International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6945
***Article Received**: 30 March 2023 **Revised**: 19 May 2023 **Accepted**: 01 June 2023*

Break Down Resumes into Sections to Extract Data and Perform Text Analysis using Python

Arvind Kumar Sinha[1], Md. Amir Khusru Akhtar[2], Mohit Kumar[3]
[1]Faculty of Computing and IT
Usha Martin University
Ranchi, India
passionarvind@gmail.com
[2]Faculty of Computing and IT
Usha Martin University
Ranchi, India
akakhtar.2020@gmail.com
[3]Department of Information Technology
MIT Art, Design and Technology University
Pune, India
mohitsmailbox13@gmail.com

Abstract— The objective of AI-based resume screening is to automate the screening process, and text, keyword, and named entity recognition extraction are critical. This paper discusses segmenting resumes in order to extract data and perform text analysis. The raw CV file has been imported, and the resume data cleaned to remove extra spaces, punctuation and stop words. To extract names from resumes, regular expressions are used. We have also used the spaCy library which is considered the most accurate natural language processing library. It includes already-trained models for entity recognition, parsing, and tagging. The experimental method is used with resume data sourced from Kaggle, and external Source (MTIS).

Keywords- Resume Parser, Text Analysis, text classification, python, regular expressions, tagging, parsing.

I. INTRODUCTION

Resume/CV is extracted from the given document using natural language processing with complex patterns / language analysis techniques. It is a method of converting unstructured information from unknown resumes provided to a structured target. Typically, this process converts pdf, doc and docx type of files, as all companies accepts these types of files as resume, into structured data format. To understand human language and automate processes, AI and NLP technologies have been used. Semantic search is used by resume parsers to understand the resume data and screening criteria is applied to find the shortlisted candidate. The extraction of an indigenous language is a complex process because indigenous languages are extremely different as well as incomprehensible. Because text forms like resumes are written and interpreted by “n” number of ways, the analytical tool should be able to capture it using complex rule engine and mathematical algorithms. There are many instances where the same word would mean different things in different contexts, there is ambiguity that needs to be handled in the parsing tool. A few digit number in the resume/cv can be part of phone, address, email etc. As a result, the idea behind training a machine is to analyze the context of documents written as if by a person.

Hiring agencies use analytics system to streamline the process and cut down on hiring time for employers. The resume launcher automatically categorizes information based on the conditions found in CV/resume, which contains personal details like candidate name, address, email, phone, experience like companies worked, start-end date, designation, education details, hobbies and so on [1]. Resumes can be written in a variety of ways, making it difficult for online recruitment companies to store this information in relational databases. In this study, Kariyer.net (Turkey's largest online recruitment website) and TUBITAK (Turkey's Scientific and Technological Research Council) proposed a system that allows free structured resume formats to be transformed into an ontological structure model. [2]

The specific information from free text sources is extracted and the process is called as “information extraction”. Sentence segmentation or sentence boundary detection is used to divide textual information into sentences [3]. The rule-based AI method [4] for subdivision employs list of punctuation like '; ', '?', '.' and so on, but it doesn’t work when it finds contractions like 'e.g.', 'etc.', 'n.d.' and so on. The supervised machine learning approach was implemented to classify punctuation marks. It uses decision tree to build sentence boundaries to categorize punctuation [5].

The supervised AI and ML model builds large amounts of training and contraction knowledge [3]. Kiss and Strunk proposed a type-based classification approach for unsupervised machine learning. This method analyses a word throughout the text and annotates it with sentence boundary and abbreviation annotation [3].

Tokenization is the process by which the application splits a line of resume into tokens after segmenting the sentence boundaries [21-22]. Several tokenization approaches, including rule-based and statistical approaches, have been discussed in the paper. A rule-based tokenizer like the "Penn Tree Bank" tokenizer (PTB) [6], applies a set of rules to classify tokens. Hidden Markow Model (HMM)s [7] are used in the statistical approach to classify "word" and "sentence boundaries" [8]. Tokenization is accomplished through the use of scan and the "HMM Boundary" detector modules.

To determine the sense from a word, "part of speech" (POS) tagging techniques like "Penn Treebank Tagset" (PTT) [6] and the "CLAWS 5 (C5)" Tagset [9] to be used. The "Name Entity Recognition" (NER) is one of the very important task in information extraction because it understands names of entities such as candidate names, companies worked at etc. [10].

The structure of the remaining paper has various sections; section 2 discusses the methodology. Section 3 explains results and step wise implementation. Finally, section 4 concludes the paper.

II. METHODOLOGY

Text analysis is an AI and NLP learning technique that allows businesses to analyse unstructured text data like resume/CV analysis [11]. It is commonly used in the screening of resumes. Further, "Text classification" used for mapping pre-defined tags to random text. It is regarded as important process for dealing with natural language because it is versatile and can edit, edit, and parse any type of text to bring logical facts and issue resolution. NLP i.e., Natural Language Processing is a sub system of AI and Machine learning, where system can decipher as well as comprehend text in the same way that humans do. The sentiment analysis scans and classifies text for polarity (positive, negative, or neutral) and beyond using sophisticated machine learning algorithms. This method is concerned with the author's moods, emotions, context, and sarcasm. Topic analysis is also useful in text classification, which organises text by topic or subject. Intent classification is used to understand the intent rather than looking for literal meaning [10].

