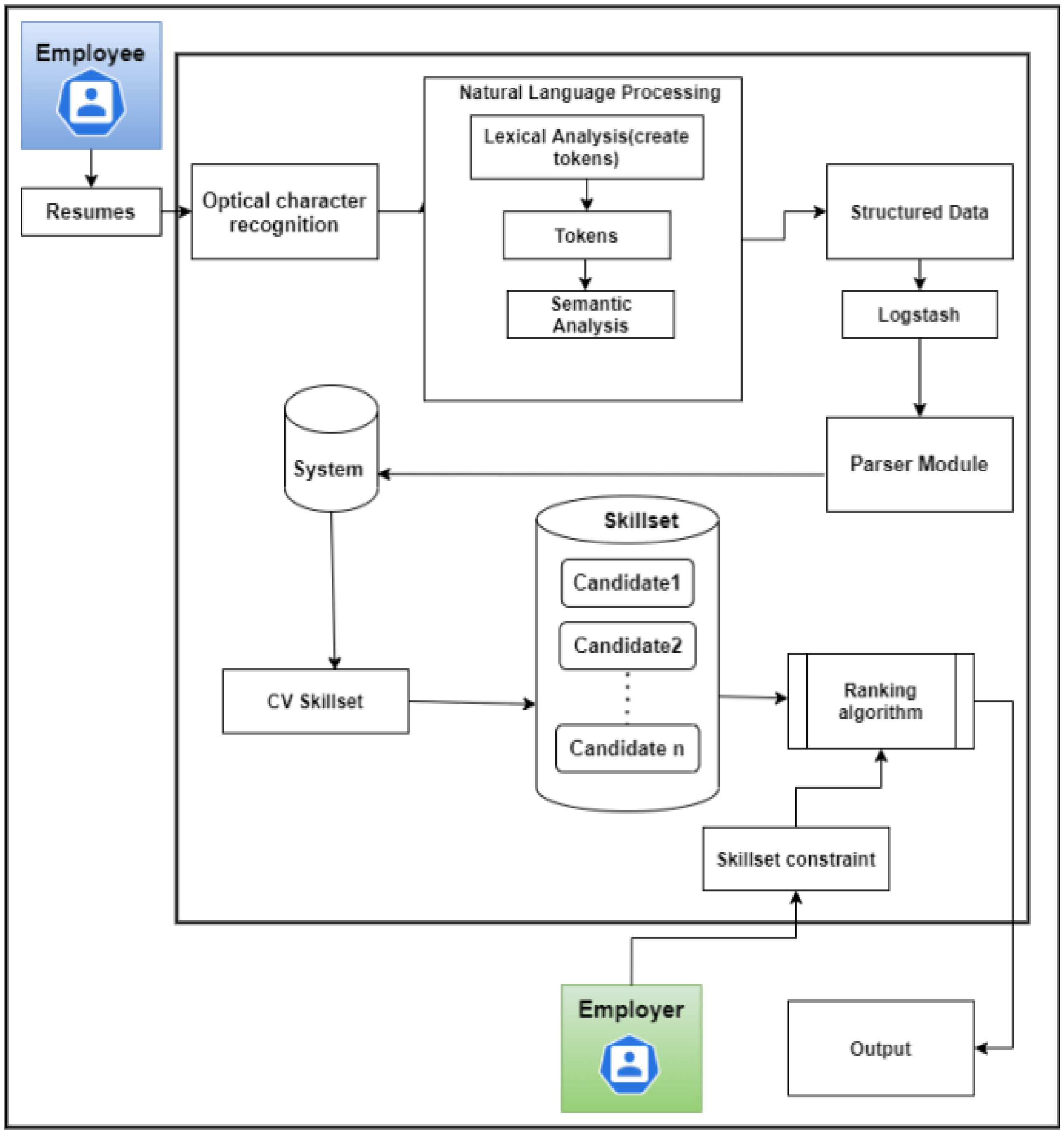


Figure 6: Proposed System Architecture

To use the Resume Parser system both the employee and company needs to provide the detailed resume by employee and job requirements skill set constraints by company. The working of the system architecture is as follows:

1. A web portal is provided to the HR, to define the constraints and the required skill sets of the company, on which the applicants are to be judged.

2. The Candidate also needs to use the web portal to create his account and upload the resume. The Resumes uploaded by the candidate are fetched and fed to the OCR.

3. As the resumes uploaded can be of any format such as '.txt', '.pdf', '.doc', '.docx', '.odt' we will use Optical Character Recognition to convert the resume to a single text format.

4. The Converted resume is then Fed to the Natural Language Processing module it takes the plain text as input and converts it into meaningful data. Using NLP, we are going to parse the resume, NLP requires the following for parsing:

1) Lexical Analysis: It is the first phase of NLP parsing, in which the plain text input is segmented into words and paragraphs and then the tokens are created.

2) Syntactic Analysis: In Syntactic analysis the analysis of the grammar and the arrangement of words in a meaningful manner is checked, sentences like “College goes to girl” is rejected.

3) Named Entity Recognition (NER): One of the problems with using the same NLP module for all the companies is the jargon and words that mean something for that company’s domain and may mean something else in general. This hindrance is overcome in our system with the help of “Named Entity Recognition” or NER.A named entity is an object that exists in the real world.

5. The company or the HR of the company will provide the skill set requirement for the job posted by them. Further the skills mentioned by the applicants in the resume which was tokenized is compared with the required skill set.

6. Using queries inbuilt in the whole process, the applicant resumes will be scored and then they will be provided in the form of a bar graph and pie chart with whole statistics.

7. At the end, the percentage of the bar graph will be used to sort the applicants. A

final list of shortlisted applicants for the further placement process will be generated.

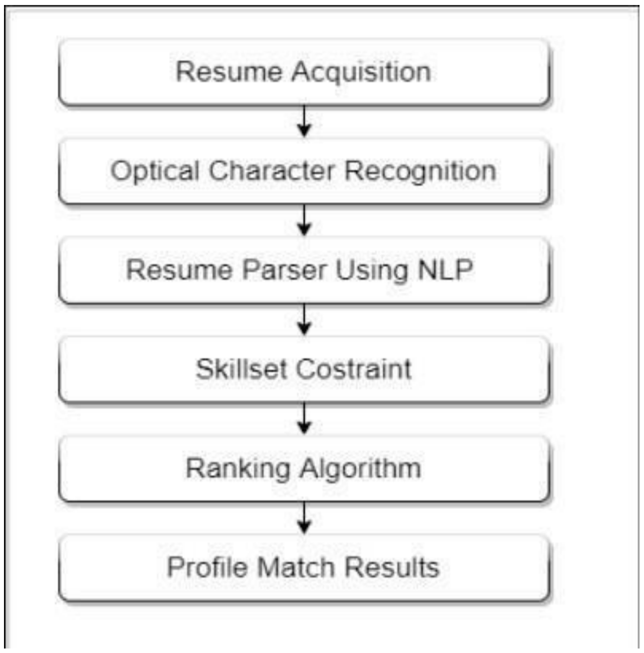


Figure 7: overview

ADVANTAGES OF PROPOSED SYSTEM

- Time saving.
- Cost saving.
- Hire with Quality.

CHAPTER-4
METHODOLOGY

4.1 DATA COLLECTION

Because training data is not necessary, I only need testing data for my "Hireability" system. So, I got some data from Kaggle's "Hire A Perfect Machine Learning Engineer" and the rest through beam jobs. There were 48 resume templates available in the beam jobs collection.

4.2 DATA PRE-PROCESSING

Natural Language Processing is a subfield of data science that works with textual data. When it comes to handling the Human language, textual data is one of the most unstructured types of data available. NLP is a technique that operates behind the it, allowing for extensive text preparation prior to any output. Before using the data for analysis in any Machine Learning work, it's critical to analyze the data. To deal with NLP-based problems, a variety of libraries and algorithms are employed. For text cleaning, a regular expression(re) is the most often used library. The next libraries are NLTK (Natural language toolkit) and spacy, which are used to execute natural language tasks like eliminating stop words.

Preprocessing data is a difficult task. Text preprocessing is done in order to prepare the text data for model creation. It is the initial stage of any NLP project The following are some of the preprocessing steps:

- Removing Stop words
- Lower casing
- Tokenization
- Lemmatization

1. Tokenization

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

A. Sentence tokenization: using the sent_tokenize () function, dividing a paragraph into a collection of sentences.
B. Word tokenization: using the word_tokenize () technique, dividing a statement into a list of words.

2. Removing Stop Words

To eliminate noise from data, data cleaning is essential in NLP. Stop words are the most frequently repeated words in a text that give no useful information. The NLTK library includes a list of terms that are considered stop words in English. [I, no, nor, me, mine, myself, some, such we, our, you'd, your, he, ours, ourselves, yours, yourself, yourselves, you, you're, you've, you'll, most, other] are only a few of them.

The NLTK library is a popular library for removing stop words, and it eliminates about 180 stop words. For certain difficulties, we can develop a customized set of stop words. Using the add technique, we can easily add any new word to a collection of terms.

3. Lemmatization

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

4. Lower casing

When the text is in the same case, a computer can easily read the words since the machine treats lower and upper case differently. Words like Cat and cat, for example, are processed differently by machines. To prevent such issues, we must make the word in the same case, with lower case being the most preferable instance. In python lower () is a function that is mostly used to handle strings. The lower () function accepts no parameters. It converts each capital letter to lowercase to produce lowercased strings from the provided string. If the supplied string has no capital characters, it returns the exact string.

4.3 MODULE IMPLEMENTATION

This part contains information about data collecting, how the code is working in the project, different libraries and modules utilized in the project, and so forth. In today's world, everyone wants to complete the tasks as early as possible, but it all relies on the methods they use. There may be a point when we must write many lines of code; if we continue writing code on our own, this will take time. A Library is a group of interconnected modules. It provides code bundles that may be reused in a variety of programs. Modules also play a role in Python. We may define commonly used functions as modules and import them into a code wherever there is a necessity, rather than repeating the same code in multiple programs and making the code complicated. The following are some of the most important libraries and modules:

1. Pandas

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

2. IO

   The io module is used to control file-related input and output activities. The advantage of utilizing the IO module is that we can extend the capability to allow writing to Unicode data using the classes and methods provided.

3. Utils

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

4. Spacy

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

5. pprint

   The pprint module allows to "pretty-print" any python data structure in a more understandable and well-formatted fashion.

6. Matcher

   The Matcher assists in the finding of words and phrases by utilizing rules that describe their token properties. After applying the matcher on a Doc, we can see the matched tokens in context.

7. Multiprocessing

   In Python, multiprocessing is a built-in package that allows the system to execute many processes at the same time.

8. Warnings

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

9. Nltk.corus

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

10. Re

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

11. NLTK

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

12. Docx2txt

    A pure Python-based library for extracting text from docx files.

13. Constants

    The module is just a separate file that contains variables, functions, and other data that is imported into the main file.

14. String

    This module will allow to retrieve string constants easily. Cap words is a single utility function in the Python string module.

15. Pdfminer

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

4.4 SYSTEM REQUIREMENTS

4.4.1 *Hardware Requirements*:

- PC with 128 GB of SSD, Python Development Environment and a stable internet connection
- Windows and Linux for testing
- Tkinter

4.4.2 *Software Requirements*:

- Figma for logo design
- Google Chrome or Mozilla (Alternate) for research and debugging
- Visual Studio Code for Python Development
- Git for version control
- Google Drive for the document (Word/ PDF) controls
- Microsoft Visio/draw.io for creating WBS and other diagrams
- Excel for creating Gantt Chart
- MySQL for storing data if needed
- Pip for installing python packages
- NLTK and SpaCy for resume parsing and extraction
- Flask for developing web application
- Jupyter Notebook or VsCode for evaluating and exploring data

4.5 SYSTEM WORK FLOW

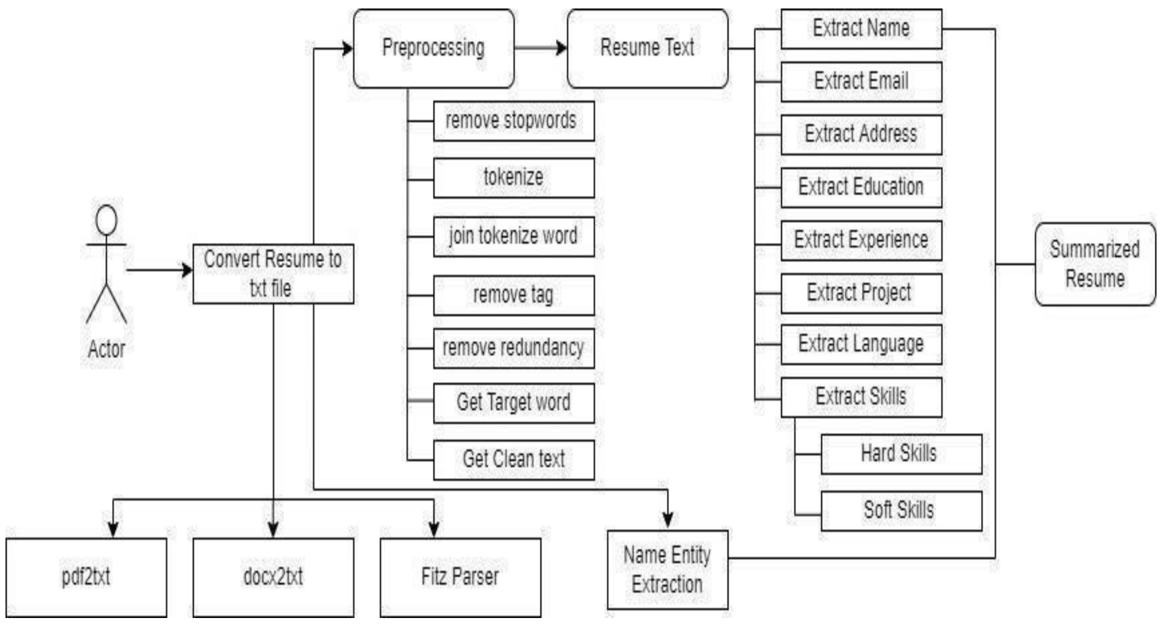


Figure 8: System Work Flow

The first step after importing the libraries is to read the resume. As resumes do not have a set file type, they can be in any format, including.pdf,.doc, and.docx. As a result, our key priority is to read the resume and turn it into plain text. I used Pdfminer and doc2text Python modules for this. Text can be extracted from.pdf,.doc, and.docx files using these modules.

a) Extract the text from pdf

This step defined a function to extract the plain text from .pdf files. The pdf_path parameter gives path to pdf files that is to extracted. We will begin by establishing a resource_manager instance. Then, using Python's io module, we will then build a file-like object. The next step is to build a converter. We'll use the TextConverter for that. Finally, we construct a PDF interpreter object that will extract the text from our resource management and converter objects. It returns the iterator of string of extracted text.

b) Extract the text from Doc files

This step defined a function to extract the plain text from .docx or .doc files. The doc_path parameter gives path to .doc, .docx files that is to extracted. It returns the string of extracted text.

c) Detect the file extension

This step defined a function to detect the file extension and call the text extraction function accordingly. If the file is on .pdf or .docx or .doc then it returns the text. The file_path parameter gives path of the file of which text is to be extracted. After that the param_extension is the extension of the file 'file_name'.

d) Extract the entities

This step defined a function to extract all the raw text from the sections of resume. I have already created the list for sections of resume in constant.py file. The split () function divides a string by the separator supplied and produces a list object containing string elements. Whitespace character, such as, \t, \n, are used as the separator by default. The text parameter gives the raw text of the resume. It returns the dictionary of the entities.

e) Extract the email

This step defined a function to extract the email id from the text. The text parameter gives the plain text extracted from the resume files. As we know, email addresses have a set format, which includes an alphanumeric string, a @ symbol, another string, a. (dot), and a string at the last. We can extract such expressions from text using regular expressions.

f) Extract the Name from the resume

This step defined a function to extract the name from the text. The text parameter gives the plain text extracted from the resume files. Whereas the matcher parameter is the object of spacy. matcher. Spacy is a text and language processing module. It includes pre-trained models for tagging, parsing, and entity recognition. The primary goal here is to extract names using Entity Recognition. One of the phases in extracting information from unstructured text is rule-based matching. It recognizes and extracts tokens and words based on patterns. Regular expressions can be used in rule-based matching to extract entities. We can extract a first and last name, which are generally proper nouns, using rule-based matching. It has two PROPN POS tags in it (proper noun). As a result, the pattern is made up of two objects, each with PROPN POS tags. The NAME and the match id are then used to add the pattern to Matcher. Finally, matches are found using their starting and ending indexes.

g) Extract the Skills from the resume

After gathering some basic information about the individual, let's focus on what matters most to recruiters: skills. For that, I've created a function that extracts skills from spacy text. The object of spacy. tokens is the NLP text parameter. Noun chunks are extracted from the text using the noun chunks parameter. A method known as tokenization can be used to extract skills. In order to perform tokenization, we need to establish a sample dataset against which we can compare the abilities in a given résumé. a comma separated values (.csv) file with the necessary skill sets. For instance, if I'm in HR and need someone with NLP, ML, and AI abilities, I can create a csv file with the following keywords: machine learning, artificial intelligence, natural language processing. Following that, we renamed the file to skills.csv. Then, we can proceed to tokenize our extracted text and compare the skills to those in skills.csv. The panda’s module will be used to read a csv file. After reviewing the material, we will remove all of the stop words from our resume. At last, it returns the list of skills extracted.

h) Extract the Technical Skills

Non-technical skills are extracted using the same procedure as technical skills.

i) Extract the Education and Year

The degree and the year of graduation will be the details we will extract. If XYZ finished BS in 2022, for example, we will extract a tuple like ('BS', '2022'). We'll need to get rid of all the stop words for this. We'll use the nltk module to load a long list of stop words, remove from our resume text. HR teams are quite specific about the level of education or degree necessary for a given job. As a result, we'll be putting together an EDUCATION list that will include all of the similar degrees that meet the standards.

j) Extract the Address

This step defined a function to extract the address from the text using Spacy. GPE often refers to geopolitical entities such as cities, states, nations, continents, and so on. In addition, LOCATION also can symbolize well-known mountains, rivers, and other natural features. The act of identifying named entities in unstructured text and then categorizing them into pre-defined categories is known as Named entity recognition (NER).

k) Extract the Experience

This step defined a function to extract the experience from the resume text. The resume_text parameter gives the plain text extracted from the resume files. After doing different preprocessing part. It searches the word ‘experience’ in the chunk and then print outs the text after it.

4.6 ACTIVITY DIAGRAM

The behavior of a system is depicted in an activity diagram. The control flow from a start point to a completion point is represented in an activity diagram, which shows the numerous decision routes that exist while the activity is being performed. The activity diagram for the admin management system is shown below.