On the other hand, cluster analysis is a method of organising a collection to identify and classify based on the similarity within the group. It is a common mathematical data analysis process that is used in a variety of industries like pattern identification, bio-metric based recognition, text analysis etc. [12]. Text collections can interpret and collect large amounts of unrelated information. Compilation algorithms are faster to use than classification algorithms because we don't have to label models for training, despite being less accurate. Unsupervised learning machines are intelligent data for algorithms which helps in predicting results without training data.

At the basic, token making is used for dividing a string of letters into logical blocks which is analysed while removal of useless data. Building tokens is important for text data processing. Tokens for all words or tones in a sentence [13]. It separates the various objects in a sentence by the word border such as tokens. Basically, it is the same as the separation of words. Eg: Python is a Programing Language

"Python", "is", "a", "Programing", "Language"

It provides various tokens with varying values.

Now the question arises in mind that why is Tokenization required when dealing with Indigenous Languages? Definitely, before we can process natural language, we must first identify the words that comprise the strings. This is why creating tokens is a fundamental step forward in NLP (text data). This is significant because analyzing the text beforehand allows the meaning of the text to be easily interpreted.

Once the tokens have been received, they must be classified. There is a half-speech tagging, in which a program category like noun or action is mapped to received tokens. Part of the process of marking a speech is translating a sentence into list of words and a list of rows. The basic foundation of marking that is part of speech is automatic tagging. This is accomplished through the use of the Default Tagger section. In the Default Tagger section, 'tag' is treated as a single argument. The singular noun tag NN works with most common part of speech tag, Default Tagger comes in handy. As a result, a name tag is advised.

The tokens when separated from the language model, the system can now generate complex textual presentations for analysis. This is referred to as analysis. Separation refers to the process of determining the structure of a text. The mathematical algorithm accomplishes this by utilizing the grammar in which the text is written.

Further, dependency parsing means the language that establishes a direct relationship between words or sentences is known as dependent grammar. The process of using dependency grammar to prevent my own sentence formation is known as dependency parsing [14-15]. Various tags in Dependency Separation indicate the relationship between two words in a sentence. These tags are dependent on one another. In the sentence 'rain,' for example, the word rainy modifies the meaning of the noun weather. As a result, there is dependence

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6945
***Article Received**: 30 March 2023 **Revised**: 19 May 2023 **Accepted**: 01 June 2023*

on weather conditions -> it rains when the weather acts as the head and the rain acts as a dependent or child. The amod tag, which represents the adjectival modifier, represents this dependency.

The structure of grammatical structures is a model of usage that employs unambiguous nodes with respect to words and other incomprehensible categories and their indirect relationships. The method of identifying sentence structure using a constituency grammar is referred to as regional segregation. [16]

The membrane removal and membrane formation are for removing all conjunctions like suffixes, prefixes, etc. used in the word in order to preserve lexical structure. It is also regarded as root/stem/dictionary or lemma form. The key difference between the two processes is that the stem is generally rule based, that reduce the beginnings and endings of words , while lemmatization uses dictionaries and very complex behavioral analysis. [13]

To provide an accurate automatic text analysis, we must eliminate words which are not relevant for semantic information and are meaningless. These words are also termed as stop words like a, or, and, the, and so on [11]. In every language, there are numerous lists of stop words. However, it is critical to understand that depending on the text to be analyzed, we may need to add or remove words from the list. To determine which words should be included in the list of stop words, we may want to conduct an open-ended analysis of the content of our texts.

We know that informal text analysis is unreliable. There are methods for textual analysis, but the two most critical ones are “text classification” and “text extraction”. Text classification is the process of classifying tags as texts based on their content and Text-splitting earlier used to be done by hand, which was very time-consuming, inefficient, and inaccurate. Now automated text-analysis models work in seconds and with great accuracy.

A. *Text Classification*

The text classification functions include attributes for emotional analysis like if it conveys positive or negative impression, topic discovery to get the details about topic and objective discovery i.e determining the main purpose or objective of a text.

a) Rule based Systems

The separation of given text data and law is a man-made relationship between a language pattern found in text and tags. The rules typically refer to lexical, morphological or syntactic patterns, Semantics or phonology are the other aspect that can be referred.

Example of product differentiation:

(C ++ | Java | Python | R) → Editing Language

The benefits of legal-based programs are easy for humans to understand them. However, developing complex legal applications requires significant knowledge of both languages as well as the topics in the literature that the system must analyze.

Furthermore, legislation-based programs are tough to measure and sustain because new rules get introduced or any change in existing rules necessitates extensive impact.

b) Machine Learning-Based Systems

Machine Learning based systems learn from previous observations and data to predict the results. The programs require many text examples as well as predictable expectations. This is known as training data. The final predictions will be much better if our training details are consistent and accurate.

When using machine learning to train a division, the training data must be converted into something that the machine can comprehend, like vectors i.e., a list of numbers that contain data. The system can excerpt applicable topographies from current data and make forecasts for future data by using vectors. There are several approaches to this, but one of the most common is vectorization. After text conversion into vectors, a machine learning algorithm is applied on top of it, which should produce a segmentation model capable of determining which features represent texts more accurately and performing predictive text forecasts. The trained model is applied to unknown data text to a vector, performs feature extraction, and predicts data with Model Classification [14].

c) Machine learning algorithms

Few of the machine learning methods has been applied for text classification. The Naive Bayes algorithms SVM (Support Vector Machines) referred in Figure 1. This in-depth learning algorithm that is very widely used. [14]

Naive Bayes algorithm family is built on top of Bayes' Theorem and conditional obligations for the appearance of sample text names within a set of tagged text. Vectors representing documents include information about how text words can appear in the tags provided. The writing possibilities of any mark given to the model can be calculated using this information. The possible outcomes are included in the provided document, the marker with the highest probability as the output is returned by the segmentation model for the provided inputs [14]. This algorithm provides good results even with less training data.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6945
***Article Received**: 30 March 2023 **Revised**: 19 May 2023 **Accepted**: 01 June 2023*

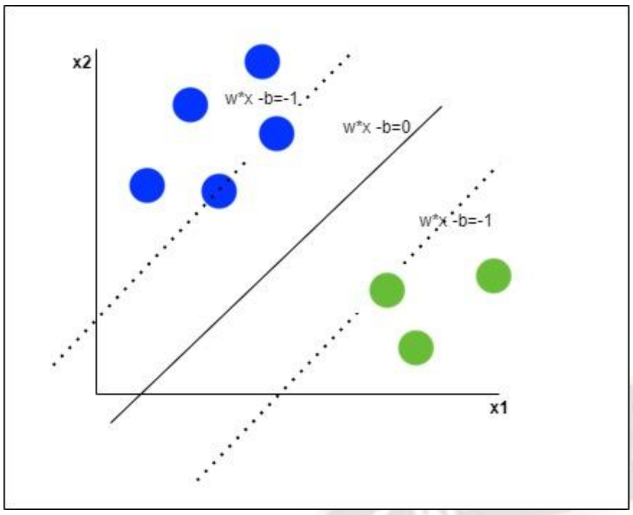


Figure 1. Support Vector Hyperplanes

Support Vector Hyperplanes (SVM) results are better compared to Naive Bayes, however, it requires high resources in training shown in Figure 1.

There is another algorithm that is very popular, that uses artificial neural networks to process data in a similar manner that the human brain does. The neural network uses a large amount of training data and uses extensive resource during training and perform excellently in predicting results [18]

B. *Hybrid Systems*

Hybrid Machine Learning typically is combination of machine learning and rule-based systems for better prediction [15]. Classifier performance is frequently evaluated using machine learning industry standard metrics such as correctness, meticulousness, memory, and F1-score. Comprehending the result will provide a clear picture of the effectiveness of classifier [16] for analysing the provided text data. [17]

The testing is done with limited test data, through cross-verification is required on the expectation from the result.

The process of knowing the edited pieces of information from an unavailable text is referred to as text extraction. It may be useful to automatically find keywords that are more relevant for analysing and predicting the text data like resume/CV.

The rules defined in the classification functions are followed by common expressions (regexes). The common denominator in this case describes the pattern recognition.

This approach has pros and cons, while text extractors are fast and accurate with the known patterns on the other hand, it can be had for maintenance as anytime there is change of rules, it needs to be updated otherwise results will be incorrect.

C. *Open-source library/API*

The text analysis solutions are available in the form of open source library as well as software as a service APIs. The library usage needs good technical understanding, while software as a service provides faster text analysis solution.

a) Python

Python is one of the best and very widely used computer programming language. NumPy and SciPy have ability to call C and Fortran libraries whenever required. It has great community support and has number of libraries to support data science and natural language processing (NLP), which has helped Python to become one of the most popular scripting languages.

b) NLTK

Natural Language Toolkit library also called as NLTK, is a cutting-edge library of analysing text. Using the Rapid Automatic Keyword Extraction Rake algorithm and the NLTK tool kit, you can create a powerful keyword extraction tool. Known as Rake NLTK.

c) SpaCy

SpaCy is a statistical NLP library for industry. It enhances the standard features by combining in-depth learning with multimedia neural network models. Spacy is a free and open library that makes use of Natural Language Development (NLP) [18]. It is written in Python and Cython and is especially useful for applications that require text comprehension. It is used in deep reading to process text (Machine Learning section). SpaCy, in contrast to NLTK text-based library.

d) Scikit-learning

Skikit-learning is an advances Python Data science and ML library built on top of SciPy, NumPy and matplotlib, that provides very good performance in building text-based analysis models.