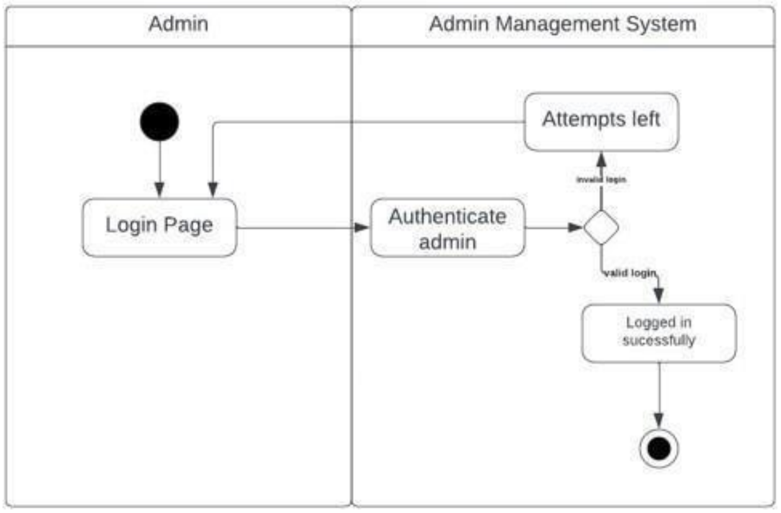


Figure 9: Activity Diagram of Admin Management System

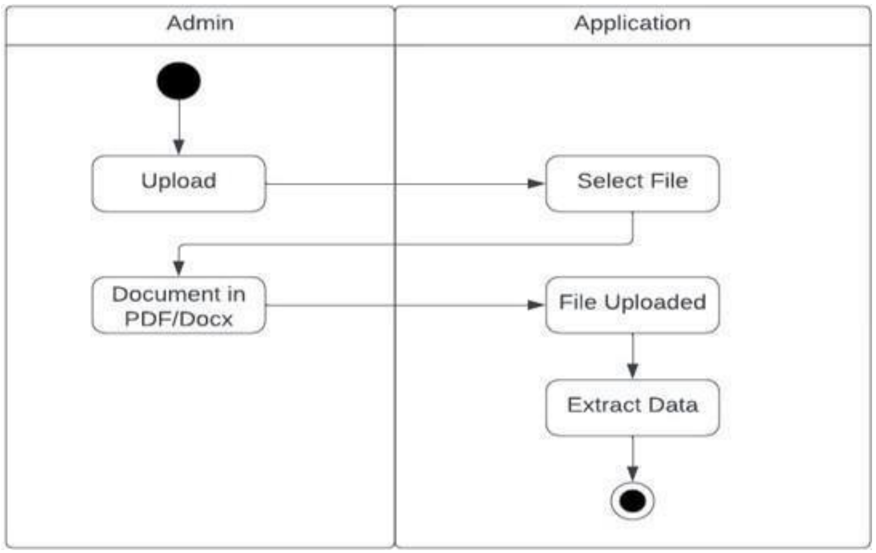


Figure 10: Activity Diagram for Parsing System

4.7 USE CASE DIAGRAM

The interactions between the system and its actors are mostly identified with the use case diagram. In use-case diagrams, use cases and actors define what the system does and how the actors interact with it. There is just one user in the diagram below, and that is the admin. Admin can log in, navigate to the upload page, upload their resume, and view the extracted data.

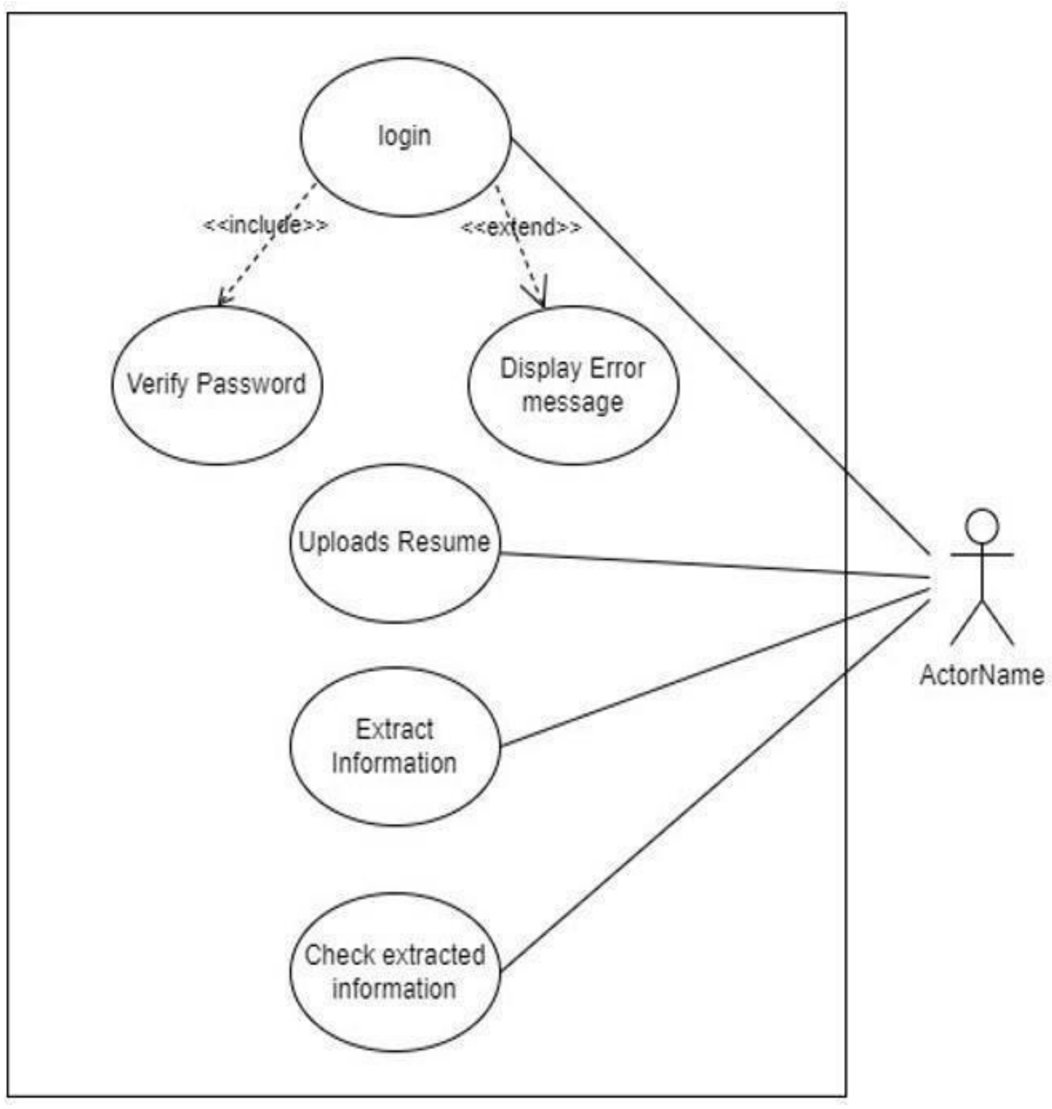


Figure 11: Use Case Diagram

4.8 SEQUENCE DIAGRAM

A sequence diagram displays the order in which items interact in a sequential way. It shows how and in what sequence the components of a system work together. The sequence diagram for the admin management system is shown below.

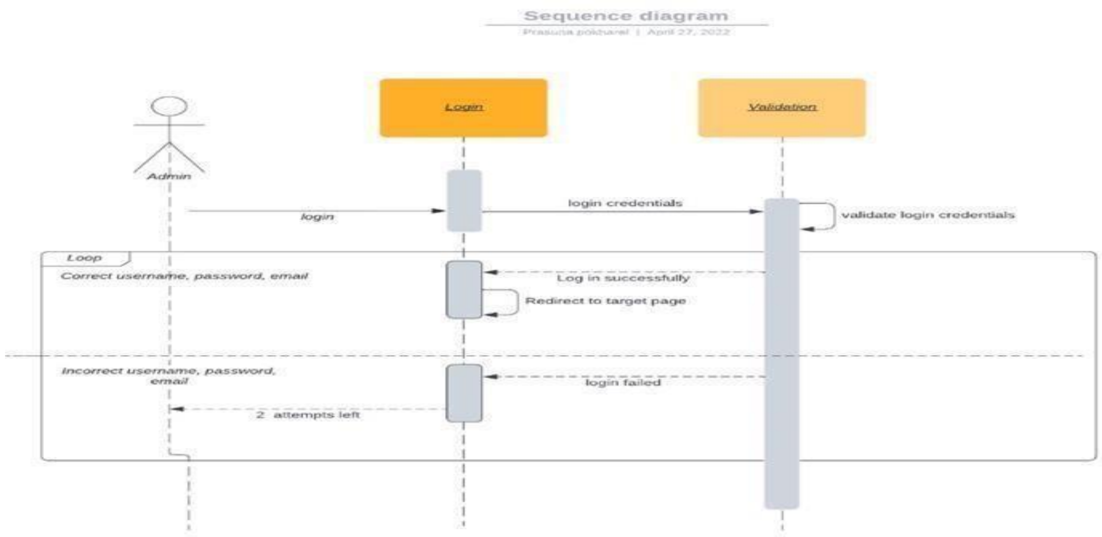
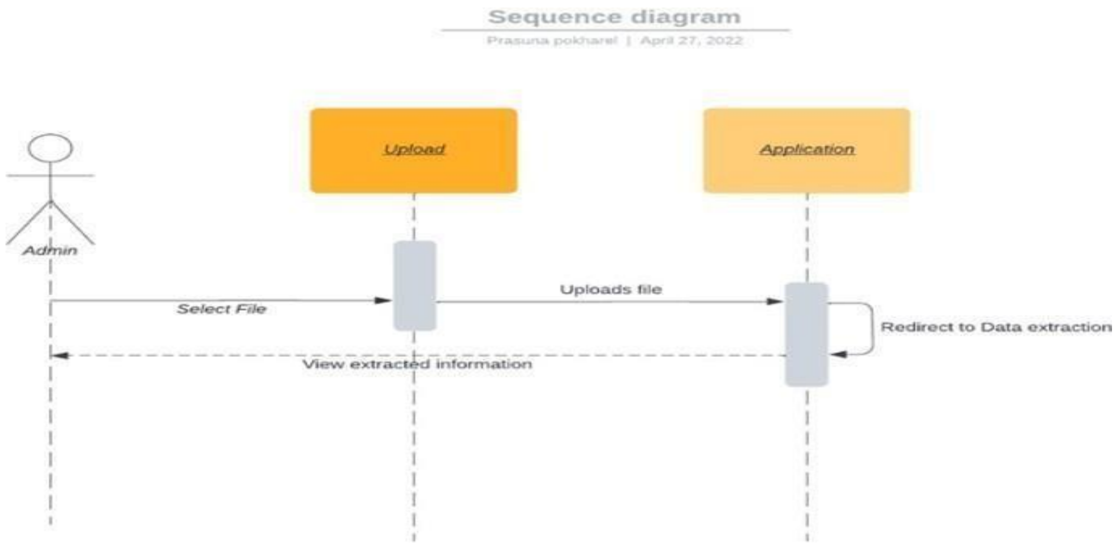


Figure 12: Admin Management System



Sequence diagram
Prasuna pokharel | April 27, 2022
Upload
Application
Admin
Select File
Uploads file
Redirect to Data extraction
View extracted information

Figure 13: File Upload System

CHAPTER-5
RESULTS AND DISCUSSIONS

Resume parsing using NLP is a common application of natural language processing (NLP) techniques. The goal of resume parsing is to extract relevant information from a job candidate's resume, such as their contact information, education, work experience, and skills. This information can then be used to populate a database or applicant tracking system (ATS) to facilitate recruitment and hiring processes. There are several approaches to resume parsing using NLP, including rule-based systems, machine learning models, and hybrid systems that combine both. Rule-based systems use predefined rules to extract information from resumes, such as patterns for identifying email addresses or phone numbers. Machine learning models, on the other hand, use algorithms to learn from large amounts of training data to automatically identify relevant information in resumes. One popular approach to resume parsing using NLP is to use named entity recognition (NER) techniques. NER is a subfield of NLP that focuses on identifying named entities in text, such as people, organizations, and locations. In the context of resume parsing, NER can be used to extract information such as the candidate's name, the name of their previous employers, and the name of the schools they attended. Another important aspect of resume parsing is extracting information about the candidate's skills. This can be challenging, as skills can be expressed in many different ways, and may include both technical and soft skills. One approach to skill extraction is to use keyword-based techniques, where a list of relevant skills is predefined and used to match against the text of the resume. Another approach is to use machine learning models to identify patterns in the text that are indicative of certain skills. Overall, resume parsing using NLP is a complex task that requires a combination of rule-based systems, machine learning models, and domain-specific knowledge. While there are many challenges associated with this task, advances in NLP and machine learning are making it increasingly feasible to automate the process of resume parsing and improve the efficiency and accuracy of recruitment and hiring processes.

CHAPTER-6

CONCLUSION

6.1 CONCLUSION

The Organization gets a lot of applicants for each job posting. It is difficult to locate the proper candidate’s application from a sea of resumes. It takes a lot of time and resources to categorization a candidates resume. An automated resume parser is developed to get the necessary information from the resume of a candidate. The candidate’s resume will be suggested by the system to HR. The email was sent to the candidates based on their resume rank.

A normal resume is a compilation of information about a person's work experience, academic background, qualifications, and personal details. These elements might be present in a variety of ways or not at all. It's difficult to keep up with the jargon used in resumes. A resume is made up of corporate names, institutions, degrees, and other information that can be written in a variety of ways. It will take time to review all the resume by an individual.

Machine works faster than human and their accuracy to do any task was also good. Therefore, I have made a system which includes machine learning that extract the important information from resumes within a minute or less than a minute. The hiring individual can use this system for hiring any individual.

6.2 FUTURE WORK

As stated in the paper, the project has a broad reach in the current context. The proposal's majority of proposed features have been implemented. So, if I continue working on this project, I intend to create a database for the system where the admin may keep the extracted data. Further, future study will include a more in-depth examination of certain techniques, further research on other libraries, and new approaches to explore different methods.

6.3 RESEARCH ISSUES

There was some uncertainty when it came to select the right libraries and modules for data extraction. After conducting research, I was able to select a library. Following that, obtaining the appropriate dataset was difficult. I looked for a number of resumes dataset but couldn't find one in the correct format. After that, I went online and found several templates and started using them for training data.

6.4 IMPLEMENTATION ISSUES

While developing the system, there were a number of technical difficulties. Following the development of the module, the next step is to integrate it into the system. I was having trouble integrating the module while working on the flask. I tried everything but couldn't get the component to work. As a result, I switched to Django as my framework. I was able to integrate after reading through several python and Django documentation. The integration part took up more time than expected.

Despite the time constraints, the majority of the features proposed in the proposal were implemented. However, the Tika parser library, which I specified in the proposal for the information extraction section, has not been implemented. It is used to detect the kind of document. As I began working on the project, I found that Spacy and Nltk were more suitable to my requirements, so I chose themover Tika.

REFERENCES

[1] Smart Innovation, Systems and Technologies book series (SIST, number 72), Abeer Zaroor, Mohammed Maree, Muath Sabha, 2017.

[2] "User-Based Efficient Video Recommendation System," Lecture Notes on Data Engineering and Communications Technologies, Vol. 26, pp. 1307–1316, 2019, by Albert Mayan J, Aroul Canesaane R, Jabez J, Kamalesh M.D, and Rama Mohan Reddy G.

[3] Intelligent Recruitment System Using NLP, Anushka Sharma, Smiti Singhal, and Dhara Ajudia, International Conference on Artificial Intelligence and Machine Vision (AIMV), 2021.

[4] Abdul Samad Ebrahim Yahya, Fahad, and SK Ahmed. "Deep learning inflectional review on natural language processing." Smart Computing and Electronic Enterprise Conference 2018 (ICSEE). IEEE, 2018.

[5] Ahmed SK and Samad Abdul Fahad, Ebrahim Yahya, and. Inflectional analysis of deep learning for processing natural English. 2018 Conference on Smart Enterprise and Computing (ICSEE). IEEE, 2018.

[6] Jitendra Purohit, Aditya Bagwe, Rishabh Mehta, Ojaswini Mangaonkar, and Elizabeth George, "Natural Language Processing based Jaro-The Interviewing Chatbot," 3rd International Conference on Computing Methodologies and Communication (ICCMC), August 2019.

[7] "Blood donation and life saver app," 2017 2nd International Conference on Communication and Electronics Systems (ICCES), pp. 446–451, by M. R. Annish Brislin, J. Albert Mayan, R. Aroul Canessane, and M. R. Anish Hamlin.

[8] "Predicting the Risk of Diabetes Mellitus to Subpopulations Using Association Rule Mining," in Advances in Intelligent Systems and Computing, issue 397, Springer, 2016, pp. 59–69.

[9] Nihar R. Mahapatra and Dena F. Mujtaba. Ethical Aspects of AI-Based Hiring. International Symposium on Technology and Society, IEEE 2019, 2019 (ISTAS). IEEE, 2019.

[10] V. V. Nguyen, V. L. Pham, and N. S. Vu, 2018. Study of Resume Information Extraction. semantic expert.

[11] A CV Parser Model Using Big Data Tools and Entity Extraction Process 2018 IJITCS, Papiya Das, Manjusha Pandey, and Siddharth Swarup Rautaray.

[12] Procedia Computer Science, Volume 167, Pages 2318–2327, 2020, "A Machine Learning Approach for automation of Resume Recommendation system." Rocky Bhatia, Pradeep Kumar Roy, and Sarabjeet Singh Chowdhary.

[13] Rajath V, Riza Tanaz Fareed, and Sharadadevi Kaganurmath, "Resume Classification and Ranking Using KNN And Cosine Similarity," international JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY, volume 10, issue 08, AUGUST 2021.

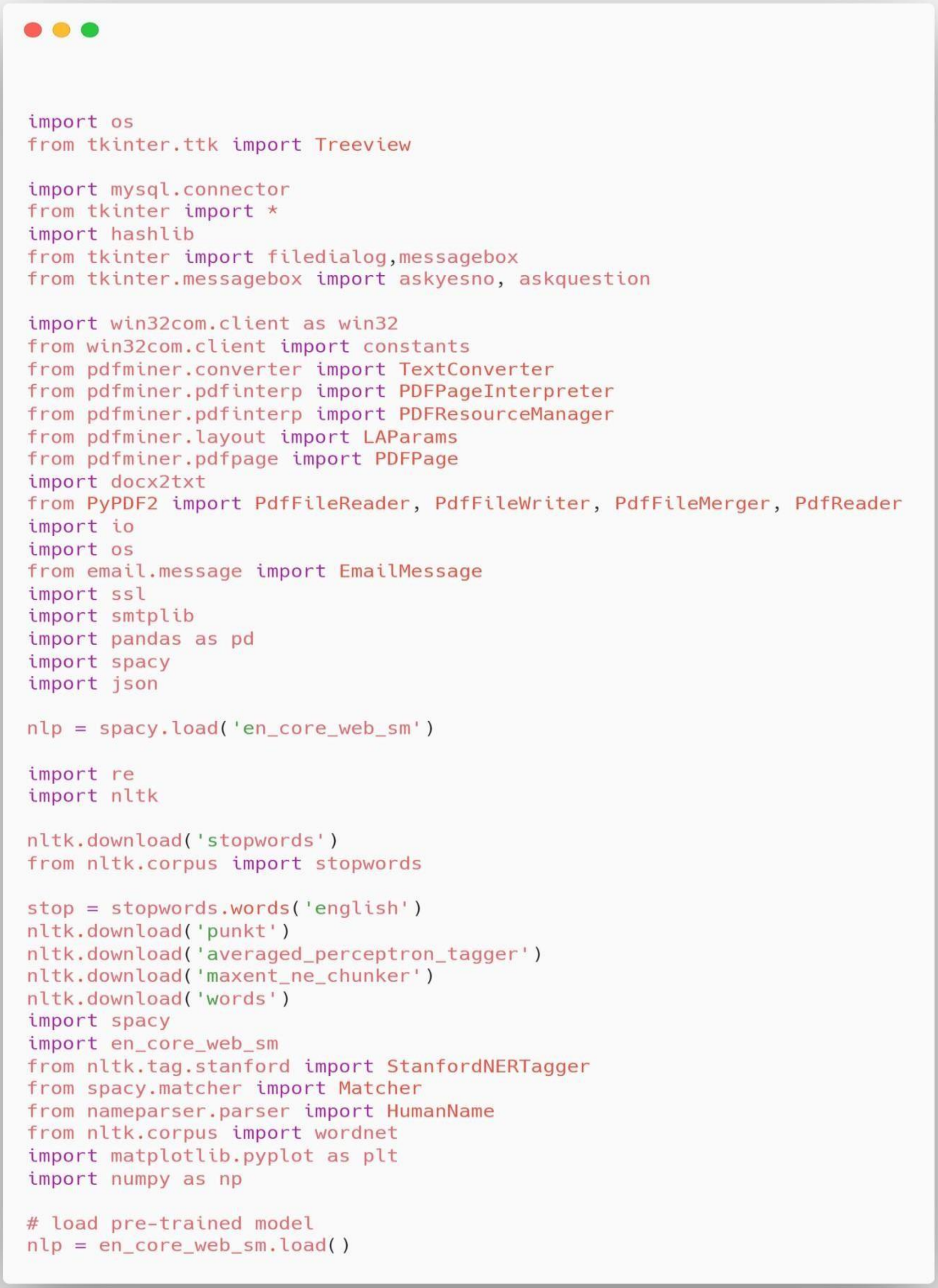
[14] 'Design and Development of Machine Learning based Resume Ranking System', Global Transitions Proceedings, October 2021. Sanjay Revanna, Shashank M. Kadiwal, Tejaswini K., and Umadevi V.