D. *High Level Design*

The Artificial intelligence-based resume parser provides user the ability to upload a file, and based on the type of the file uploaded, it applies text extraction by calling the appropriate API like reading from PDF or from word format. The text is extracted and transformed further by applying various techniques like tokenization, cleansing by removing stop words. Further, a hybrid approach of rule based and NLP based algorithm is applied to extract specific information. As an end state, the extracted content is matched against the resume description and cosine similarity and cosine distance algorithms are used to derive the prediction. Figure 2, explains the high-level design.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6945
***Article Received**: 30 March 2023 **Revised**: 19 May 2023 **Accepted**: 01 June 2023*

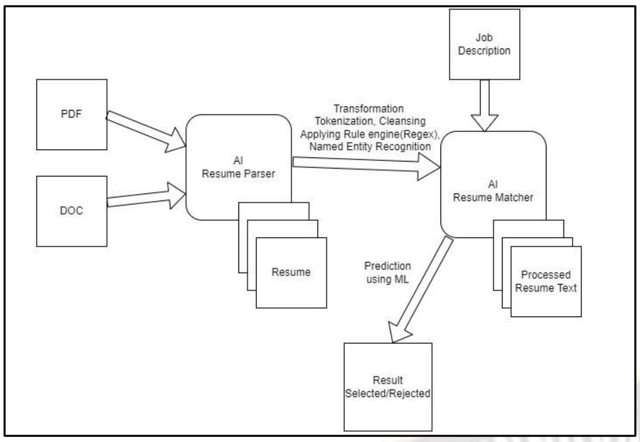


Figure 2. Resume Parser using AI/NLP: System Design

III. Results and Discussion

A. *Pilot implementation through Python*

This section will go over the step-by-step pilot implementation in Python [19].

a) Pre-processing

The resume is provided as input in this process and is cleaned to eliminate any unusual or jumble characters. All unusual characters and single-letter words like 'a', '.' etc. are removed during cleaning. Following these steps, a clean dataset is obtained. It is further tokenized using NLTK tokenizers [20]. Stop-word removal along with stemming, and lemmatization are some of the pre-processing steps used on tokenized datasets. The raw CV file is imported, and the resume/CV content data is cleansed. The System design of Resume parser is shown in Figure 2.

b) Reading the Resume

The critical issue with CV/Resumes is that they are not in structured form and there is no standard file format, like .pdf,.doc, or.docx. So, our first step is to convert the CV into plain text irrespective of the format. We have used mainly two Python modules for this: PyMuPDF and doc2text. These modules help in the extraction of text from.pdf,.doc, and.docx file formats.

c) Python code to extract text from PDF

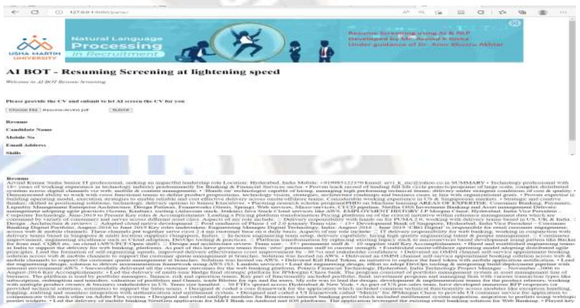
```
import sys, fitz

""""

pip install PyMuPDF
```

This PyMuPDF library works better than PyPDF2, tested with all types of PDF extract_text_from_doc function extracts the text of the given resume Library is developed as part of research project in Usha Martin University Usage:read_text_from_pdf API needs filename as input parameter

Output: Text content retrieved from the resume file passed

```
""""

def read_text_from_pdf(filename):

    document = fitz.open(filename)

    temp_resume_text = ""

    for resume_pg in document:

        temp_resume_text=temp_resume_text + str(resume_pg.get_text())

    tx = " ".join(temp_resume_text.split('\n'))

    return tx

resume_content =read_text_from_pdf('Resume-Arvind.pdf')

print(resume_content)
```

Output: It is shown in Figure 3.



AI BOT - Resuming Sereening at lightening speed

Welcome to AI BOT Resume Screening

Please provide the CV and submit to let AI screen the CV for you

Resume
Candidate Name
Mobile No
Email Address
Skills

Resume
[illegible]

Figure 3. Shows the tools screen to upload resume

d) Reading doc and docx to get the resume text content:

```
import docx2txt

import re

""""
```

extract_text_from_doc function extracts the text of the given resume

Library is developed as part of research project in Usha Martin University

Usage:read_resume_with_doc_extension API needs filename as input parameter

Output:Text content retrieved from the resume file passed

```
""""

def read_resume_with_doc_extension(filename):

    print('Reading MS Word file')

    temp = docx2txt.process(filename)

    resume_content = [rows.replace('\t', ' ') for rows in temp.split('\n') if rows]

    return ' '.join(resume_content)
```

resumeContent = read_resume_with_doc_extension(Arvind.docx')

print(resumeContent)