[15] "An Unstructured Text Analytics Approach for Qualitative Evaluation of Resumes," 2015, IJIRAE, by Vinaya R. Kudatarkar, Manjula Ramanavar, and Dr. Nandini S. Sidnal.

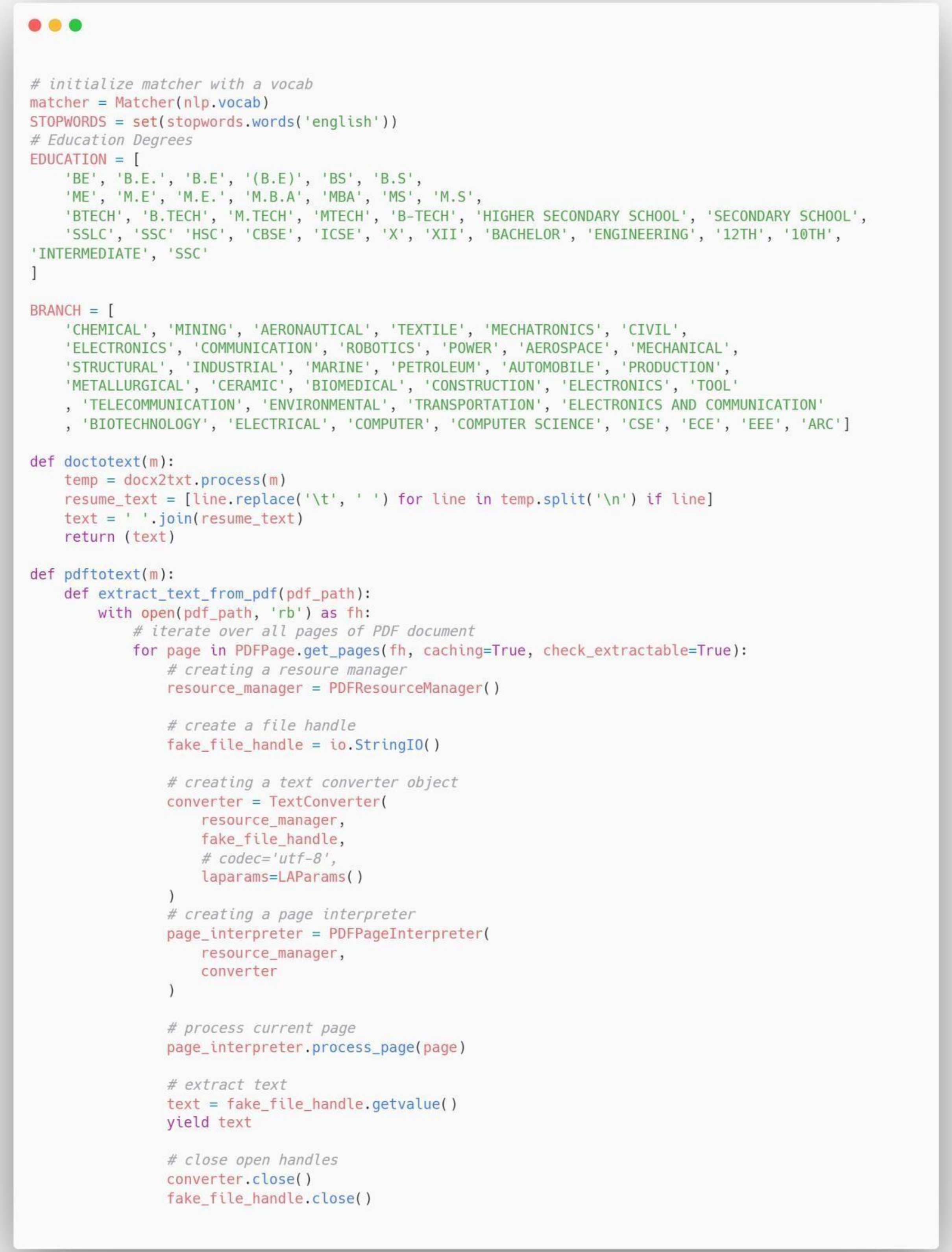
[16] "Resume Net: A Learning-Based Framework for Automatic Resume Quality Assessment," IEEE International Conference on Data Mining (ICDM), 2018, December. Huaizheng Zhang, Xinwen Zhang, Yong Luo, Yongjie Wang, and YonggangWen.

APPENDIX

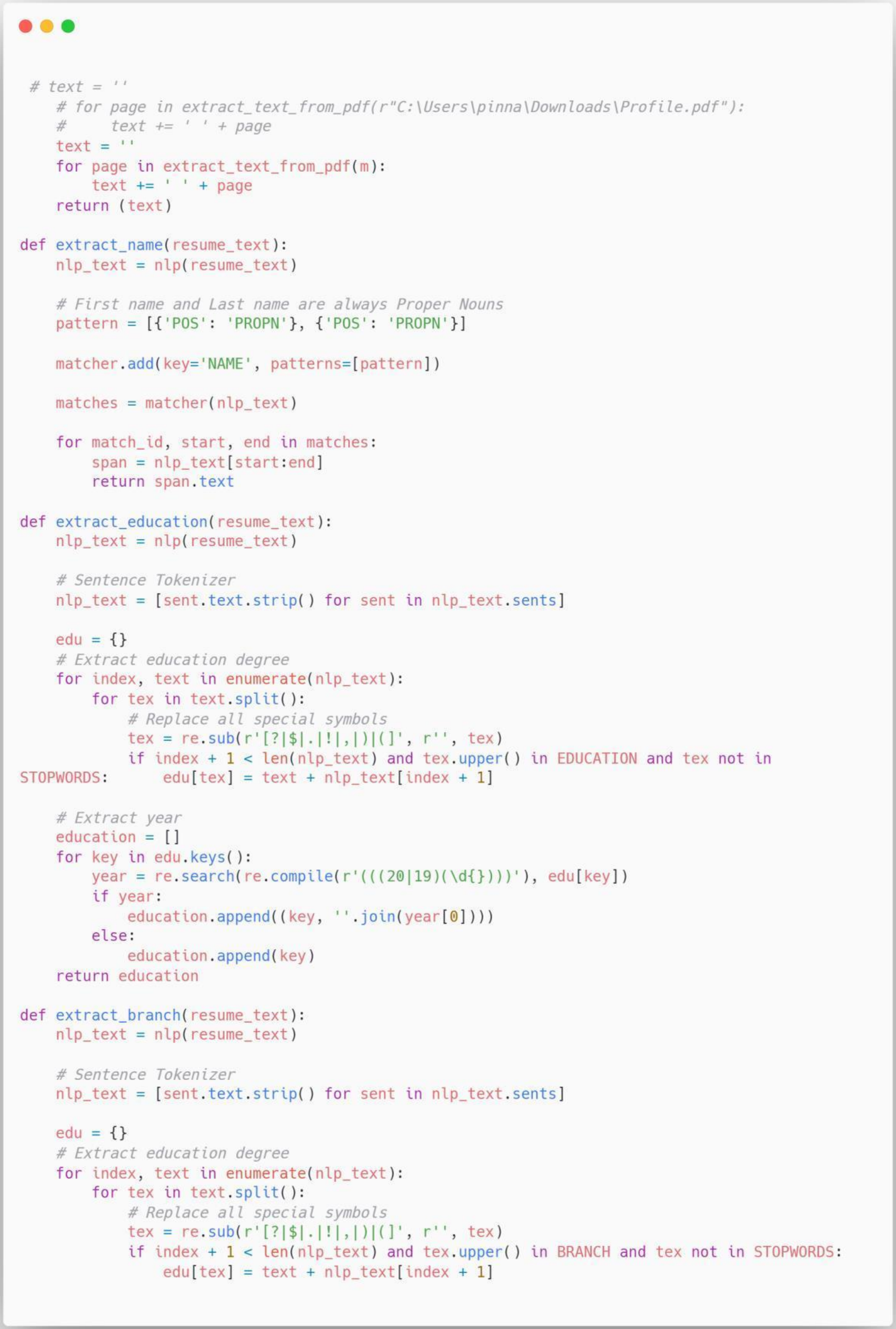
A. SOURCE CODE



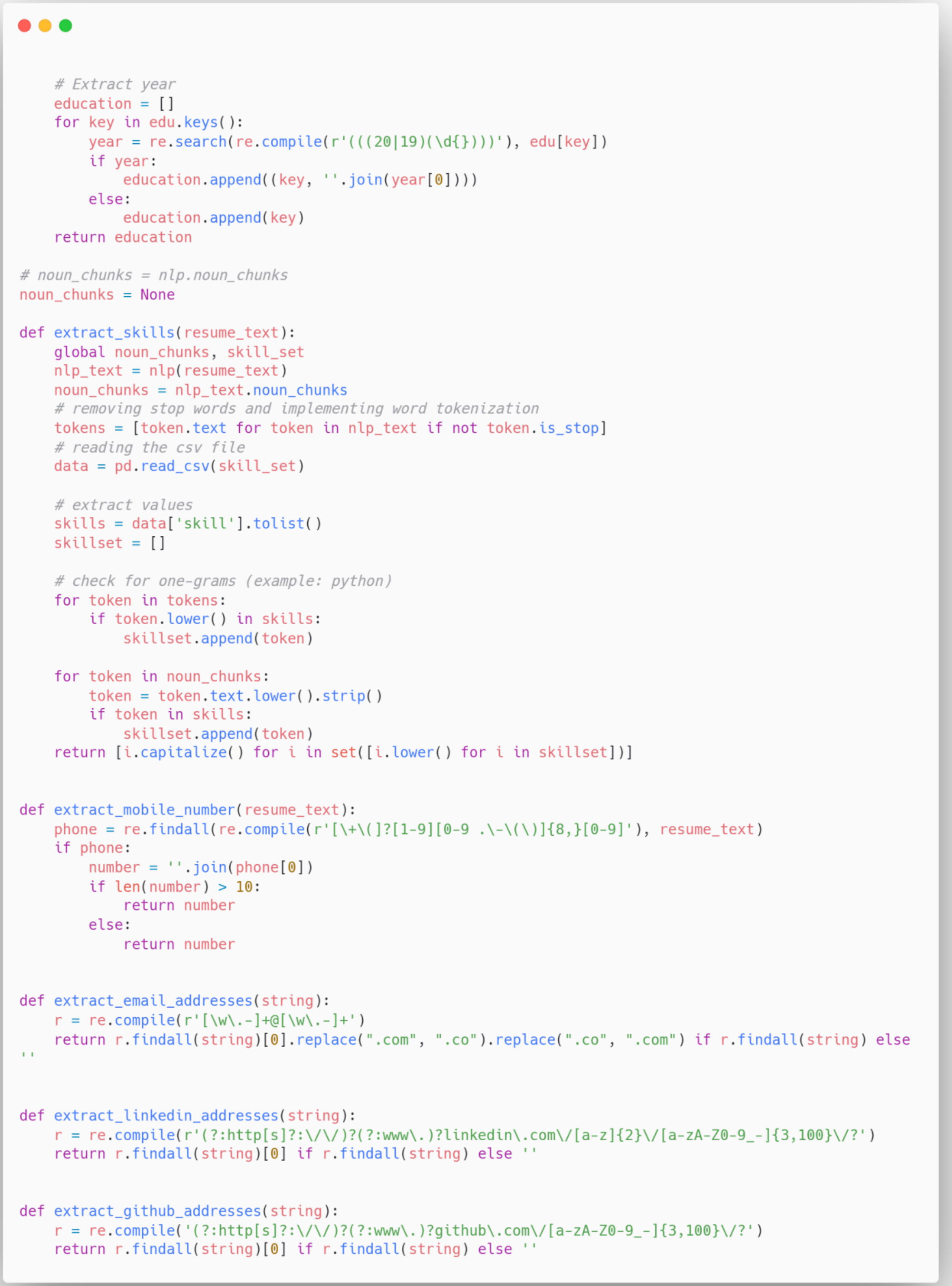
```
import os
from tkinter.ttk import Treeview

import mysql.connector
from tkinter import *
import hashlib
from tkinter import filedialog,messagebox
from tkinter.messagebox import askyesno, askquestion

import win32com.client as win32
from win32com.client import constants
from pdfminer.converter import TextConverter
from pdfminer.pdfinterp import PDFPageInterpreter
from pdfminer.pdfinterp import PDFResourceManager
from pdfminer.layout import LAParams
from pdfminer.pdfpage import PDFPage
import docx2txt
from PyPDF2 import PdfFileReader, PdfFileWriter, PdfFileMerger, PdfReader
import io
import os
from email.message import EmailMessage
import ssl
import smtplib
import pandas as pd
import spacy
import json

nlp = spacy.load('en_core_web_sm')

import re
import nltk

nltk.download('stopwords')
from nltk.corpus import stopwords

stop = stopwords.words('english')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
import spacy
import en_core_web_sm
from nltk.tag.stanford import StanfordNERTagger
from spacy.matcher import Matcher
from nameparser.parser import HumanName
from nltk.corpus import wordnet
import matplotlib.pyplot as plt
import numpy as np

# load pre-trained model
nlp = en_core_web_sm.load()
```



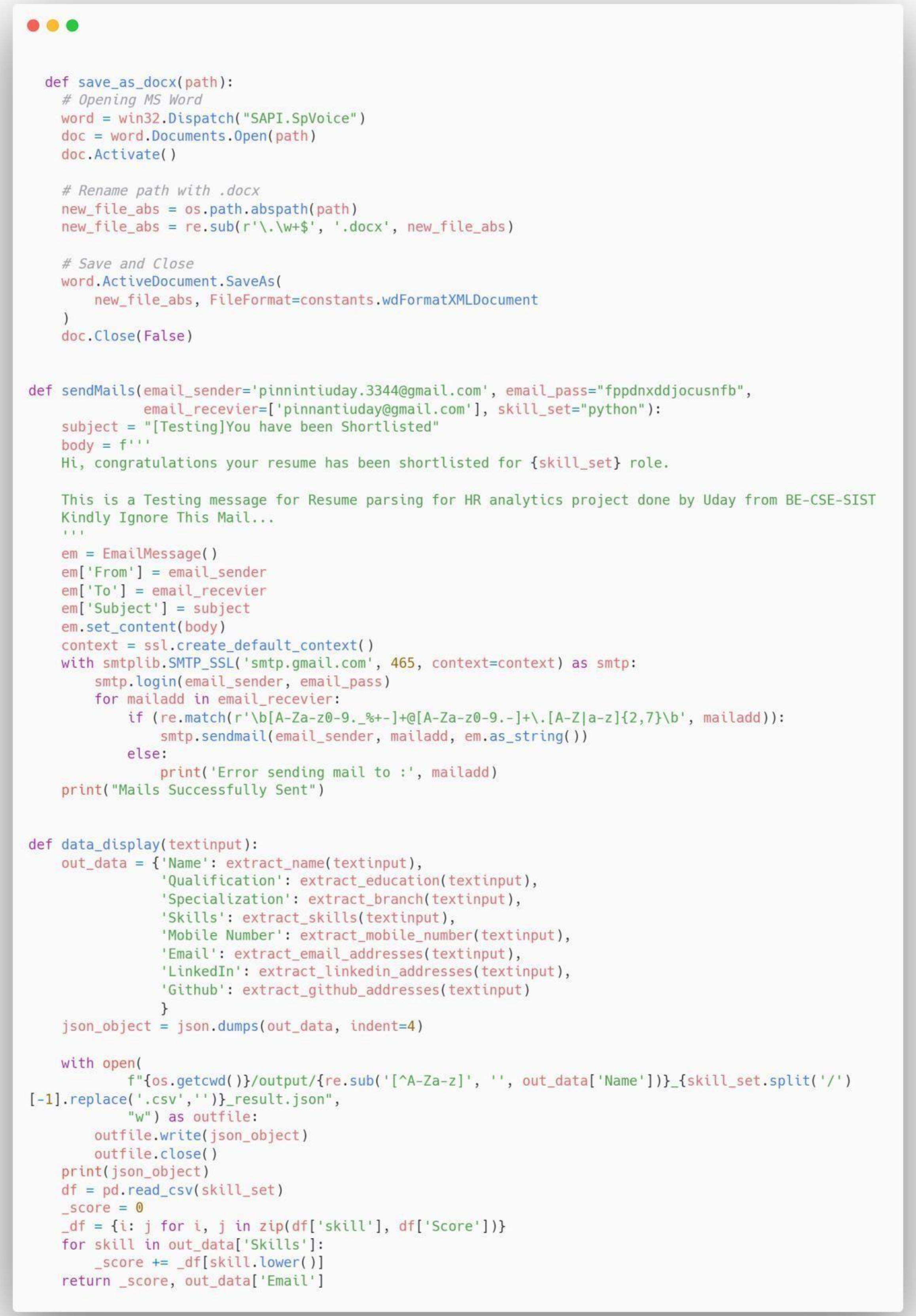
```
# initialize matcher with a vocab
matcher = Matcher(nlp.vocab)
STOPWORDS = set(stopwords.words('english'))
# Education Degrees
EDUCATION = [
    'BE', 'B.E.', 'B.E', '(B.E)', 'BS', 'B.S',
    'ME', 'M.E', 'M.E.', 'M.B.A', 'MBA', 'MS', 'M.S',
    'BTECH', 'B.TECH', 'M.TECH', 'MTECH', 'B-TECH', 'HIGHER SECONDARY SCHOOL', 'SECONDARY SCHOOL',
    'SSLC', 'SSC' 'HSC', 'CBSE', 'ICSE', 'X', 'XII', 'BACHELOR', 'ENGINEERING', '12TH', '10TH',
'INTERMEDIATE', 'SSC'
]

BRANCH = [
    'CHEMICAL', 'MINING', 'AERONAUTICAL', 'TEXTILE', 'MECHATRONICS', 'CIVIL',
    'ELECTRONICS', 'COMMUNICATION', 'ROBOTICS', 'POWER', 'AEROSPACE', 'MECHANICAL',
    'STRUCTURAL', 'INDUSTRIAL', 'MARINE', 'PETROLEUM', 'AUTOMOBILE', 'PRODUCTION',
    'METALLURGICAL', 'CERAMIC', 'BIOMEDICAL', 'CONSTRUCTION', 'ELECTRONICS', 'TOOL'
    , 'TELECOMMUNICATION', 'ENVIRONMENTAL', 'TRANSPORTATION', 'ELECTRONICS AND COMMUNICATION'
    , 'BIOTECHNOLOGY', 'ELECTRICAL', 'COMPUTER', 'COMPUTER SCIENCE', 'CSE', 'ECE', 'EEE', 'ARC']

def doctotext(m):
    temp = docx2txt.process(m)
    resume_text = [line.replace('\t', ' ') for line in temp.split('\n') if line]
    text = ' '.join(resume_text)
    return (text)

def pdftotext(m):
    def extract_text_from_pdf(pdf_path):
        with open(pdf_path, 'rb') as fh:
            # iterate over all pages of PDF document
            for page in PDFPage.get_pages(fh, caching=True, check_extractable=True):
                # creating a resoure manager
                resource_manager = PDFResourceManager()

                # create a file handle
                fake_file_handle = io.StringIO()

                # creating a text converter object
                converter = TextConverter(
                    resource_manager,
                    fake_file_handle,
                    # codec='utf-8',
                    laparams=LAParams()
                )
                # creating a page interpreter
                page_interpreter = PDFPageInterpreter(
                    resource_manager,
                    converter
                )

                # process current page
                page_interpreter.process_page(page)

                # extract text
                text = fake_file_handle.getvalue()
                yield text

                # close open handles
                converter.close()
                fake_file_handle.close()
```



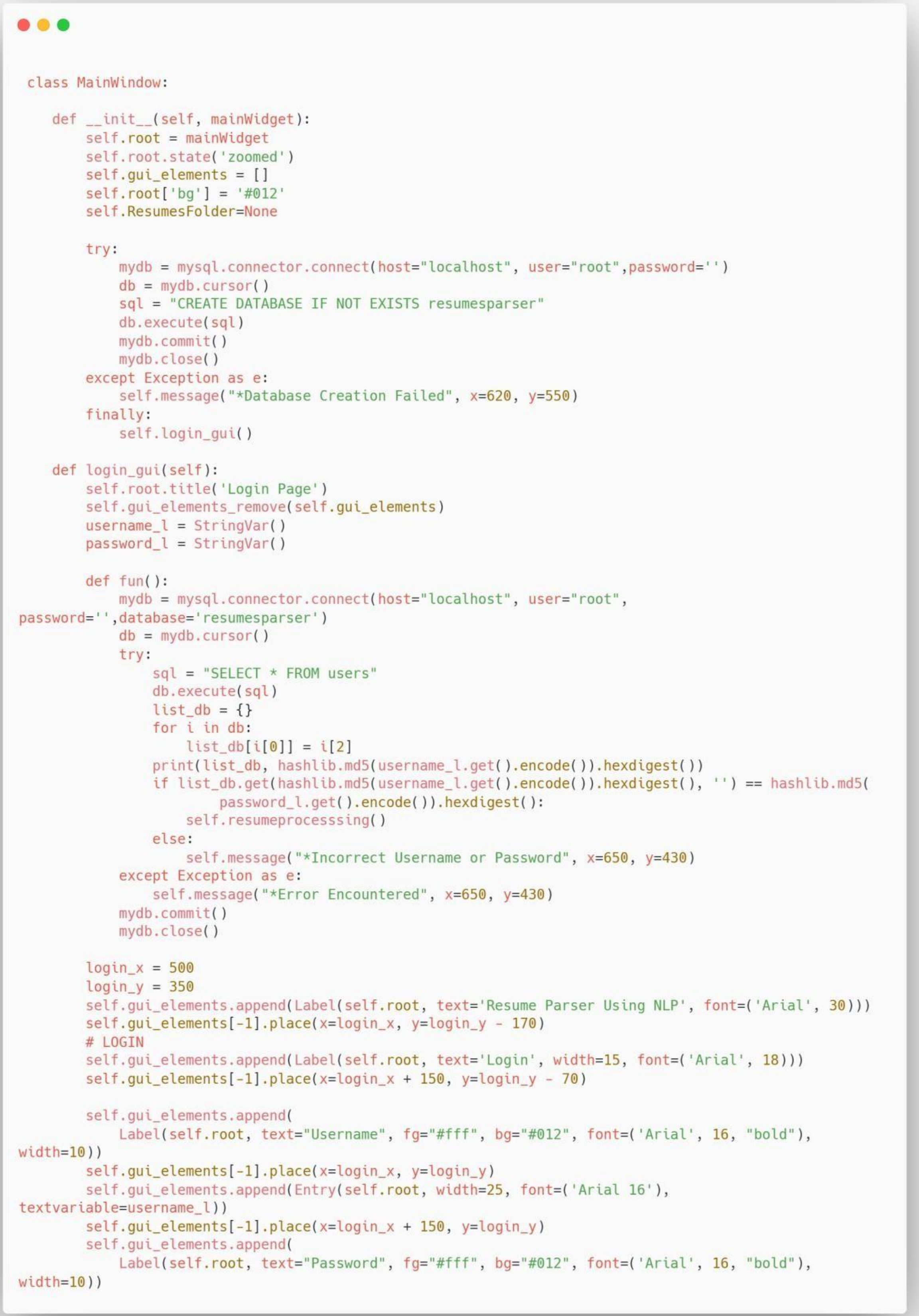
```
 # text = ''
    # for page in extract_text_from_pdf(r"C:\Users\pinna\Downloads\Profile.pdf"):
    #     text += ' ' + page
    text = ''
    for page in extract_text_from_pdf(m):
        text += ' ' + page
    return (text)

def extract_name(resume_text):
    nlp_text = nlp(resume_text)

    # First name and Last name are always Proper Nouns
    pattern = [{'POS': 'PROPN'}, {'POS': 'PROPN'}]

    matcher.add(key='NAME', patterns=[pattern])

    matches = matcher(nlp_text)

    for match_id, start, end in matches:
        span = nlp_text[start:end]
        return span.text

def extract_education(resume_text):
    nlp_text = nlp(resume_text)

    # Sentence Tokenizer
    nlp_text = [sent.text.strip() for sent in nlp_text.sents]

    edu = {}
    # Extract education degree
    for index, text in enumerate(nlp_text):
        for tex in text.split():
            # Replace all special symbols
            tex = re.sub(r'[?|$|.|!|,|)|(]', r'', tex)
            if index + 1 < len(nlp_text) and tex.upper() in EDUCATION and tex not in
STOPWORDS:      edu[tex] = text + nlp_text[index + 1]

    # Extract year
    education = []
    for key in edu.keys():
        year = re.search(re.compile(r'(((20|19)(\d{})))'), edu[key])
        if year:
            education.append((key, ''.join(year[0])))
        else:
            education.append(key)
    return education

def extract_branch(resume_text):
    nlp_text = nlp(resume_text)

    # Sentence Tokenizer
    nlp_text = [sent.text.strip() for sent in nlp_text.sents]

    edu = {}
    # Extract education degree
    for index, text in enumerate(nlp_text):
        for tex in text.split():
            # Replace all special symbols
            tex = re.sub(r'[?|$|.|!|,|)|(]', r'', tex)
            if index + 1 < len(nlp_text) and tex.upper() in BRANCH and tex not in STOPWORDS:
                edu[tex] = text + nlp_text[index + 1]
```



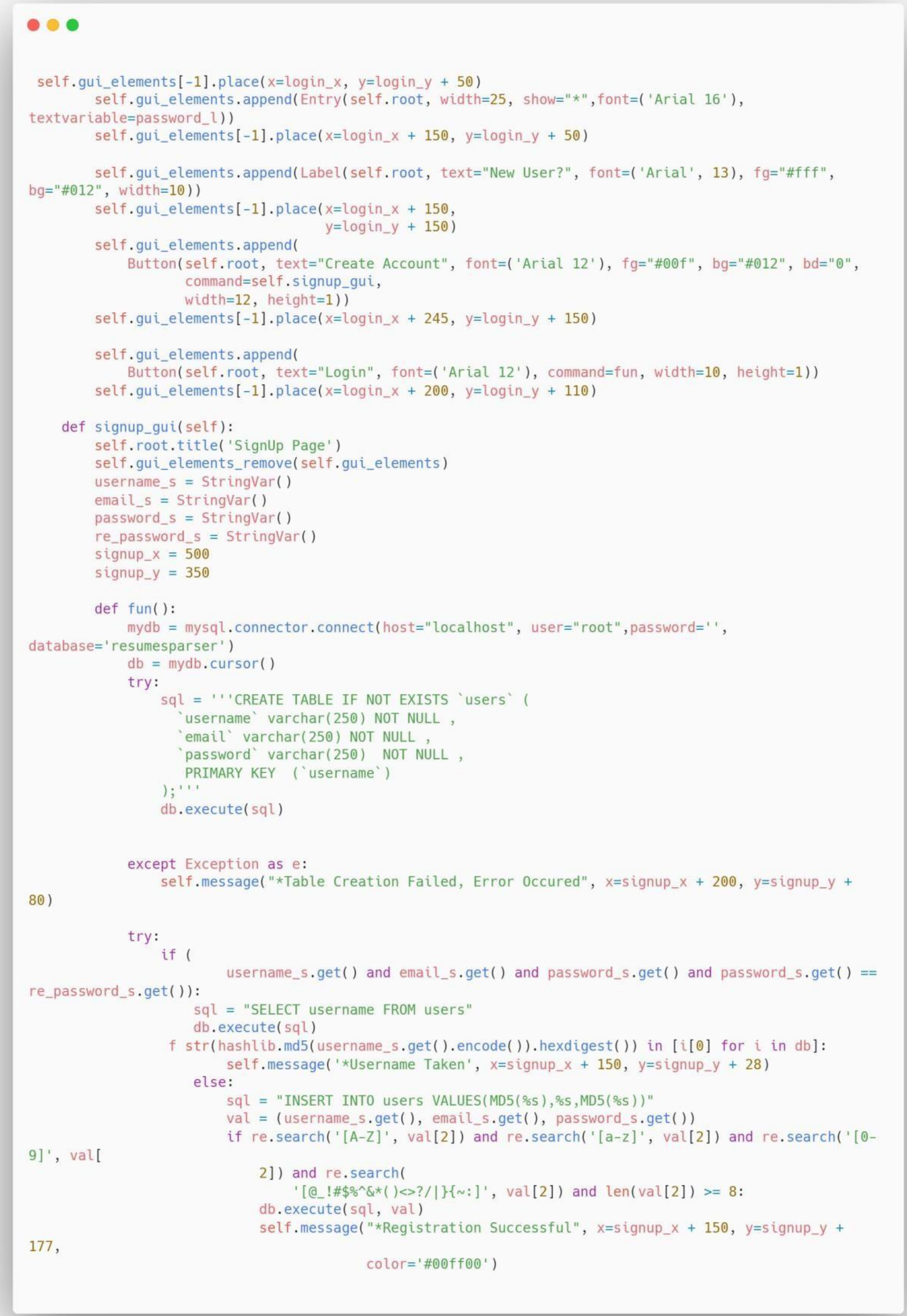
```python
    # Extract year
    education = []
    for key in edu.keys():
        year = re.search(re.compile(r'(((20|19)(\d{})))'), edu[key])
        if year:
            education.append((key, ''.join(year[0])))
        else:
            education.append(key)
    return education

# noun_chunks = nlp.noun_chunks
noun_chunks = None

def extract_skills(resume_text):
    global noun_chunks, skill_set
    nlp_text = nlp(resume_text)
    noun_chunks = nlp_text.noun_chunks
    # removing stop words and implementing word tokenization
    tokens = [token.text for token in nlp_text if not token.is_stop]
    # reading the csv file
    data = pd.read_csv(skill_set)

    # extract values
    skills = data['skill'].tolist()
    skillset = []

    # check for one-grams (example: python)
    for token in tokens:
        if token.lower() in skills:
            skillset.append(token)

    for token in noun_chunks:
        token = token.text.lower().strip()
        if token in skills:
            skillset.append(token)
    return [i.capitalize() for i in set([i.lower() for i in skillset])]


def extract_mobile_number(resume_text):
    phone = re.findall(re.compile(r'[\+\(]?[1-9][0-9 .\-\(\)]{8,}[0-9]'), resume_text)
    if phone:
        number = ''.join(phone[0])
        if len(number) > 10:
            return number
        else:
            return number


def extract_email_addresses(string):
    r = re.compile(r'[\w\.-]+@[\w\.-]+')
    return r.findall(string)[0].replace(".com", ".co").replace(".co", ".com") if r.findall(string) else
''


def extract_linkedin_addresses(string):
    r = re.compile(r'(?:http[s]?:\/\/)?(?:www\.)?linkedin\.com\/[a-z]{2}\/[a-zA-Z0-9_-]{3,100}\/?')
    return r.findall(string)[0] if r.findall(string) else ''


def extract_github_addresses(string):
    r = re.compile('(?:http[s]?:\/\/)?(?:www\.)?github\.com\/[a-zA-Z0-9_-]{3,100}\/?')
    return r.findall(string)[0] if r.findall(string) else ''
```



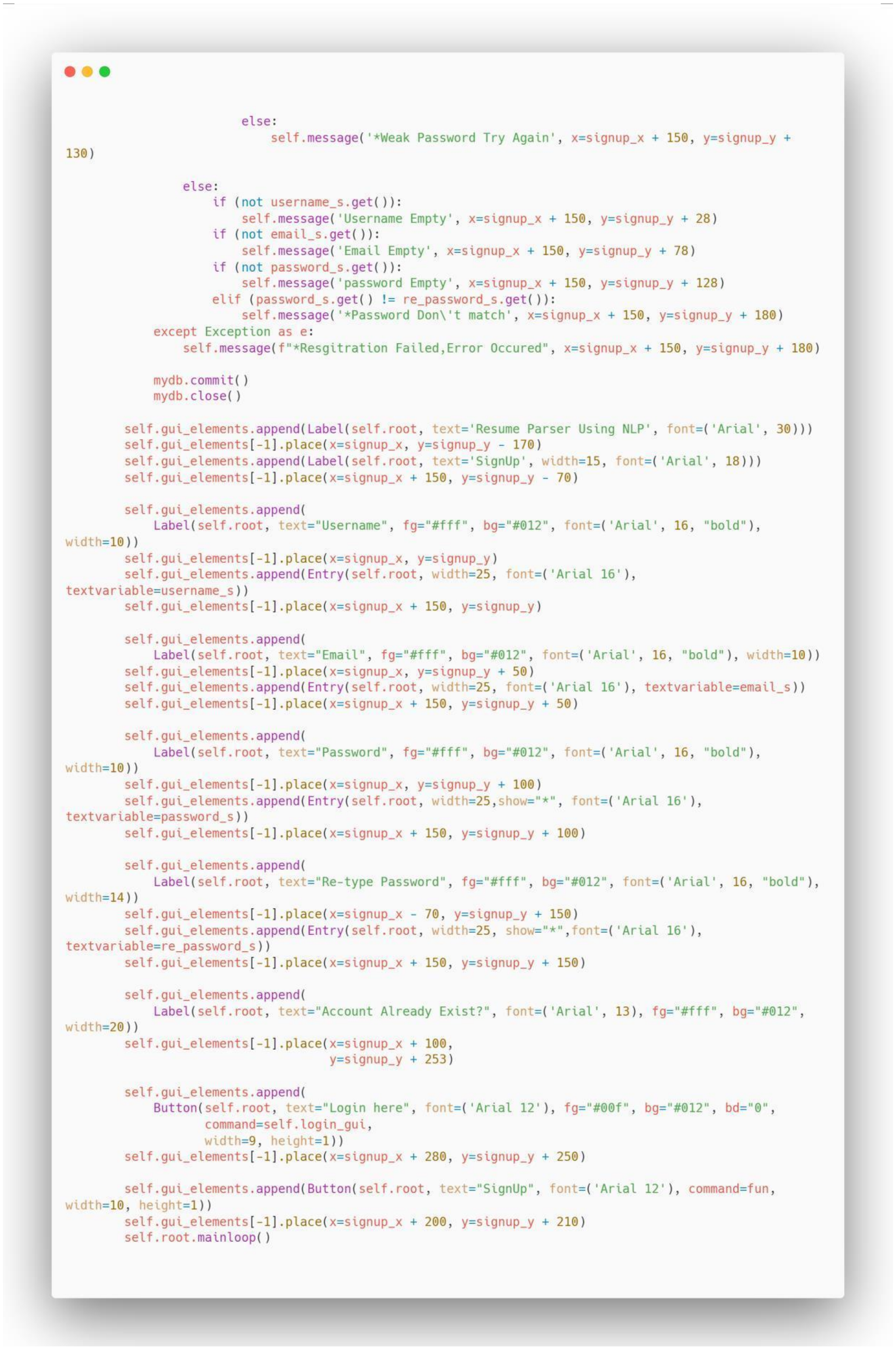
```
  def save_as_docx(path):
    # Opening MS Word
    word = win32.Dispatch("SAPI.SpVoice")
    doc = word.Documents.Open(path)
    doc.Activate()

    # Rename path with .docx
    new_file_abs = os.path.abspath(path)
    new_file_abs = re.sub(r'\.\w+$', '.docx', new_file_abs)

    # Save and Close
    word.ActiveDocument.SaveAs(
        new_file_abs, FileFormat=constants.wdFormatXMLDocument
    )
    doc.Close(False)


def sendMails(email_sender='pinnintiuday.3344@gmail.com', email_pass="fppdnxddjocusnfb",
              email_recevier=['pinnantiuday@gmail.com'], skill_set="python"):
    subject = "[Testing]You have been Shortlisted"
    body = f'''
    Hi, congratulations your resume has been shortlisted for {skill_set} role.

    This is a Testing message for Resume parsing for HR analytics project done by Uday from BE-CSE-SIST
    Kindly Ignore This Mail...
    '''
    em = EmailMessage()
    em['From'] = email_sender
    em['To'] = email_recevier
    em['Subject'] = subject
    em.set_content(body)
    context = ssl.create_default_context()
    with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp:
        smtp.login(email_sender, email_pass)
        for mailadd in email_recevier:
            if (re.match(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,7}\b', mailadd)):
                smtp.sendmail(email_sender, mailadd, em.as_string())
            else:
                print('Error sending mail to :', mailadd)
    print("Mails Successfully Sent")


def data_display(textinput):
    out_data = {'Name': extract_name(textinput),
                'Qualification': extract_education(textinput),
                'Specialization': extract_branch(textinput),
                'Skills': extract_skills(textinput),
                'Mobile Number': extract_mobile_number(textinput),
                'Email': extract_email_addresses(textinput),
                'LinkedIn': extract_linkedin_addresses(textinput),
                'Github': extract_github_addresses(textinput)
                }
    json_object = json.dumps(out_data, indent=4)

    with open(
            f"{os.getcwd()}/output/{re.sub('[^A-Za-z]', '', out_data['Name'])}_{skill_set.split('/')
[-1].replace('.csv','')}_result.json",
            "w") as outfile:
        outfile.write(json_object)
        outfile.close()
    print(json_object)
    df = pd.read_csv(skill_set)
    _score = 0
    _df = {i: j for i, j in zip(df['skill'], df['Score'])}
    for skill in out_data['Skills']:
        _score += _df[skill.lower()]
    return _score, out_data['Email']
```