Partial Output is shown in Figure 4.

```
Arvind Kumar Sinha Senior IT professional, seeking an impactful
leadership role Location:  Hyderabad, India Mobile: +919985122370
Email:  arvi_k_rnc@yahoo.co.in SUMMARY • Technology professional
with 18+ years of working experience in technology industry
predominantly for Banking & Financial Services sector.  • Proven track
record of leading full life cycle projects/programs of large scale, complex
distributed systems across digital channels viz web, mobile & content
management.  • 'Hands on' technologist capable of hiring, managing
high performing technical teams; delivery  under stringent conditions of
cost & quality.  • Demonstrated ability to work with cross functional
teams to define product propositions,  technology vision, strategies,
architecture roadmaps and business cases in line with  organizational
goals. • Proven expertise in building operating model, execution
strategies to enable reliable and cost  effective delivery across
onsite/offshore teams. Considerable working experience in US &
Singaporean markets. • Strategic and creative thinker; skilled in
positioning solutions, technology, delivery options to  Senior Executives.
• Pursuing research scholar program(PHD) on Machine learning
AREAS OF EXPERTISE: Consumer Banking, Payments,  Liquidity
Management Enterprise Architecture, Design Patterns, Frameworks
(Struts, Spring), Web services, Micro-services, CI/CD  pipeline, Netflix
OSS  Investment Banking and Asset  management Delivery management
adopting agile practices  (Scrum, Kanban), tooling (Atlassian stack,
Jenkins)  Java Stack Cloud services like AWS.  EMPLOYMENT
HISTORY J.P Morgan Chase, Hyderabad, India Vice President –
Corporate Technology, June-2019 to Present Key
```

Figure 4. Partial Output

e) *Extracting Name*

Regular expressions can be used to extract names from resumes. However, we have employed natural language processing library for text, spaCy. It is a text processing module with industrial-strength natural language processing. It includes models for tagging, parsing, and entity recognition that have already been trained.

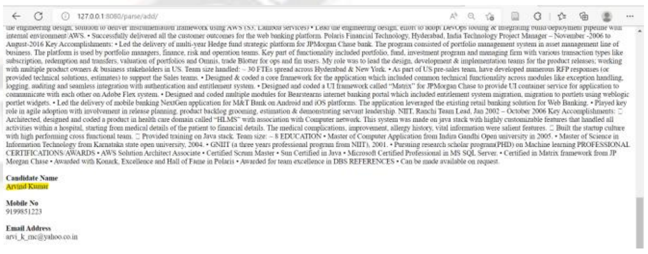
Installing spaCy

The command to install spaCy is shown below

```
pip install spacy

python -m spacy download en_core_web_sm

import spacy

from spacy.matcher import Matcher

import docfilereader as docr

"""Load a spaCy model from an installed package or a local path.

extract_candidate_name function extracts the name of a candidate from

the resume content passed to it

Library is developed as part of research project in Usha Martin University


Usage:get_candidate_name API needs resume content(text data) as input parameter

Output: Name of the candidate is extracted from the resume content passed in form of text

"""

nlp_for_ne = spacy.load('en_core_web_sm')

spacy_matcher = Matcher(nlp_for_ne.vocab)

def get_candidate_name(resume_content_data):

    print(resume_content_data)

    nlp_text = nlp_for_ne(resume_content_data)

    # to get name out of the resume content

    ne_pattern = [{'POS': 'PROPN'}, {'POS': 'PROPN'}]

    spacy_matcher.add('NAME', [ne_pattern])

    name_found = spacy_matcher(nlp_text, as_spans=True)

    print(name_found[0])

    return name_found[0]

resume_content = docr.read_resume_with_doc_extension(Arvind.docx')

candidate_name = get_candidate_name(resume_content)

print("Candidate Name : " + str(candidate_name))
```

Output: It is shown in Figure 5.

Candidate Name: Arvind Kumar



the engineering design, solution to deliver instrumentation framework using AWS (S3, Lambda services) • Lead the engineering design, effort to adopt DevOps tooling & integrating build-deployment pipeline with internal environment/AWS. • Successfully delivered all the customer outcomes for the web banking platform. Polaris Financial Technology, Hyderabad, India Technology Project Manager – November -2006 to August-2016 Key Accomplishments: • Led the delivery of multi-year Hedge fund strategic platform for JPMorgan Chase bank. The program consisted of portfolio management system in asset management line of business. The platform is used by portfolio managers, finance, risk and operation teams. Key part of functionality included portfolio, fund, investment program and managing firm with various transaction types like subscription, redemption and transfers, valuation of portfolios and Omnis, trade Blotter for ops and fin users. My role was to lead the design, development & implementation teams for the product releases; working with multiple product owners & business stakeholders in US. Team size handled: ~ 30 FTEs spread across Hyderabad & New York. • As part of US pre-sales team, have developed numerous RFP responses (or provided technical solutions, estimates) to support the Sales teams. • Designed & coded a core framework for the application which included common technical functionality across modules like exception handling, logging, auditing and seamless integration with authentication and entitlement system. • Designed and coded a UI framework called "Matrix" for JPMorgan Chase to provide UI container service for application to communicate with each other on Adobe Flex system. • Designed and coded multiple modules for Bearstearns internet banking portal which included entitlement system migration, migration to portlets using weblogic portlet widgets. • Led the delivery of mobile banking NextGen application for M&T Bank on Android and iOS platforms. The application leveraged the existing retail banking solution for Web Banking. • Played key role in agile adoption with involvement in release planning, product backlog grooming, estimation & demonstrating servant leadership. NIIT, Ranchi Team Lead, Jan 2002 – October 2006 Key Accomplishments: Architected, designed and coded a product in health care domain called "HLMS" with association with Computer network. This system was made on java stack with highly customizable features that handled all activities within a hospital, starting from medical details of the patient to financial details. The medical complications, improvement, allergy history, vital information were salient features. Built the startup culture with high perfoming cross functional team. Provided training on Java stack. Team size: ~ 8 EDUCATION • Master of Computer Application from Indira Gandhi Open university in 2005. • Master of Science in Information Technology from Karnataka state open university, 2004. • GNIIT (a three years professional program from NIIT), 2001. • Pursuing research scholar program(PHD) on Machine learning PROFESSIONAL CERTIFICATIONS/AWARDS • AWS Solution Architect Associate • Certified Scrum Master • Sun Certified in Java • Microsoft Certified Professional in MS SQL Server. • Certified in Matrix framework from JP Morgan Chase • Awarded with Konark, Excellence and Hall of Fame in Polaris • Awarded for team excellence in DBS REFERENCES • Can be made available on request.