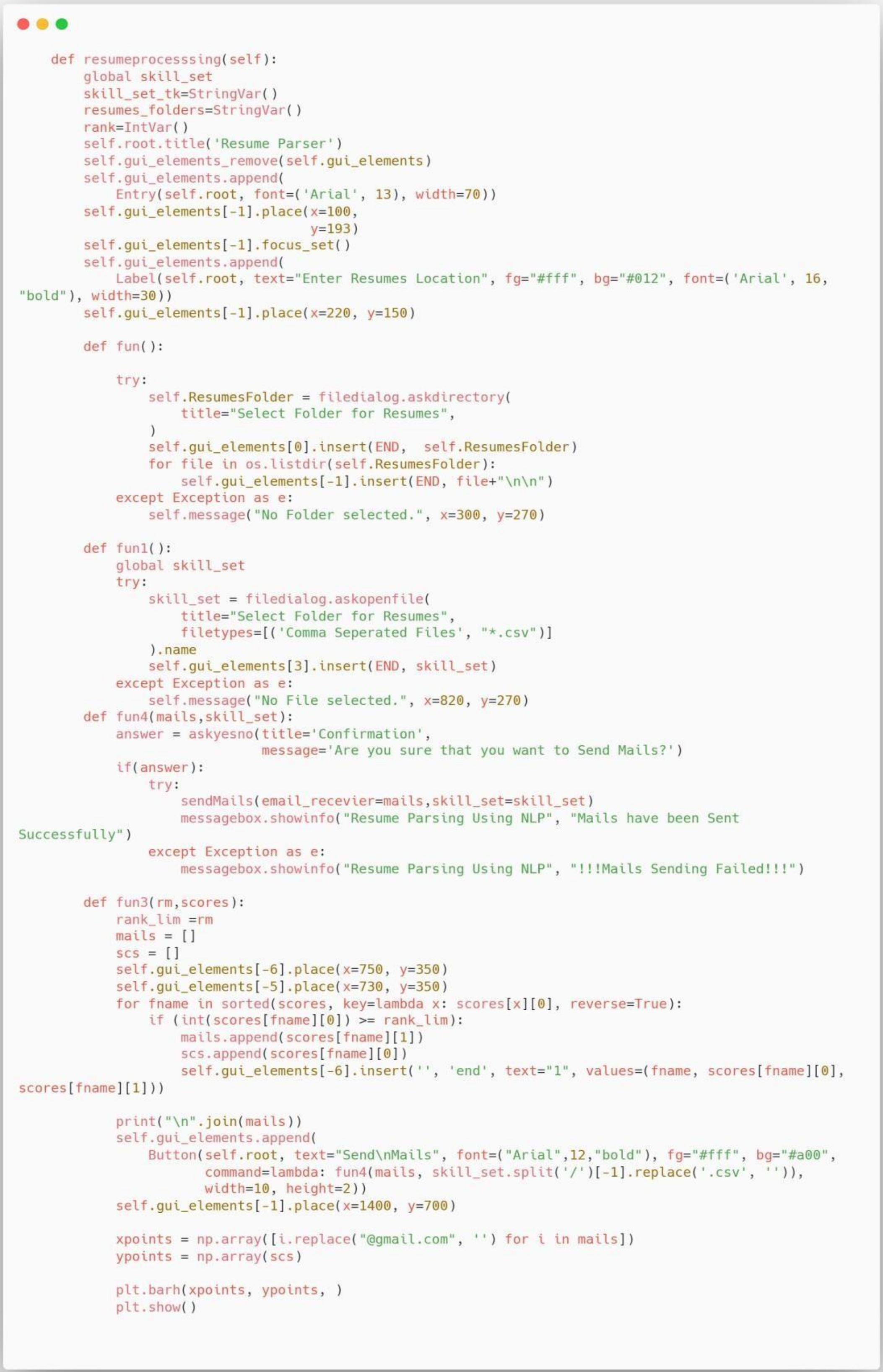
```
class MainWindow:

    def __init__(self, mainWidget):
        self.root = mainWidget
        self.root.state('zoomed')
        self.gui_elements = []
        self.root['bg'] = '#012'
        self.ResumesFolder=None

        try:
            mydb = mysql.connector.connect(host="localhost", user="root",password='')
            db = mydb.cursor()
            sql = "CREATE DATABASE IF NOT EXISTS resumesparser"
            db.execute(sql)
            mydb.commit()
            mydb.close()
        except Exception as e:
            self.message("*Database Creation Failed", x=620, y=550)
        finally:
            self.login_gui()

    def login_gui(self):
        self.root.title('Login Page')
        self.gui_elements_remove(self.gui_elements)
        username_l = StringVar()
        password_l = StringVar()

        def fun():
            mydb = mysql.connector.connect(host="localhost", user="root",
password='',database='resumesparser')
            db = mydb.cursor()
            try:
                sql = "SELECT * FROM users"
                db.execute(sql)
                list_db = {}
                for i in db:
                    list_db[i[0]] = i[2]
                print(list_db, hashlib.md5(username_l.get().encode()).hexdigest())
                if list_db.get(hashlib.md5(username_l.get().encode()).hexdigest(), '') == hashlib.md5(
                        password_l.get().encode()).hexdigest():
                    self.resumeprocesssing()
                else:
                    self.message("*Incorrect Username or Password", x=650, y=430)
            except Exception as e:
                self.message("*Error Encountered", x=650, y=430)
            mydb.commit()
            mydb.close()

        login_x = 500
        login_y = 350
        self.gui_elements.append(Label(self.root, text='Resume Parser Using NLP', font=('Arial', 30)))
        self.gui_elements[-1].place(x=login_x, y=login_y - 170)
        # LOGIN
        self.gui_elements.append(Label(self.root, text='Login', width=15, font=('Arial', 18)))
        self.gui_elements[-1].place(x=login_x + 150, y=login_y - 70)

        self.gui_elements.append(
            Label(self.root, text="Username", fg="#fff", bg="#012", font=('Arial', 16, "bold"),
width=10))
        self.gui_elements[-1].place(x=login_x, y=login_y)
        self.gui_elements.append(Entry(self.root, width=25, font=('Arial 16'),
textvariable=username_l))
        self.gui_elements[-1].place(x=login_x + 150, y=login_y)
        self.gui_elements.append(
            Label(self.root, text="Password", fg="#fff", bg="#012", font=('Arial', 16, "bold"),
width=10))
```



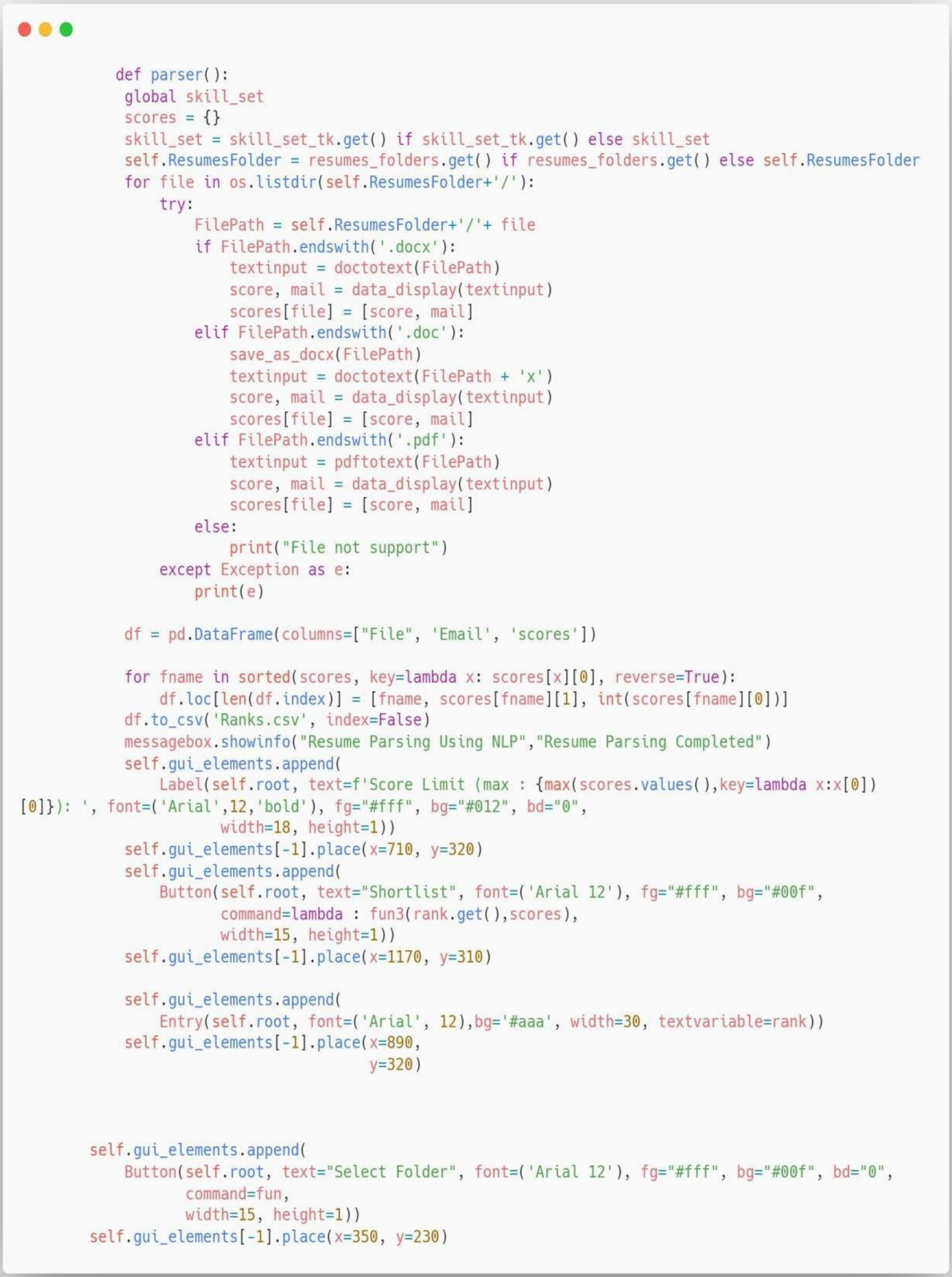
```
 self.gui_elements[-1].place(x=login_x, y=login_y + 50)
        self.gui_elements.append(Entry(self.root, width=25, show="*",font=('Arial 16'),
textvariable=password_l))
        self.gui_elements[-1].place(x=login_x + 150, y=login_y + 50)

        self.gui_elements.append(Label(self.root, text="New User?", font=('Arial', 13), fg="#fff",
bg="#012", width=10))
        self.gui_elements[-1].place(x=login_x + 150,
                                    y=login_y + 150)
        self.gui_elements.append(
            Button(self.root, text="Create Account", font=('Arial 12'), fg="#00f", bg="#012", bd="0",
                   command=self.signup_gui,
                   width=12, height=1))
        self.gui_elements[-1].place(x=login_x + 245, y=login_y + 150)

        self.gui_elements.append(
            Button(self.root, text="Login", font=('Arial 12'), command=fun, width=10, height=1))
        self.gui_elements[-1].place(x=login_x + 200, y=login_y + 110)

    def signup_gui(self):
        self.root.title('SignUp Page')
        self.gui_elements_remove(self.gui_elements)
        username_s = StringVar()
        email_s = StringVar()
        password_s = StringVar()
        re_password_s = StringVar()
        signup_x = 500
        signup_y = 350

        def fun():
            mydb = mysql.connector.connect(host="localhost", user="root",password='',
database='resumesparser')
            db = mydb.cursor()
            try:
                sql = '''CREATE TABLE IF NOT EXISTS `users` (
                  `username` varchar(250) NOT NULL ,
                  `email` varchar(250) NOT NULL  ,
                  `password` varchar(250)  NOT NULL ,
                   PRIMARY KEY  (`username`)
                );'''
                db.execute(sql)


            except Exception as e:
                self.message("*Table Creation Failed, Error Occured", x=signup_x + 200, y=signup_y +
80)

            try:
                if (
                        username_s.get() and email_s.get() and password_s.get() and password_s.get() ==
re_password_s.get()):
                    sql = "SELECT username FROM users"
                    db.execute(sql)
                 f str(hashlib.md5(username_s.get().encode()).hexdigest()) in [i[0] for i in db]:
                        self.message('*Username Taken', x=signup_x + 150, y=signup_y + 28)
                    else:
                        sql = "INSERT INTO users VALUES(MD5(%s),%s,MD5(%s))"
                        val = (username_s.get(), email_s.get(), password_s.get())
                        if re.search('[A-Z]', val[2]) and re.search('[a-z]', val[2]) and re.search('[0-
9]', val[
                            2]) and re.search(
                                '[@_!#$%^&*()<>?/|}{~:]', val[2]) and len(val[2]) >= 8:
                            db.execute(sql, val)
                            self.message("*Registration Successful", x=signup_x + 150, y=signup_y +
177,
                                         color='#00ff00')
```



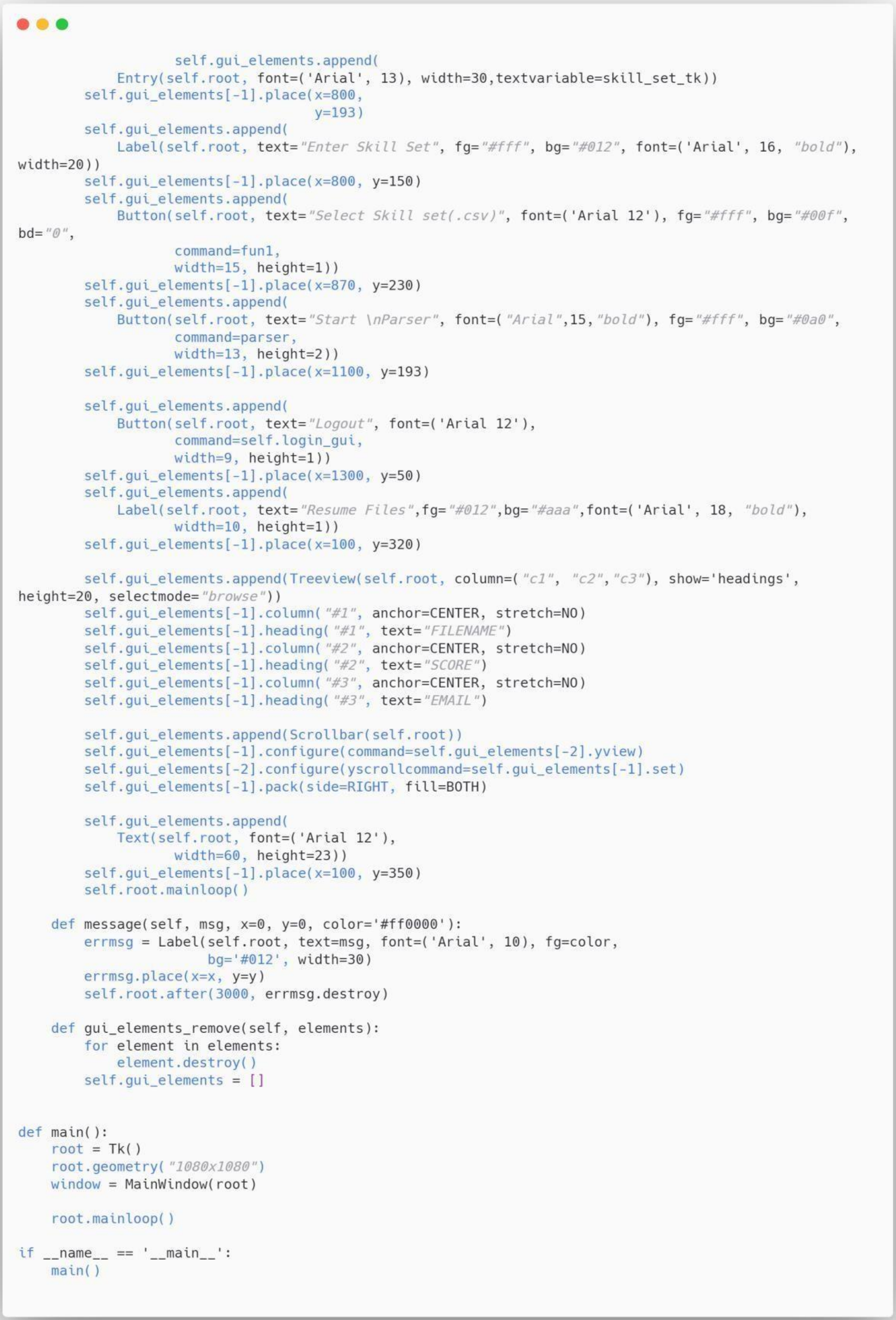
```
                        else:
                            self.message('*Weak Password Try Again', x=signup_x + 150, y=signup_y +
130)


                else:
                    if (not username_s.get()):
                        self.message('Username Empty', x=signup_x + 150, y=signup_y + 28)
                    if (not email_s.get()):
                        self.message('Email Empty', x=signup_x + 150, y=signup_y + 78)
                    if (not password_s.get()):
                        self.message('password Empty', x=signup_x + 150, y=signup_y + 128)
                    elif (password_s.get() != re_password_s.get()):
                        self.message('*Password Don\'t match', x=signup_x + 150, y=signup_y + 180)
            except Exception as e:
                self.message(f"*Resgitration Failed,Error Occured", x=signup_x + 150, y=signup_y + 180)

            mydb.commit()
            mydb.close()

        self.gui_elements.append(Label(self.root, text='Resume Parser Using NLP', font=('Arial', 30)))
        self.gui_elements[-1].place(x=signup_x, y=signup_y - 170)
        self.gui_elements.append(Label(self.root, text='SignUp', width=15, font=('Arial', 18)))
        self.gui_elements[-1].place(x=signup_x + 150, y=signup_y - 70)

        self.gui_elements.append(
            Label(self.root, text="Username", fg="#fff", bg="#012", font=('Arial', 16, "bold"),
width=10))
        self.gui_elements[-1].place(x=signup_x, y=signup_y)
        self.gui_elements.append(Entry(self.root, width=25, font=('Arial 16'),
textvariable=username_s))
        self.gui_elements[-1].place(x=signup_x + 150, y=signup_y)

        self.gui_elements.append(
            Label(self.root, text="Email", fg="#fff", bg="#012", font=('Arial', 16, "bold"), width=10))
        self.gui_elements[-1].place(x=signup_x, y=signup_y + 50)
        self.gui_elements.append(Entry(self.root, width=25, font=('Arial 16'), textvariable=email_s))
        self.gui_elements[-1].place(x=signup_x + 150, y=signup_y + 50)

        self.gui_elements.append(
            Label(self.root, text="Password", fg="#fff", bg="#012", font=('Arial', 16, "bold"),
width=10))
        self.gui_elements[-1].place(x=signup_x, y=signup_y + 100)
        self.gui_elements.append(Entry(self.root, width=25,show="*", font=('Arial 16'),
textvariable=password_s))
        self.gui_elements[-1].place(x=signup_x + 150, y=signup_y + 100)

        self.gui_elements.append(
            Label(self.root, text="Re-type Password", fg="#fff", bg="#012", font=('Arial', 16, "bold"),
width=14))
        self.gui_elements[-1].place(x=signup_x - 70, y=signup_y + 150)
        self.gui_elements.append(Entry(self.root, width=25, show="*",font=('Arial 16'),
textvariable=re_password_s))
        self.gui_elements[-1].place(x=signup_x + 150, y=signup_y + 150)

        self.gui_elements.append(
            Label(self.root, text="Account Already Exist?", font=('Arial', 13), fg="#fff", bg="#012",
width=20))
        self.gui_elements[-1].place(x=signup_x + 100,
                                    y=signup_y + 253)

        self.gui_elements.append(
            Button(self.root, text="Login here", font=('Arial 12'), fg="#00f", bg="#012", bd="0",
                   command=self.login_gui,
                   width=9, height=1))
        self.gui_elements[-1].place(x=signup_x + 280, y=signup_y + 250)

        self.gui_elements.append(Button(self.root, text="SignUp", font=('Arial 12'), command=fun,
width=10, height=1))
        self.gui_elements[-1].place(x=signup_x + 200, y=signup_y + 210)
        self.root.mainloop()
```



```
    def resumeprocesssing(self):
        global skill_set
        skill_set_tk=StringVar()
        resumes_folders=StringVar()
        rank=IntVar()
        self.root.title('Resume Parser')
        self.gui_elements_remove(self.gui_elements)
        self.gui_elements.append(
            Entry(self.root, font=('Arial', 13), width=70))
        self.gui_elements[-1].place(x=100,
                                    y=193)
        self.gui_elements[-1].focus_set()
        self.gui_elements.append(
            Label(self.root, text="Enter Resumes Location", fg="#fff", bg="#012", font=('Arial', 16,
"bold"), width=30))
        self.gui_elements[-1].place(x=220, y=150)

        def fun():

            try:
                self.ResumesFolder = filedialog.askdirectory(
                    title="Select Folder for Resumes",
                )
                self.gui_elements[0].insert(END,  self.ResumesFolder)
                for file in os.listdir(self.ResumesFolder):
                    self.gui_elements[-1].insert(END, file+"\n\n")
            except Exception as e:
                self.message("No Folder selected.", x=300, y=270)

        def fun1():
            global skill_set
            try:
                skill_set = filedialog.askopenfile(
                    title="Select Folder for Resumes",
                    filetypes=[('Comma Seperated Files', "*.csv")]
                ).name
                self.gui_elements[3].insert(END, skill_set)
            except Exception as e:
                self.message("No File selected.", x=820, y=270)
        def fun4(mails,skill_set):
            answer = askyesno(title='Confirmation',
                              message='Are you sure that you want to Send Mails?')
            if(answer):
                try:
                    sendMails(email_recevier=mails,skill_set=skill_set)
                    messagebox.showinfo("Resume Parsing Using NLP", "Mails have been Sent
Successfully")
                except Exception as e:
                    messagebox.showinfo("Resume Parsing Using NLP", "!!!Mails Sending Failed!!!")

        def fun3(rm,scores):
            rank_lim =rm
            mails = []
            scs = []
            self.gui_elements[-6].place(x=750, y=350)
            self.gui_elements[-5].place(x=730, y=350)
            for fname in sorted(scores, key=lambda x: scores[x][0], reverse=True):
                if (int(scores[fname][0]) >= rank_lim):
                    mails.append(scores[fname][1])
                    scs.append(scores[fname][0])
                    self.gui_elements[-6].insert('', 'end', text="1", values=(fname, scores[fname][0],
scores[fname][1]))

            print("\n".join(mails))
            self.gui_elements.append(
                Button(self.root, text="Send\nMails", font=("Arial",12,"bold"), fg="#fff", bg="#a00",
                       command=lambda: fun4(mails, skill_set.split('/')[-1].replace('.csv', '')),
                       width=10, height=2))
            self.gui_elements[-1].place(x=1400, y=700)

            xpoints = np.array([i.replace("@gmail.com", '') for i in mails])
            ypoints = np.array(scs)

            plt.barh(xpoints, ypoints, )
            plt.show()
```



```
    def parser():
     global skill_set
     scores = {}
     skill_set = skill_set_tk.get() if skill_set_tk.get() else skill_set
     self.ResumesFolder = resumes_folders.get() if resumes_folders.get() else self.ResumesFolder
     for file in os.listdir(self.ResumesFolder+'/'):
         try:
             FilePath = self.ResumesFolder+'/'+ file
             if FilePath.endswith('.docx'):
                 textinput = doctotext(FilePath)
                 score, mail = data_display(textinput)
                 scores[file] = [score, mail]
             elif FilePath.endswith('.doc'):
                 save_as_docx(FilePath)
                 textinput = doctotext(FilePath + 'x')
                 score, mail = data_display(textinput)
                 scores[file] = [score, mail]
             elif FilePath.endswith('.pdf'):
                 textinput = pdftotext(FilePath)
                 score, mail = data_display(textinput)
                 scores[file] = [score, mail]
             else:
                 print("File not support")
         except Exception as e:
             print(e)

     df = pd.DataFrame(columns=["File", 'Email', 'scores'])

     for fname in sorted(scores, key=lambda x: scores[x][0], reverse=True):
         df.loc[len(df.index)] = [fname, scores[fname][1], int(scores[fname][0])]
     df.to_csv('Ranks.csv', index=False)
     messagebox.showinfo("Resume Parsing Using NLP","Resume Parsing Completed")
     self.gui_elements.append(
         Label(self.root, text=f'Score Limit (max : {max(scores.values(),key=lambda x:x[0])
[0]}): ', font=('Arial',12,'bold'), fg="#fff", bg="#012", bd="0",
                width=18, height=1))
     self.gui_elements[-1].place(x=710, y=320)
     self.gui_elements.append(
         Button(self.root, text="Shortlist", font=('Arial 12'), fg="#fff", bg="#00f",
                command=lambda : fun3(rank.get(),scores),
                width=15, height=1))
     self.gui_elements[-1].place(x=1170, y=310)

     self.gui_elements.append(
         Entry(self.root, font=('Arial', 12),bg='#aaa', width=30, textvariable=rank))
     self.gui_elements[-1].place(x=890,
                                 y=320)



 self.gui_elements.append(
     Button(self.root, text="Select Folder", font=('Arial 12'), fg="#fff", bg="#00f", bd="0",
            command=fun,
            width=15, height=1))
 self.gui_elements[-1].place(x=350, y=230)
```



```
                   self.gui_elements.append(
            Entry(self.root, font=('Arial', 13), width=30,textvariable=skill_set_tk))
        self.gui_elements[-1].place(x=800,
                                    y=193)
        self.gui_elements.append(
            Label(self.root, text="Enter Skill Set", fg="#fff", bg="#012", font=('Arial', 16, "bold"),
width=20))
        self.gui_elements[-1].place(x=800, y=150)
        self.gui_elements.append(
            Button(self.root, text="Select Skill set(.csv)", font=('Arial 12'), fg="#fff", bg="#00f",
bd="0",
                   command=fun1,
                   width=15, height=1))
        self.gui_elements[-1].place(x=870, y=230)
        self.gui_elements.append(
            Button(self.root, text="Start \nParser", font=( "Arial",15,"bold"), fg="#fff", bg="#0a0",
                   command=parser,
                   width=13, height=2))
        self.gui_elements[-1].place(x=1100, y=193)

        self.gui_elements.append(
            Button(self.root, text="Logout", font=('Arial 12'),
                   command=self.login_gui,
                   width=9, height=1))
        self.gui_elements[-1].place(x=1300, y=50)
        self.gui_elements.append(
            Label(self.root, text="Resume Files",fg="#012",bg="#aaa",font=('Arial', 18, "bold"),
                   width=10, height=1))
        self.gui_elements[-1].place(x=100, y=320)

        self.gui_elements.append(Treeview(self.root, column=( "c1", "c2","c3"), show='headings',
height=20, selectmode="browse"))
        self.gui_elements[-1].column( "#1", anchor=CENTER, stretch=NO)
        self.gui_elements[-1].heading( "#1", text="FILENAME")
        self.gui_elements[-1].column( "#2", anchor=CENTER, stretch=NO)
        self.gui_elements[-1].heading( "#2", text="SCORE")
        self.gui_elements[-1].column( "#3", anchor=CENTER, stretch=NO)
        self.gui_elements[-1].heading( "#3", text="EMAIL")

        self.gui_elements.append(Scrollbar(self.root))
        self.gui_elements[-1].configure(command=self.gui_elements[-2].yview)
        self.gui_elements[-2].configure(yscrollcommand=self.gui_elements[-1].set)
        self.gui_elements[-1].pack(side=RIGHT, fill=BOTH)

        self.gui_elements.append(
            Text(self.root, font=('Arial 12'),
                   width=60, height=23))
        self.gui_elements[-1].place(x=100, y=350)
        self.root.mainloop()

    def message(self, msg, x=0, y=0, color='#ff0000'):
        errmsg = Label(self.root, text=msg, font=('Arial', 10), fg=color,
                       bg='#012', width=30)
        errmsg.place(x=x, y=y)
        self.root.after(3000, errmsg.destroy)

    def gui_elements_remove(self, elements):
        for element in elements:
            element.destroy()
        self.gui_elements = []


def main():
    root = Tk()
    root.geometry("1080x1080")
    window = MainWindow(root)

    root.mainloop()

if __name__ == '__main__':
    main()
```