Candidate Name
Arvind Kumar

Mobile No
9199851223

Email Address
arvi_k_rnc@yahoo.co.in

Figure 5. Shows the name extraction from resume

f) *Extracting Phone Numbers*

We have used regular expressions to extract phone numbers. Phone numbers can also be written as (+91) 1234567890, +911234567890, +91 123 456 7890, or +91 1234567890. As a result, regular expression can be used to get the phone number of the candidate.

```
import re
import docfilereader as docr
""""
extract_mobile_number function extracts the mobile number of the given candidate
from the given resume
Library is developed as part of research project in Usha Martin University
Usage:get_candidate_mobile_number API needs resume content(text data) as input parameter
Output:Mobile number of the candidate is extracted from the resume content passed in form of text
""""
mobile_no = ""
def get_candidate_mobile_number(resume):
        #standard reg ex is used to fetch mob no
    mobile_no = re.findall(re.compile(
        r'(?:(?:\+?([1-9]|[0-9][0-9]|[0-9][0-9][0-9])\s*(?:[.-
]\s*)?)?(?:\(\s*([2-9]1[02-9]|[2-9][02-8]1|[2-9][02-8][02-
9])\s*\)|([0-9][1-9]|[0-9]1[02-9]|[2-9][02-8]1|[2-9][02-8][02-
9]))\s*(?:[.-]\s*)?)?([2-9]1[02-9]|[2-9][02-9]1|[2-9][02-
9]{2})\s*(?:[.-]\s*)?([0-
9]{4})(?:\s*(?:#|x\.?|ext\.?|extension)\s*(\d+))?'),
        resume)
    if mobile_no:
        cand_ph_n = ''.join(mobile_no[0])
        if len(cand_ph_n) > 10:
            return '+' + cand_ph_n
        else:
            return cand_ph_n
resume_content = docr.read_resume_with_doc_extension("Arvind.docx")
num = get_candidate_mobile_number(resume_content)
print('Mobile Number : ' + num)
```

Output: It is shown in Figure 6.

```
Mobile Number: 9985122370
```

Figure 6. Mobile Number

g) Extracting Email

We can use the same method we used to extract mobile numbers to extract email addresses from resumes. Email addresses have a known pattern, and regular expression provides good result.

```
import re
import docfilereader as docr
""""
extract_email function extracts the email of the given candidate from the resume
Library is developed as part of research project in Usha Martin University
Usage:get_candidate_mobile_number API needs resume content(text data) as input parameter
Output: Email address of the candidate is extracted from the resume content passed in form of text
""""
def extract_email(resume):
    candidate_em = re.findall("([^@|\s]+@[^@]+\.[^@|\s]+)", resume)
    if candidate_em:
        try:
            return candidate_em[0].split()[0].strip(';')
        except IndexError:
            return ""
resume_content = docr.read_resume_with_doc_extension("Arvind.docx")
email = extract_email(resume_content)
print("Email Address extracted:"+email)
```

Output: It is shown in Figure 7.

```
Email Address extracted: arvi_k_rnc@yahoo.co.in
```

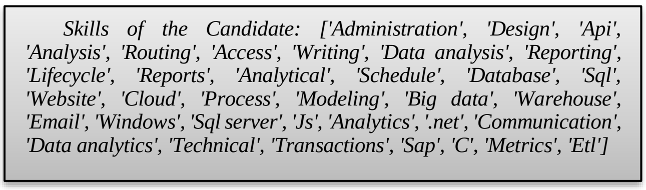
Figure 7. Email Address

h) Extracting Skills

```
import pandas as pd
import nameextractor as ne
import docfilereader as docr
import spacy
```

```
"""Load a spaCy model from an installed package or a local path.

extract_candidate_name function extracts the name of a candidate from

the resume content passed to it

Library is developed as part of research project in Usha Martin University

Usage:get_candidate_skills API needs resume content(text data) as input parameter

Output: Skills of the candidate is extracted from the resume content passed in form of text

"""

nlp_skill = spacy.load('en_core_web_sm')

def get_candidate_skills(resume_comtent_skills):

    content = nlp_skill(resume_comtent_skills)

    noun_chunks = content.noun_chunks

    nominal = [nom.text for nom in content if not nom.is_stop]

    print(nominal)

    #

    data = pd.read_csv("allskills.csv")

    candidate_skills = list(data.columns.values)

    all_av_skills = []

    for token in nominal:

        if token.lower() in candidate_skills:

            all_av_skills.append(token)

    for token in noun_chunks:

        token = token.text.lower().strip()

        if token in candidate_skills:

            all_av_skills.append(token)

    return [i.capitalize() for i in set([i.lower() for i in all_av_skills])]

resume_content = docr.read_resume_with_doc_extension(Arvind.docx')

skills = get_candidate_skills(resume_content)

print("Skills of the candidate : " + str(skills))
```

Output: It is shown in Figure 8.