B. SCREENSHOTS

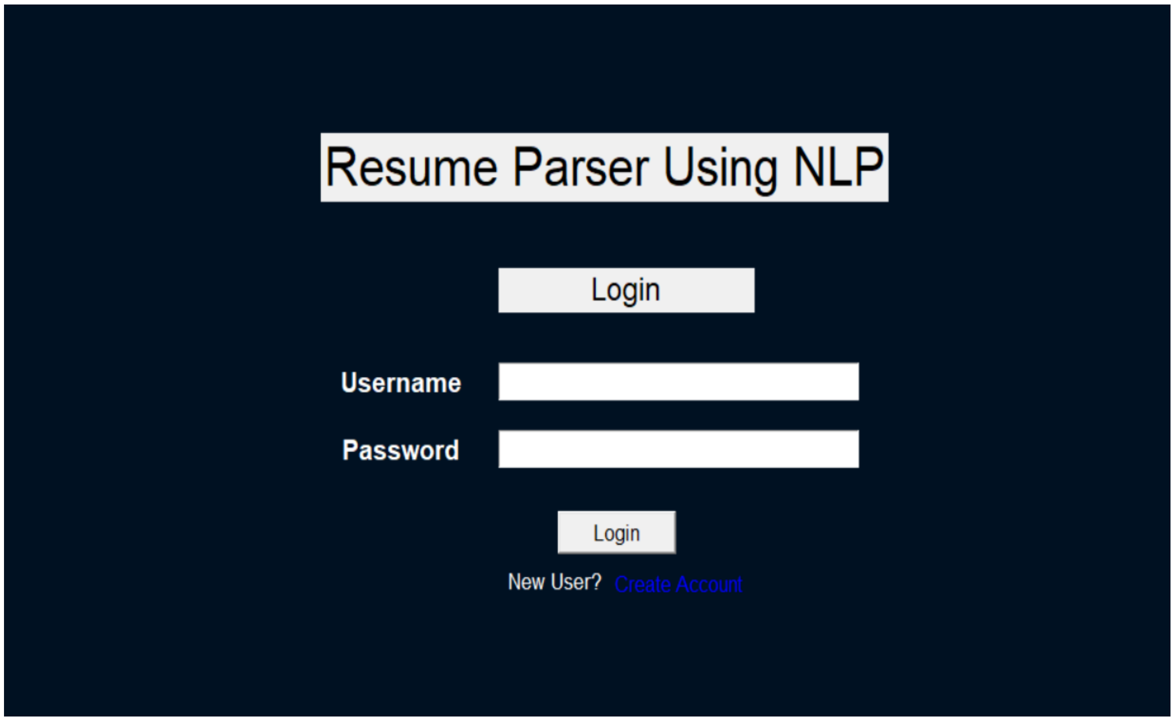


Figure 14: Login Page

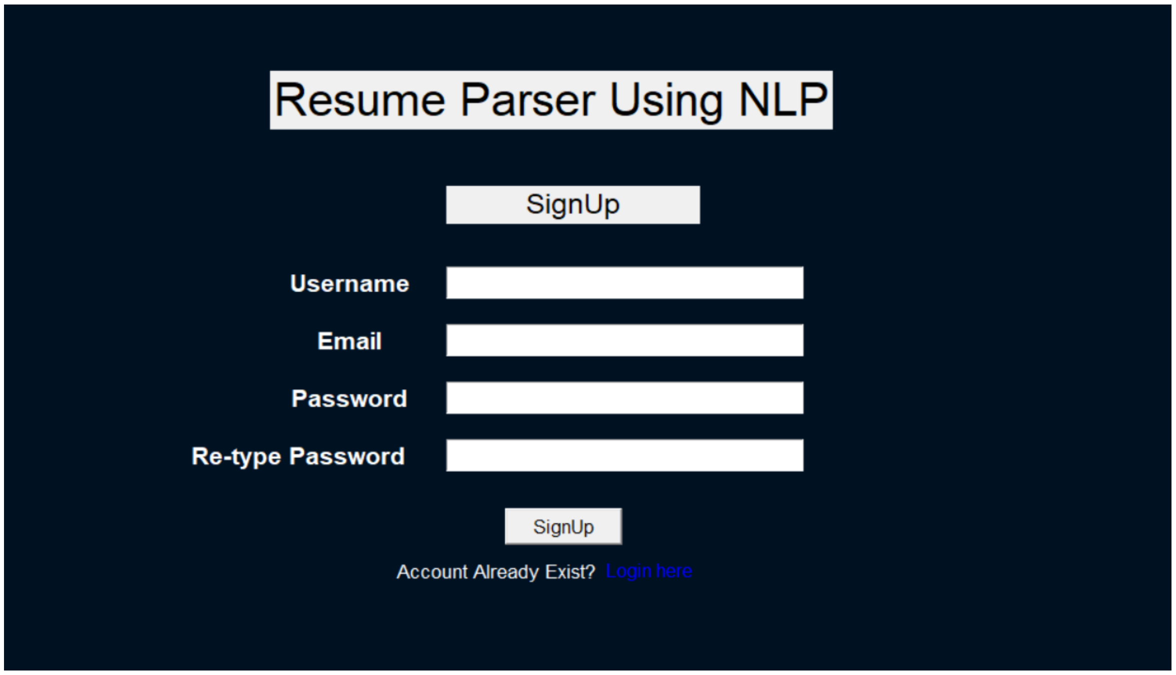


Figure 15: Registration Page

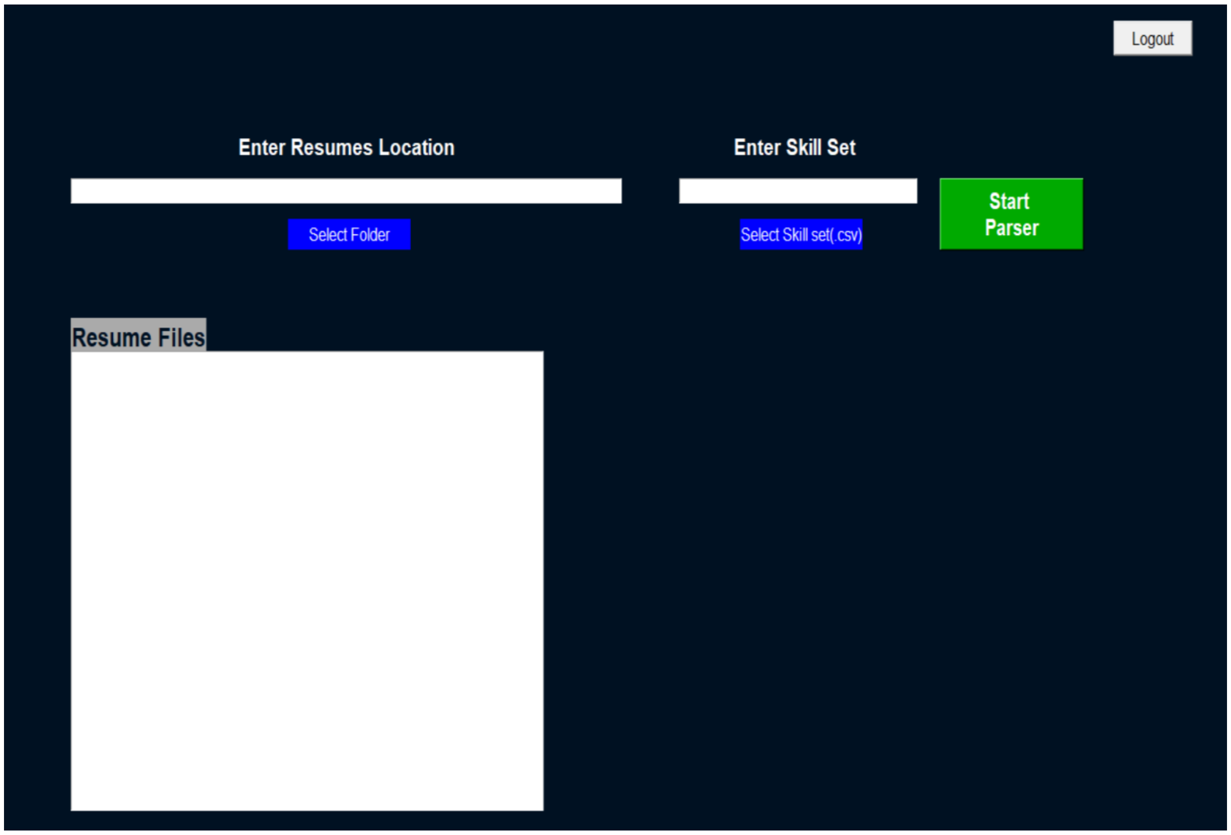
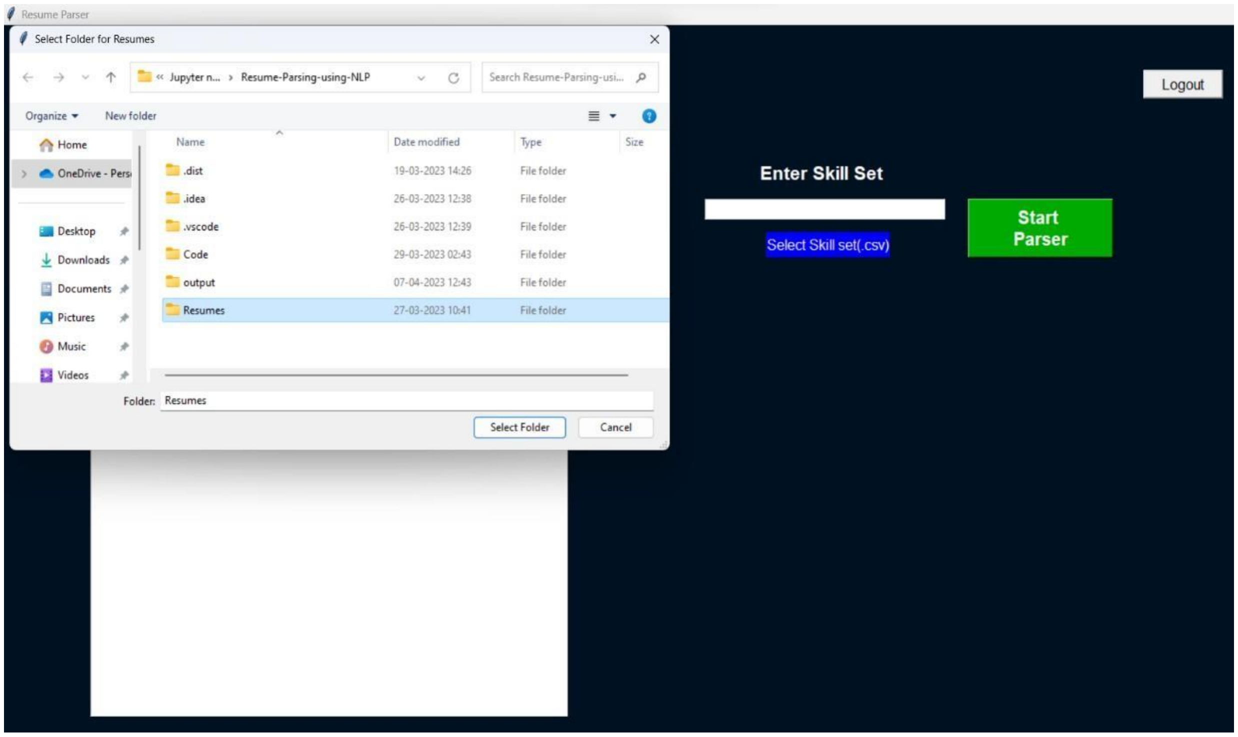


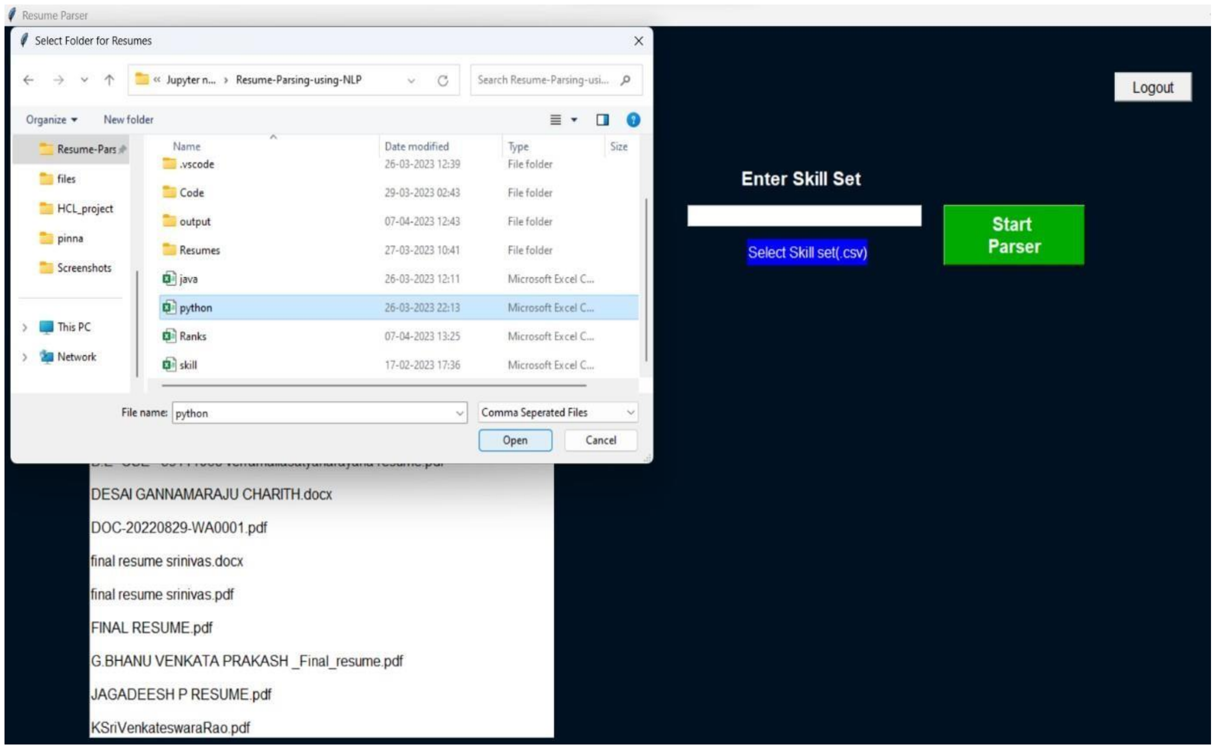
Figure 16: File Upload Page



Name	Date modified	Type	Size
.dist	19-03-2023 14:26	File folder	
.idea	26-03-2023 12:38	File folder	
.vscode	26-03-2023 12:39	File folder	
Code	29-03-2023 02:43	File folder	
output	07-04-2023 12:43	File folder	
Resumes	27-03-2023 10:41	File folder	

Folder: Resumes

Figure 17: Insert Data Set Folder Page

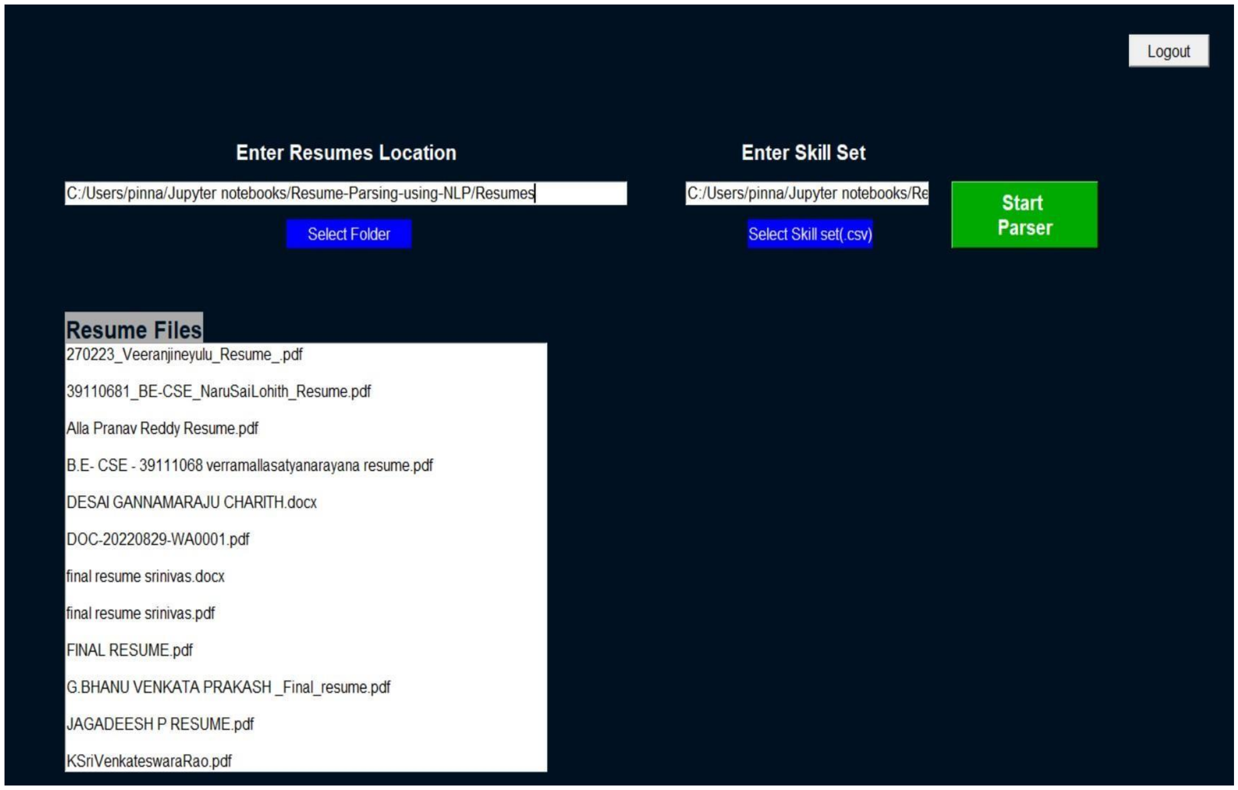


Name	Date modified	Type	Size
.vscode	26-03-2023 12:39	File folder	
Code	29-03-2023 02:43	File folder	
output	07-04-2023 12:43	File folder	
Resumes	27-03-2023 10:41	File folder	
java	26-03-2023 12:11	Microsoft Excel C...	
python	26-03-2023 22:13	Microsoft Excel C...	
Ranks	07-04-2023 13:25	Microsoft Excel C...	
skill	17-02-2023 17:36	Microsoft Excel C...	

File name: python
Comma Seperated Files

DESAI GANNAMARAJU CHARITH.docx
DOC-20220829-WA0001.pdf
final resume srinivas.docx
final resume srinivas.pdf
FINAL RESUME.pdf
G.BHANU VENKATA PRAKASH _Final_resume.pdf
JAGADEESH P RESUME.pdf
KSriVenkateswaraRao.pdf

Figure 18: Skill set Folder

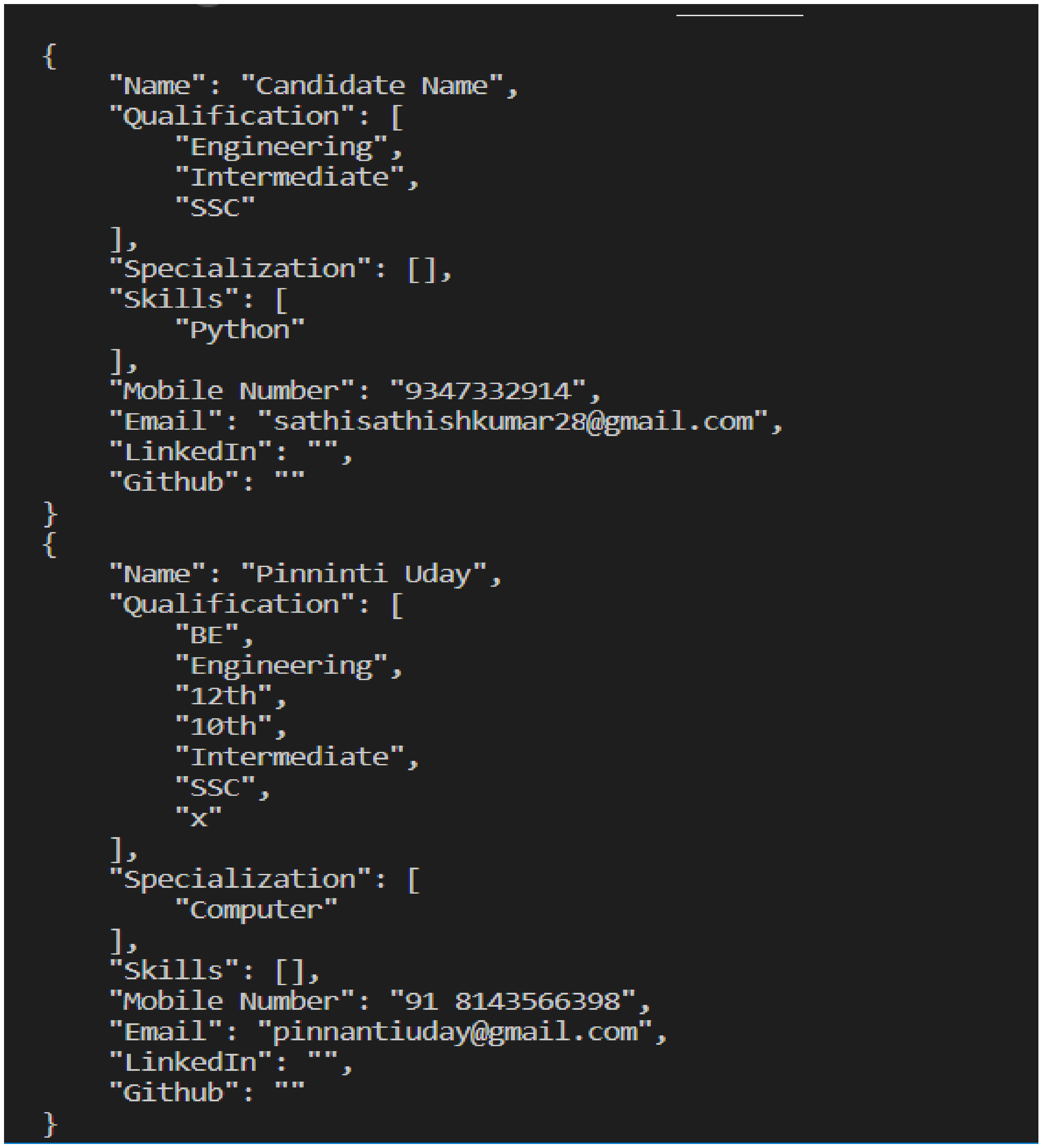


C:/Users/pinna/Jupyter notebooks/Resume-Parsing-using-NLP/Resumes

C:/Users/pinna/Jupyter notebooks/Re

Resume Files
270223_Veeranjineyulu_Resume_.pdf
39110681_BE-CSE_NaruSaiLohith_Resume.pdf
Alla Pranav Reddy Resume.pdf
B.E- CSE - 39111068 verramallasatyanarayana resume.pdf
DESAI GANNAMARAJU CHARITH.docx
DOC-20220829-WA0001.pdf
final resume srinivas.docx
final resume srinivas.pdf
FINAL RESUME.pdf
G.BHANU VENKATA PRAKASH _Final_resume.pdf
JAGADEESH P RESUME.pdf
KSriVenkateswaraRao.pdf

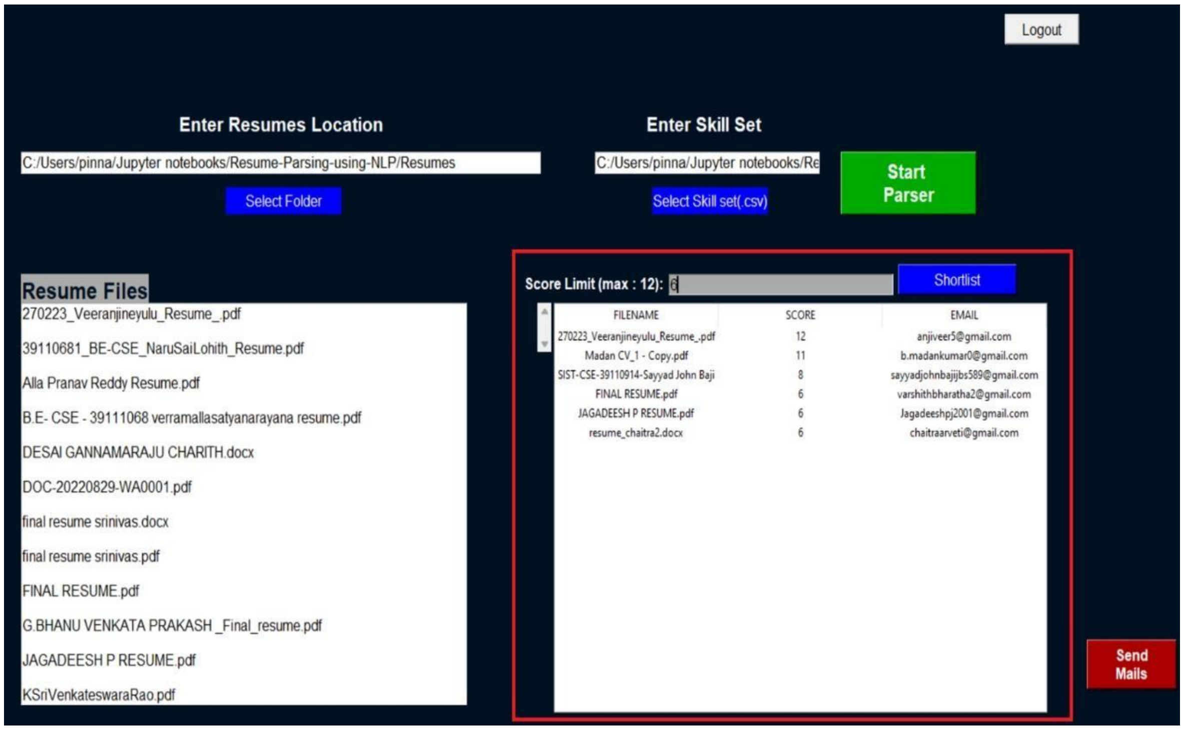
Figure 19: Start Parsing



```
{
    "Name": "Candidate Name",
    "Qualification": [
        "Engineering",
        "Intermediate",
        "SSC"
    ],
    "Specialization": [],
    "Skills": [
        "Python"
    ],
    "Mobile Number": "9347332914",
    "Email": "sathisathishkumar28@gmail.com",
    "LinkedIn": "",
    "Github": ""
}
{
    "Name": "Pinninti Uday",
    "Qualification": [
        "BE",
        "Engineering",
        "12th",
        "10th",
        "Intermediate",
        "SSC",
        "x"
    ],
    "Specialization": [
        "Computer"
    ],
    "Skills": [],
    "Mobile Number": "91 8143566398",
    "Email": "pinnantiuday@gmail.com",
    "LinkedIn": "",
    "Github": ""
}
```

Figure 20: Entities Extracted from Resume

. json format is used to display the retrieved data. This allows the HR staff to quickly identify candidates that meet the organization's needs. This technique also benefits the HR department by saving time by eliminating the need to manually review each resume.



C:/Users/pinna/Jupyter notebooks/Resume-Parsing-using-NLP/Resumes

C:/Users/pinna/Jupyter notebooks/Re

Resume Files

270223_Veeranjineyulu_Resume_.pdf
39110681_BE-CSE_NaruSaiLohith_Resume.pdf
Alla Pranav Reddy Resume.pdf
B.E- CSE - 39111068 verramallasatyanarayana resume.pdf
DESAI GANNAMARAJU CHARITH.docx
DOC-20220829-WA0001.pdf
final resume srinivas.docx
final resume srinivas.pdf
FINAL RESUME.pdf
G.BHANU VENKATA PRAKASH _Final_resume.pdf
JAGADEESH P RESUME.pdf
KSriVenkateswaraRao.pdf

Score Limit (max : 12): 6

FILENAME	SCORE	EMAIL
270223_Veeranjineyulu_Resume_.pdf	12	anjiveer5@gmail.com
Madan CV_1 - Copy.pdf	11	b.madankumar0@gmail.com
SIST-CSE-39110914-Sayyad John Baji	8	sayyadjohnbajijbs589@gmail.com
FINAL RESUME.pdf	6	varshithbharatha2@gmail.com
JAGADEESH P RESUME.pdf	6	Jagadeeshpj2001@gmail.com
resume_chaitra2.docx	6	chaitraarveti@gmail.com

Figure 21: Shotlisted Resumes

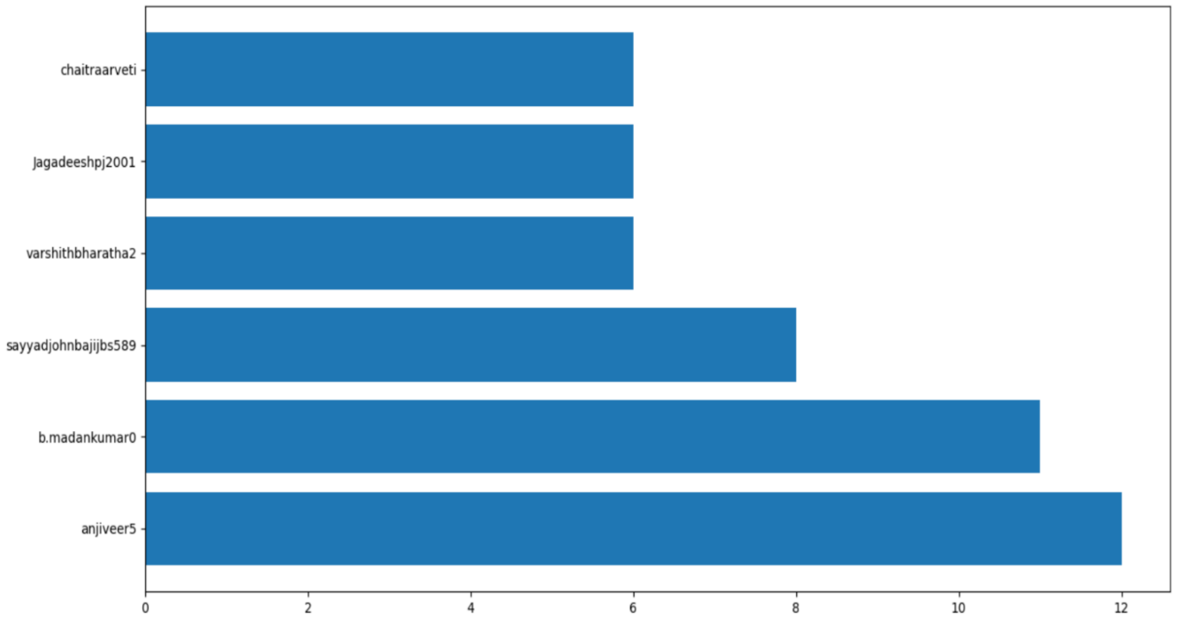
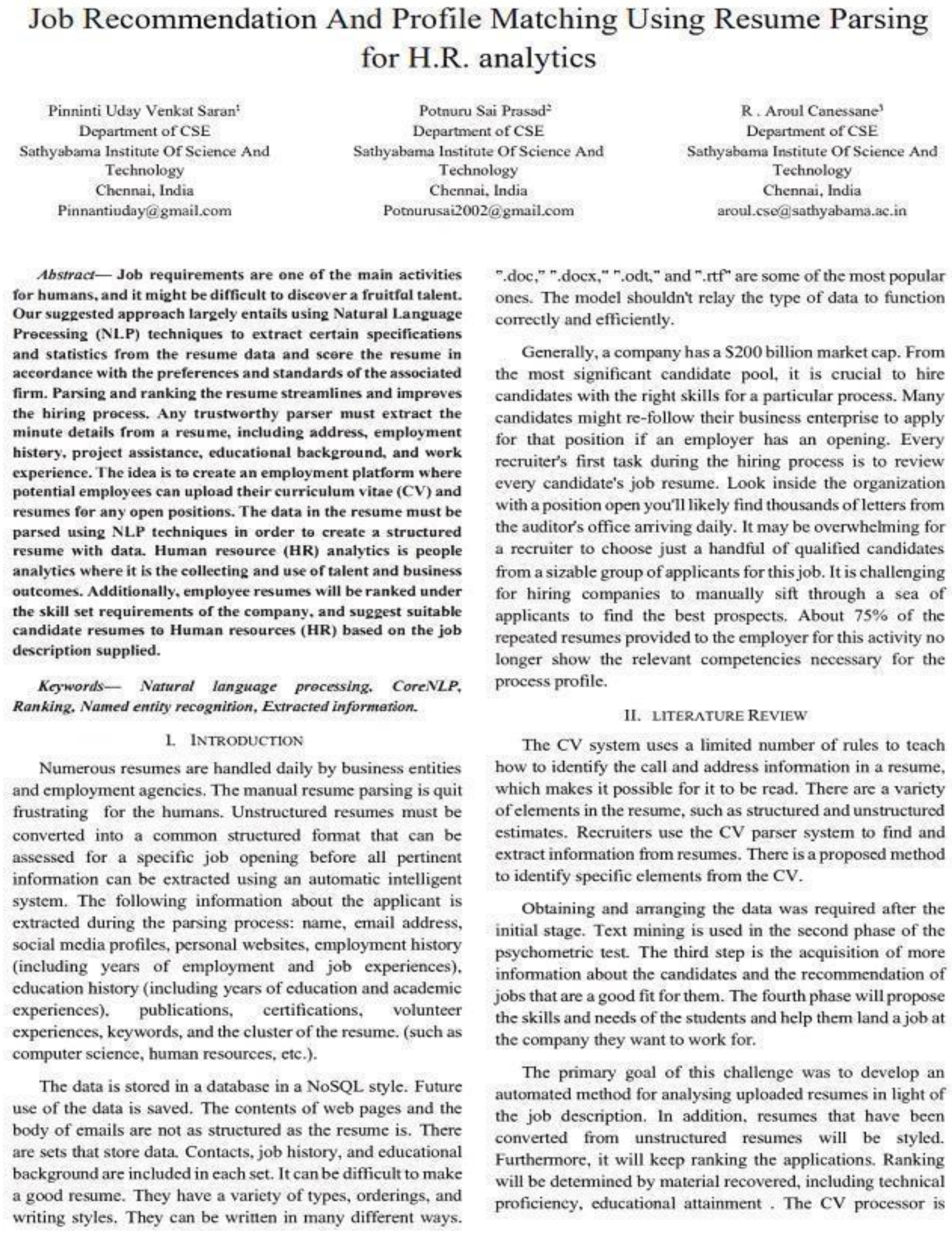


Figure 22: Bar Graph

C. RESEARCH PAPER



Job Recommendation And Profile Matching Using Resume Parsing for H.R. analytics

Pinninti Uday Venkat Saran[1]
Department of CSE
Sathyabama Institute Of Science And Technology
Chennai, India
Pinnantiuday@gmail.com

Potnuru Sai Prasad[2]
Department of CSE
Sathyabama Institute Of Science And Technology
Chennai, India
Potnurusai2002@gmail.com

R . Aroul Canessane[3]
Department of CSE
Sathyabama Institute Of Science And Technology
Chennai, India
aroul.cse@sathyabama.ac.in

Abstract— **Job requirements are one of the main activities for humans, and it might be difficult to discover a fruitful talent. Our suggested approach largely entails using Natural Language Processing (NLP) techniques to extract certain specifications and statistics from the resume data and score the resume in accordance with the preferences and standards of the associated firm. Parsing and ranking the resume streamlines and improves the hiring process. Any trustworthy parser must extract the minute details from a resume, including address, employment history, project assistance, educational background, and work experience. The idea is to create an employment platform where potential employees can upload their curriculum vitae (CV) and resumes for any open positions. The data in the resume must be parsed using NLP techniques in order to create a structured resume with data. Human resource (HR) analytics is people analytics where it is the collecting and use of talent and business outcomes. Additionally, employee resumes will be ranked under the skill set requirements of the company, and suggest suitable candidate resumes to Human resources (HR) based on the job description supplied.**

Keywords— Natural language processing, CoreNLP, Ranking, Named entity recognition, Extracted information.