Skills of the Candidate: ['Administration', 'Design', 'Api', 'Analysis', 'Routing', 'Access', 'Writing', 'Data analysis', 'Reporting', 'Lifecycle', 'Reports', 'Analytical', 'Schedule', 'Database', 'Sql', 'Website', 'Cloud', 'Process', 'Modeling', 'Big data', 'Warehouse', 'Email', 'Windows', 'Sql server', 'Js', 'Analytics', '.net', 'Communication', 'Data analytics', 'Technical', 'Transactions', 'Sap', 'C', 'Metrics', 'Etl']

Figure 8. Skills of the Candidate

i) Extracting Education

Moving on to the final step, we have extracted candidate's education information. The specific details that we will be obtaining are grade and year of degree. We have used the nltk module to load an entire list of stopwords and then remove them from our resume text.

Installing nltk:

```
pip install nltk

python -m nltk nltk.download('words')
```

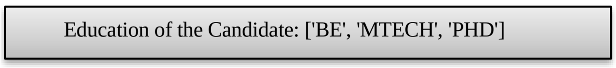
Recruiters are very specific about the level of education or degree required for a specific job. As a result, we will be preparing a list of EDUCATION that will specify all of the equivalent degrees that are required.

```
import re

import spacy

import nameextractor as ne

from nltk.corpus import stopwords

import docfilereader as docr

import pandas as pd

"""Load a spaCy model from an installed package or a local path.

extract_candidate_education function extracts the education of a candidate from

the resume connect passed to it

Library is developed as part of research project in Usha Martin University

Usage:get_candidate_mobile_number API needs resume content(text data) as input parameter

Output: Email address of the candidate is extracted from the resume content passed in form of text

"""

nlp_qualification = spacy.load('en_core_web_sm')

# Grad all general stop words
```

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6945
***Article Received**: 30 March 2023 **Revised**: 19 May 2023 **Accepted**: 01 June 2023*

```
STOPWORDS_IN_RESUME = set(stopwords.words('english'))

def get_candidate_education(resume_content_qual):
    nlp_qual = nlp_qualification(resume_content_qual)
    nlp_qual.sents
    candidate_education = pd.read_csv("degree.csv")
    CANDIDATE_QUAL = list(candidate_education)
    nlp_qual = [qual_s.text.strip() for qual_s in nlp_qual.sents]
    edu = {}
    for idx, txt in enumerate(nlp_qual):
        for tex in txt.split():
            tex = re.sub(r'[?|$|.|!|,]', r'', tex)
            if tex.upper() in CANDIDATE_QUAL and tex not in STOPWORDS_IN_RESUME:
                edu[tex] = txt + nlp_qual[idx]
    qual = []
    for pk in edu.keys():
        yop = re.search(re.compile(r'(((20|19)(\d{2})))'), edu[pk])
        if yop:
            qual.append((pk, ''.join(yop[0])))
        else:
            qual.append(pk)
    return qual

resume_content = docr.read_resume_with_doc_extension(Arvind.docx')
education = get_candidate_education(resume_content)
print("Education of the Candidate : " + str(education))
```

Output: It is shown in Figure 9.



```
Education of the Candidate: ['BE', 'MTECH', 'PHD']
```

Figure 9. Education of the Candidate

Code Available at GitHub:

https://github.com/arvindindia123/ResumeParser

https://github.com/arvindindia123/ResumeParserUI

B. *Findings*

This paper discusses segmenting resumes in order to extract data and perform text analysis. Python is used to demonstrate the step-by-step pilot implementation. The pilot implementation accepts the resume in PDF/Doc format and is cleansed as part of first step to remove stop-words, punctuations etc. Cleansing and removing unnecessary content is the first and critical step to perform natural language processing. The raw CV file was imported, and the resume field data was cleansed to remove extra spaces and numbers in the date. To extract names from resumes, regular expressions are used. However, we have also used spaCy library which is considered the most accurate natural language processing library. Spacy is an industrial-strength natural language processing text and language processing module. It includes already trained models for entity recognition, parsing, tagging etc. Regular expressions were also used to extract phone numbers and emails. Email IDs have a consistent format: an alphanumeric string followed by a @ symbol, another string, a. (dot), followed by a string (com, in, org etc.). Furthermore, the candidate's educational information is extracted, as well as qualification and passing year. For example, if candidate completed PhD in 2022, we extracted a row that looks like (PhD, '2022'). All stop-words must be removed as a result of this. The NLTK module was used to load an entire list of stop-words and then remove from our resume text for further processing.