I. INTRODUCTION

Numerous resumes are handled daily by business entities and employment agencies. The manual resume parsing is quit frustrating for the humans. Unstructured resumes must be converted into a common structured format that can be assessed for a specific job opening before all pertinent information can be extracted using an automatic intelligent system. The following information about the applicant is extracted during the parsing process: name, email address, social media profiles, personal websites, employment history (including years of employment and job experiences), education history (including years of education and academic experiences), publications, certifications, volunteer experiences, keywords, and the cluster of the resume. (such as computer science, human resources, etc.).

The data is stored in a database in a NoSQL style. Future use of the data is saved. The contents of web pages and the body of emails are not as structured as the resume is. There are sets that store data. Contacts, job history, and educational background are included in each set. It can be difficult to make a good resume. They have a variety of types, orderings, and writing styles. They can be written in many different ways. ".doc," ".docx," ".odt," and ".rtf" are some of the most popular ones. The model shouldn't relay the type of data to function correctly and efficiently.

Generally, a company has a $200 billion market cap. From the most significant candidate pool, it is crucial to hire candidates with the right skills for a particular process. Many candidates might re-follow their business enterprise to apply for that position if an employer has an opening. Every recruiter's first task during the hiring process is to review every candidate's job resume. Look inside the organization with a position open you'll likely find thousands of letters from the auditor's office arriving daily. It may be overwhelming for a recruiter to choose just a handful of qualified candidates from a sizable group of applicants for this job. It is challenging for hiring companies to manually sift through a sea of applicants to find the best prospects. About 75% of the repeated resumes provided to the employer for this activity no longer show the relevant competencies necessary for the process profile.

II. LITERATURE REVIEW

The CV system uses a limited number of rules to teach how to identify the call and address information in a resume, which makes it possible for it to be read. There are a variety of elements in the resume, such as structured and unstructured estimates. Recruiters use the CV parser system to find and extract information from resumes. There is a proposed method to identify specific elements from the CV.

Obtaining and arranging the data was required after the initial stage. Text mining is used in the second phase of the psychometric test. The third step is the acquisition of more information about the candidates and the recommendation of jobs that are a good fit for them. The fourth phase will propose the skills and needs of the students and help them land a job at the company they want to work for.

The primary goal of this challenge was to develop an automated method for analysing uploaded resumes in light of the job description. In addition, resumes that have been converted from unstructured resumes will be styled. Furthermore, it will keep ranking the applications. Ranking will be determined by material recovered, including technical proficiency, educational attainment . The CV processor is utilised here. The process of collecting CVs is called CV extraction.

The semantic talent mapping, multilingual support, recruiter support, and customization capabilities of the CV parser are only a few of its characteristics. As a consequence of its integration, customers are given an API pass. The parser functions by adhering to a number of principles that define the name and location. Because resumes come in a variety of forms and data kinds, including organised and unstructured data, metadata, and other information, employers who engage recruiters often choose resumes that have been parsed using a CV parser method. A foundation for extracting things from submitted CVs is provided by the recommended CV parser technology.

In support of this effort, a thorough evaluation of applications was done while taking into account different standards of excellence. A collection of resumes was also used along with a straightforward text analysis approach. A complete quality rating was produced from the scores that had been gathered after completing two qualitative covering processes on the comprehensibility of the resume gathering and the aspects. A resume quality measure was produced by reliably combining all the factors into a 1–5 grading system. The computational approach produced qualitative assessment results that were consistent with and backed up by the collective wisdom of numbers.

III. Proposed work

The recommended approach uses regular expression libraries to analyse the resume. Every organisation has a site that all companies can access, where they can also see the resumes of the applicants for the position that have been gathered. Every resume submitted via this website will be screened and examined in line with the unique criteria of the company. Additionally, the project aims to gather crucial data from social media wireless networks like the professional networking website for potential employment applications. After gathering outstanding candidates throughout many regions, they attempt to reduce unfair and prejudiced behaviours to streamline the recruiting procedure.

The first step in the process is called lexical evaluation, which involves dividing the text into words and paragraphs. Syntactic Analysis Sentences like "College goes to girl" are rejected because they don't follow the correct rules. One of the challenges with using the same NLP module in various businesses is that the jargon and terms that are exclusive to each company's industry may have a broader meaning. The use of named object recognition can get around this problem. A physical thing is called a named entity. It is possible to improve the comprehension of actual word objects using named entity recognition.

The employer or the company's human resources (HR) department will outline the skill set requirements for the job that they have posted. Additionally, the tokenized resumes of the applicant's skills are compared to the necessary skill set. The candidate resumes will be scored using built-in queries, and their results will be supplied as bar graphs and pie charts with complete statistics. In the end, the applicants will be sorted using the bar graph's percentage. A final list of candidates who have been shortlisted for the subsequent placement procedure will be created.

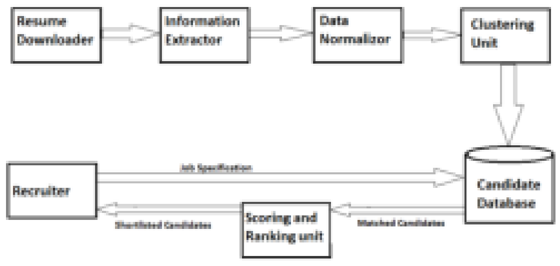


Fig.3.1. Resume Parsing

Text can be prepared before analyzing the data with the help of the NLP technology which works in the background. There are a lot of libraries used to address challenges. Regular expressions in Python can be used for text manipulation and scanning. NLTK and spaCy are used to carry out activities related to natural languages. The process of preprocessing data is difficult. Preprocessed data can be used to build a model. Any attempts start with the CPU. Lemmatization, lower casting, and stop-word elimination are afewofthepreparation techniques

A. *Tokenization*

The first section of the book examines tokenization. It makes it possible to recognize the text's main concepts. Tokens are the basic elements. Tokenization is useful because it divides the content into manageable chunks. Within the spaCy structure, the system naturally decides whether to consider a period (".") as a distinct token or an abbreviation, such as "B.A." Word tokenization or phrase tokenization can be used, depending on the issue.

a. Using the sentence tokenization, a paragraph is broken into several sentences function.

b. Word tokenization is the process of breaking a statement down into a list of terms method.

B. *Removing Stop words*

The data needs to be cleaned in order to remove noise. Stop words are the most frequently used words in literature. "I", "no", "nor", "me", "mine", "myself", "some", "such" are some of the stop words in the English language. The stop words are often removed during text processing to improve the accuracy of natural language processing algorithms.

The NLTK library eliminates about 180 and is well-liked for stop-word removal. To address specific challenges, it is possible to create a customized set of stop words. By utilizing this approach, new words can be easily added to the existing set of terms as needed.

C. *Lemmatization*

A word's inflected forms are compressed into a simpler, more basic form known as a lemma through the process of lemmatization, which preserves the syntax of the original word. Lemmatization makes use of a pre-existing lexicon to guarantee that the term is retained in the vocabulary and to maintain its meaning. For instance, the word "organized" is the lemma for the group of words collectively referred to as "organizing." A word's inflection can reveal grammatical information such as tense. (Organized vs. organized). To make it easier to analyze an inflected version of a word, lemmatization must reduce it to a single element. Additionally, it can help normalize and stop the use of terms with similar meanings repeatedly.

D. *Lower casting*

Computers consider upper and lower case letters as separate characters when text is written in various case. For instance, in the context of word processing, "cat" and "Cat" have distinct meanings. It is advised to change all of the letters to the same case, ideally lowercase, to avoid these problems. The "lower()" function in Python is frequently used to carry out this action on strings. The parameter less "lower()" method does not accept any input. It converts each uppercase character into a lowercase version in order to create lowercase strings from the input text. It returns the exact string if the supplied string contains no capital letters.

IV. Results and discussions

The resume is transformed from formats such as *.pdf, *.doc, and *.rtf into a set of plain text tokens that represent data entities. Comparisons between extracted entities and necessary keywords were used to score the resume, and the results then showed as a pie chart and bar graph.

Reading the resume comes first after importing the libraries. Resumes can be in any file type, including *.pdf, *.doc, and *.docx, as they don't have a specific file type. As a result, reading the resume and converting it to plain language is our top priority. For implementing the task, doc2text and pdf miner python modules is imported. These modules enable the extraction of text from various text document data formats. A function has been created to extract plain text from PDF files at this location. The pdf path argument specifies the location of the extracted PDF files. Setting up a resource manager instance will be the first step. An object is created that resembles a file using Python's io module. The creation of a converter is the next phase. For that, Text Converter is used.

A function is developed to pull raw content from the various resume sections. A list of resume sections is available in the "sections.py" file. Default separators include whitespace characters like '\s' and '\n'. The text argument provides the resume's original text. It returns a dictionary of entities. In this case, Another function is developed which is used to pull the email address out of the text. The text argument provides the resume files to extract plain text. The standard format for email addresses is a string of letters followed by a symbol There are two strings at the end. Regular expressions can be used to extract text.

A method has been created to get the name from the text. The method takes the text from the resume files. The module is called Spacy. It includes models that have been trained. The primary goal of entity recognition is to obtain names. Rule-based matching is one of the stages in the extraction of data from unstructured data. Based on patterns, it recognises and extracts words and symbols. To extract entities in rule-based matching, regular expressions can be employed. Proper nouns, such as first and last names are extracted. A set of objects known as a pattern defines a set of tokens that must match. There are two PROPN POS tags on it.

As a consequence, objects with PROPN POS tags are included in the pattern. The pattern is added to the matcher using the NAME and match id. Finally, matches are found by using their starting and ending indices.

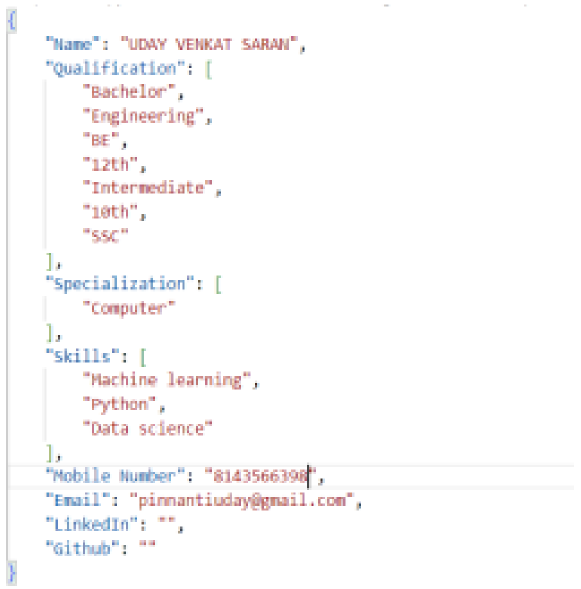
In this case, to pull the phone number out of the text a function is developed. The phone numbers will be retrieved using regular expressions. There are many different ways to format phone numbers, such as (+91) 1234567890, +911234567890, +91 123 456 7890, or +91 1234567890. I consequently developed a general regular expression that matches all identical phone number patterns.

Let's concentrate on the factors that matter most to recruiters, skills after acquiring basic information about the person. To perform that, a function has been developed that pulls talents out of the spaCy text. The NLP text parameter is the object of spaCy tokens. The noun chunks option is used to extract noun chunks from the Tokenisation and is a technique that can be used to extract key points. To execute tokenization, a basic dataset has been created to compare the tokenization. Compilation of the necessary skill sets to a comma-separated value (.csv) file has been created. For example, a human resources person needs someone who knows NLP, ML, and AI and a recruiter may create a CSV file containing the following keywords like Natural language processing, machine learning, and artificial intelligence and call it as talents. Following that, tokenizing the text which has been collected and compared the skills in resume. A CSV file is used to read using the panda's module. That has been eliminated all the stop words from the resume once the content is read. The list of talents that were extracted is finally returned.

The same method used to extract technical talents is also used to extract non-technical skills. The information gathered from the resume will be the degree and the year of graduation. For instance, tuple like ('B.S.,' '2022') will be extracted, if XYZ completed their BS in 2022. In this situation there is need of eliminating the stop words. A lengthy list of stop words is imported using the NLTK package, which exclude from our resume content. The level of education or degree required for a given position is something that human resources (HR) teams are quite clear about. As a result, the creation of an EDUCATION list containing all comparable degrees that satisfy the requirements.

After completing various pre-processing steps, the extracted text option provides the plain text retrieved from the data files. The text is printed out after a word search for the phrase "experience" in the chunk. The list of experiences is finally returned. This makes it possible for the human resources (HR) team to rapidly find individuals that are a good fit for the company. The human resources (HR) department also gains from this method because it can save time by not having to review each resume personally.

Fig 4.1 indicates the attributes that have been obtained after removing all the stop words using regular expressions. These entities help HR to analyze the skills and qualifications of the particular person in a more efficient manner. It will extract the Name, qualifications, specifications, skills, mobile number, email from the resume. These extracted entities will store in the form of a JSON file and it will extract email from the JSON file.



```
{
    "Name": "UDAY VENKAT SARAN",
    "Qualification": [
        "Bachelor",
        "Engineering",
        "BE",
        "12th",
        "Intermediate",
        "10th",
        "SSC"
    ],
    "Specialization": [
        "Computer"
    ],
    "Skills": [
        "Machine learning",
        "Python",
        "Data science"
    ],
    "Mobile Number": "8143566398",
    "Email": "pinnantiuday@gmail.com",
    "LinkedIn": "",
    "Github": ""
}
```

Fig.4.1.Entities Extracted From Resume

A Pie-chart is prepared by the system which indicates the percentage of every resume which had qualified for the requirements of HR after analyzing all the resumes as shown in figure 4.2.

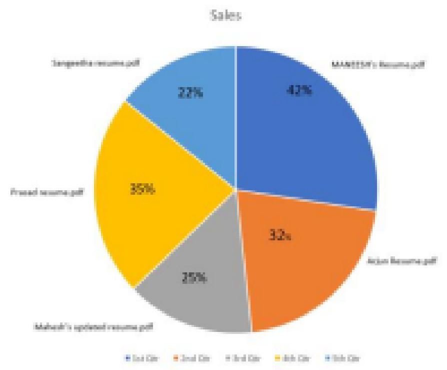


Fig.4.2.Pie-Chart

A graph is prepared by the system which indicates resumes are ranked according to keywords like Skills of the candidate and the system will sort out the resumes depending upon the requirements of HR. For the candidates who got top ranking based on the skills mentioned on their resume, there will be an automated mail-generating system that will send an email to every candidate who has qualified for the requirements of HR. Stating that those candidates have successfully been shortlisted for further rounds.

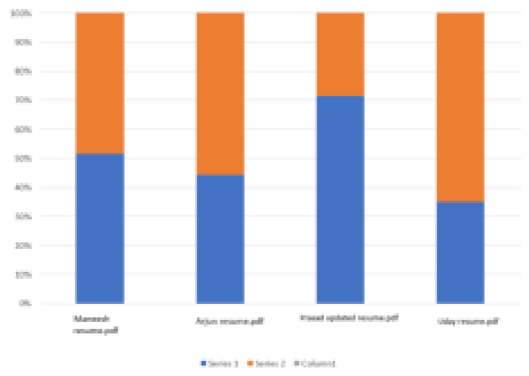


Fig.4.3.Resume ranked according to keywords

V. EXISTING SYSTEM

The Existing system for resume parsing is to parse all the resumes according to the HR requirements.By using NLP techniques and Regular expressions the resume will be extracted.After entity extraction from resume,Email will be genearted automatically to the selected candidates who's resume is shortlisted based upon their resume rank for the required firm.

VI. CONCLUSION

The Organization gets a lot of applicants for each job posting. It is difficult to locate the proper candidate's application from a sea of resumes. It takes a lot of time and resources to categorization a candidates resume. An automated resume parser is developed to get the necessary information from the resume of a candidate. The candidate's resume will be suggested by the system to HR. The email was sent to the candidates based on their resume rank.

REFERENCES

[1] Smart Innovation, Systems and Technologies book series (SIST, number 72), Abeer Zaroor, Mohammed Maree, Muath Sabha, 2017.

[2] "User-Based Efficient Video Recommendation System," Lecture Notes on Data Engineering and Communications Technologies, Vol. 26, pp. 1307–1316, 2019, by Albert Mayan J, Aroul Canesaane R, Jabez J, Kamalesh M.D, and Rama Mohan Reddy G.

[3] Intelligent Recruitment System Using NLP, Anushka Sharma, Smiti Singhal, and Dhara Ajudia, International Conference on Artificial Intelligence and Machine Vision (AIMV), 2021.

[4] Abdul Samad Ebrahim Yahya, Fahad, and SK Ahmed. "Deep learning inflectional review on natural language processing." Smart Computing and Electronic Enterprise Conference 2018 (ICSEE). IEEE, 2018.

[5] Ahmed SK and Samad Abdul Fahad, Ebrahim Yahya, and. Inflectional analysis of deep learning for processing natural English. 2018 Conference on Smart Enterprise and Computing (ICSEE). IEEE, 2018.

[6] Jitendra Purohit, Aditya Bagwe, Rishabh Mehta, Ojaswini Mangaonkar, and Elizabeth George, "Natural Language Processing based Jaro-The Interviewing Chatbot," 3rd International Conference on Computing Methodologies and Communication (ICCMC), August 2019.

[7] "Blood donation and life saver app," 2017 2nd International Conference on Communication and Electronics Systems (ICCES), pp. 446–451, by M. R. Annish Brislin, J. Albert Mayan, R. Aroul Canessane, and M. R. Anish Hamlin.

[8] "Predicting the Risk of Diabetes Mellitus to Subpopulations Using Association Rule Mining," in Advances in Intelligent Systems and Computing, issue 397, Springer, 2016, pp. 59–69.

[9] Nihar R. Mahapatra and Dena F. Mujtaba. Ethical Aspects of AI-Based Hiring. International Symposium on Technology and Society, IEEE 2019, 2019 (ISTAS). IEEE, 2019.

[10] V. V. Nguyen, V. L. Pham, and N. S. Vu, 2018. Study of Resume Information Extraction. semantic expert.

[11] A CV Parser Model Using Big Data Tools and Entity Extraction Process 2018 IJITCS, Papiya Das, Manjusha Pandey, and Siddharth Swarup Rautaray.

[12] Procedia Computer Science, Volume 167, Pages 2318–2327, 2020, "A Machine Learning Approach for automation of Resume Recommendation system." Rocky Bhatia, Pradeep Kumar Roy, and Sarabjeet Singh Chowdhary.

[13] Rajath V, Riza Tanaz Fareed, and Sharadadevi Kaganurmath, "Resume Classification and Ranking Using KNN And Cosine Similarity," international JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY, volume 10, issue 08, AUGUST 2021.

[14] 'Design and Development of Machine Learning based Resume Ranking System', Global Transitions Proceedings, October 2021. Sanjay Revanna, Shashank M. Kadiwal, Tejaswini K., and Umadevi V.

[15] "An Unstructured Text Analytics Approach for Qualitative Evaluation of Resumes," 2015, IJIRAE, by Vinaya R. Kudatarkar, Manjula Ramanavar, and Dr. Nandini S. Sidnal.

[16] "Resume Net: A Learning-Based Framework for Automatic Resume Quality Assessment," IEEE International Conference on Data Mining (ICDM), 2018, December. Huaizheng Zhang, Xinwen Zhang, Yong Luo, Yongjie Wang, and Yonggang Wen.