The application using hybrid approach of AI and regular expression, which makes it unique. It has achieved 93.5% accuracy in text-extraction which improves the accuracy by 19.5% compared to CNN models.

IV. Conclusion

In this paper, we have demonstrated the step-by-step pilot implementation of resume parsing using Python. The resume is provided as input in this process and is cleaned to remove any unusual or junk characters. The CV file is imported, and based on PDF or MS word, appropriate read method is called to get the CV content in text format. It was cleansed to remove extra spaces and dates along with removal of stop-words which doesn't add value in decision making. Hybrid approach has been adopted, where in trained models are used along with rule based regular expression approach in different scenario. This resume parse will further be used for decision making of screening of a candidate against the given job description.

References

[1] A. Sinha, Md. A. K. Akhtar, and A. Kumar, Resume Screening using Natural Language Processing and Machine Learning: A Systematic Review. In: Swain, D., Pattnaik, P.K., Athawale, T. (eds) Machine Learning and Information Processing. Advances in Intelligent Systems and Computing, vol 1311. 2021 Springer, Singapore.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6945
***Article Received**: 30 March 2023 **Revised**: 19 May 2023 **Accepted**: 01 June 2023*

[2] D. Çelik et al., "Towards an Information Extraction System Based on Ontology to Match Resumes and Jobs," in 2013 IEEE 37th Annual Computer Software and Applications Conference Workshops, Jul. 2013, pp. 333–338. doi: 10.1109/COMPSACW.2013.60.

[3] T. Kiss and J. Strunk, "Unsupervised multilingual sentence boundary detection. Computational Linguistics," pp. 485–525, 2006.

[4] J. C. Reynar and A. Ratnaparkhi, "A maximum entropy approach to identifying sentence boundaries. Proceedings of the Fifth Conference on Applied Natural Language Processing," pp. 16–19, 1997.

[5] M. D. Riley, "Some applications of tree-based modelling to speech and language. Proceedings of the Workshop on Speech and Natural Language," pp. 339–352, 1989.

[6] "The Stanford Natural Language Processing Group." https://nlp.stanford.edu/software/tokenizer.shtml (accessed Jul. 08, 2021).

[7] C. D. Manning and H. Schütze, "Foundations of Statistical Natural Language Processing. MIT Press.," 1999.

[8] B. Jurish and K. M. Würzner, "Word and Sentence Tokenization with Hidden Markov Models.," pp. 61–83, 2013.

[9] "UCREL CLAWS5 Tagset." http://ucrel.lancs.ac.uk/claws5tags.html (accessed Jul. 08, 2021).

[10] L. Derczynski, D. Maynard, G. Rizzo, and M. Van Erp, "Analysis of named entity recognition and linking for tweets. Information Processing & Management," pp. 32–49, 2015.

[11] "Text Analysis Starter Guide: What You Need to Know," MonkeyLearn. https://monkeylearn.com/text-analysis/ (accessed Jul. 08, 2021).

[12] "Cluster analysis," Wikipedia. Jun. 29, 2021. Accessed: Jul. 08, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=1031035663

[13] "Python Stemming Lemmatization." https://www.python-ds.com/python-stemming-lemmatization (accessed Jul. 08, 2021).

[14] P. K. Roy, S. S. Chowdhary, and R. Bhatia, "A Machine Learning approach for automation of Resume Recommendation system," 2019.

[15] "Machine Learning with Python: Metrics: Accuracy, precision, recall, F1-Score." https://www.python-course.eu/metrics.php (accessed Jul. 08, 2021).

[16] A. Hetherington, "Evaluating Classifier Model Performance," Medium, Jul. 05, 2020. https://towardsdatascience.com/evaluating-classifier-model-performance-6403577c1010 (accessed Jul. 08, 2021).

[17] "Precision vs Recall | Precision and Recall Machine Learning," Analytics Vidhya, Sep. 03, 2020. https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/ (accessed Jul. 08, 2021).

[18] "spaCy · Industrial-strength Natural Language Processing in Python." https://spacy.io/ (accessed Jul. 08, 2021).

[19] O. Pathak, OmkarPathak/ResumeParser. 2021. Accessed: Jul. 08, 2021. [Online]. Available: https://github.com/OmkarPathak/ResumeParser

[20] E. Loper and S. Bird, "Nltk: the natural language toolkit," 2002.

[21] H. Shah, N. ., T. Khan, D. ., A. Banu, A. ., & H. Shah, L. . (2023). Symmetric and Asymmetric Encryption Schemes for Internet of Things: A Survey . International Journal of Intelligent Systems and Applications in Engineering, 11(1), 254–260. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2465

[22] Md Amir Khusru Akhtar, Mohit Kumar, and Gadadhar Sahoo. "Automata for santali language processing." In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 939-943. IEEE, 2017.

[23] Md Amir Khusru Akhtar, Gadadhar Sahoo, and Mohit Kumar. "Digital corpus of Santali language." In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 934-938. IEEE, 2017.