SIMATS SCHOOL OF ENGINEERING
Approved By AICTE | IET-UK Accreditation

RESUME PARSING SYSTEM USING NLP

Authors:

Name: P.Naveen

Register No.:192210649

Name: K.Rajesh

Register No.:192211900

Name: D.K.V.S.K.Manohar

Register No.:19221861

Project Outline:

1 INTRODUCTION

In today's fast-paced job market, the process of hiring has become increasingly complex, with recruiters inundated by a vast volume of resumes for each job opening. To streamline this process and ensure efficient candidate selection, companies are turning to advanced technologies such as Natural Language Processing (NLP).

Resume parsing systems utilizing NLP techniques offer a sophisticated solution to automate the extraction, analysis, and organization of pertinent information from resumes. By leveraging the power of machine learning and linguistic algorithms, these systems can swiftly interpret unstructured resume data, transforming it into structured, actionable insights.

The primary goal of a resume parsing system is to enable recruiters and hiring managers to quickly identify top candidates by efficiently extracting key details such as skills, experience, education, and qualifications. NLP algorithms play a pivotal role in this process, enabling the system to understand the context, semantics, and relationships within the text, thereby enhancing accuracy and relevance.

Consider, for instance, the multifaceted nature of skills listed on a resume. While traditional parsing systems may merely identify keywords, NLPenabled systems delve deeper, discerning the depth of expertise, relevance to the job role, and even the synonyms and related terms that expand the scope of understanding. This nuanced comprehension empowers recruiters with invaluable insights, enabling them to pinpoint candidates whose proficiencies precisely match the demands of the position.

Moreover, NLP-driven parsing systems possess the agility to adapt and evolve in tandem with the everchanging dynamics of the job market. With the ability to customize parsing rules, refine algorithms, and integrate feedback loops, these systems continually refine their capabilities, ensuring relevance and accuracy across diverse industries and job roles.

Yet, beyond mere efficiency gains, the integration of NLP into resume parsing heralds a paradigm shift in recruitment practices. By automating the mundane tasks of information extraction and organization, recruiters are liberated to focus their expertise on higher-order strategic activities—such as candidate engagement, relationship building, and cultural fit assessment—that are essential for fostering longterm organizational success.

In essence, NLP-infused resume parsing represents more than just a technological advancement; it embodies a fundamental reimagining of the recruitment process—one that transcends the constraints of time and scale to elevate the art of talent acquisition to new heights. As organizations navigate the complexities of talent management in an increasingly competitive landscape, the adoption of NLP-driven parsing systems emerges not merely as a choice but as an imperative—a catalyst for unlocking the full potential of human capital and driving sustained organizational growth.

2 RATIONALE AND RELAVANCE

RELATIONALE:

In the intricate tapestry of modern recruitment, the synergy between cutting-edge technologies and human expertise emerges as a cornerstone for success. Within this context, Natural Language Processing (NLP) emerges as a transformative force, reshaping the landscape of resume parsing and unlocking new dimensions of efficiency, accuracy, and relevance.

At its essence, the relationship between NLP and resume parsing is symbiotic—a fusion of machine intelligence and human intuition that transcends traditional boundaries to redefine the recruitment paradigm. NLP empowers parsing systems with the cognitive capabilities to comprehend, interpret, and extract meaning from the rich tapestry of language woven into resumes. Through advanced algorithms and linguistic models, these systems navigate the intricacies of syntax, semantics, and context, unraveling the intricacies of each candidate's professional narrative with unprecedented depth and precision.

RELAVANCE:

In today's competitive job market, Resume Parsing Systems (RPS) leveraging Natural Language Processing (NLP) technology offer indispensable benefits. These systems enhance efficiency by automating the extraction and categorization of key information from resumes, expediting the hiring process. NLP algorithms enable RPS to accurately interpret language nuances, ensuring that candidates with the most relevant skills and experiences are prioritized. Moreover, RPS can be customized to align with specific job requirements and organizational preferences, promoting flexibility and adaptability in recruitment strategies. By applying objective criteria, NLP-powered RPS mitigate bias in candidate selection, fostering diversity and inclusion within the workforce. Additionally, these systems offer scalability to handle high-volume recruitment campaigns efficiently and provide valuable insights to inform strategic decisionmaking. Overall, the relevance of RPS using NLP lies in their ability to streamline recruitment processes, reduce bias, and drive organizational success in talent acquisition efforts.

3 ABSTRACT

In company's or organization's recruiting process, for a single opening there are multiple applying candidates. First process in recruiting procedure is to go with resumes one by one for the selection of best candidate. But this is time consuming and hectic process for recruiters. The reason behind this is each applicant has its own unique resume with different sections and different formats. So, it is not possible for recruiters to go with resumes one by one. To minimize the efforts of recruiters and investment of timing in this process we have proposed a system where recruiters can easily analyze resumes in simple file formats with the ranking. This project is based on Natural Language Processing (NLP). Two major components of this system are Job Seekers and Recruiters. Applicants will upload their resume in different formats mainly pdf or doc. Then parser will parse these resumes for field extraction, after analysis system will scale the resume and will give suggestions like required skills to be included in resume or courses to be done or fields to be included in resume to increase its rank. This ranking will be saved in databases where recruiter will get idea about most deserving candidates for a particular role.

Keywords: : Natural Language Processing (NLP), Resume Parser, Resume Analysis, Part-of-speech tagging, Named Entity Recognition (NER)

4 OBJECTIVE

The objective of integrating a Resume Parsing

System (RPS) utilizing Natural Language Processing (NLP) technology is to revolutionize and optimize the recruitment process. By leveraging NLP capabilities, the aim is to streamline the tedious task of resume screening, thereby enhancing efficiency and accuracy in candidate selection. This objective encompasses several key goals, including automating the extraction and categorization of pertinent information from resumes to expedite the screening process, ensuring that NLP algorithms accurately interpret language nuances to identify candidates with the most relevant skills and experiences. Additionally, customization features enable tailoring parsing rules to align with specific job requirements and organizational preferences, fostering a personalized and adaptable approach to resume analysis. Furthermore, the system aims to mitigate bias by applying objective criteria for candidate evaluation, promoting fairness and inclusivity in recruitment practices. Scalability is also crucial, with the system designed to efficiently handle large volumes of resumes without compromising performance or accuracy. Lastly, providing recruiters with valuable insights derived from resume data aims to inform strategic decisionmaking, refine recruitment strategies, and ultimately optimize hiring outcomes over time. Overall, the objective of implementing an RPS using NLP is to enhance talent acquisition efforts, drive organizational growth, and ensure the selection of the most suitable candidates for available positions.

5 APPLICATIONS

The applications of a Resume Parsing System (RPS) using Natural Language Processing (NLP) include:

1. Efficient Recruitment: Automating resume screening to quickly identify top candidates.
2. Candidate Matching: Matching candidates with job openings based on qualifications.
3. Candidate Relationship Management: Organizing candidate data for effective engagement.
4. Talent Pool Management: Building and managing pools of potential candidates.
5. Compliance and Reporting: Ensuring compliance and generating recruitment analytics.

6. Employee Referral Programs: Streamlining the referral process for employee-recommended candidates.

6 PROCEDURES AND METHODOLOGY

PROCEDURES:

The procedures involved in implementing a Resume Parsing System (RPS) using Natural Language Processing (NLP) typically include:

Requirement Gathering: Understand the specific needs and objectives of the organization regarding resume parsing and candidate screening.

Data Collection: Gather a diverse set of resumes to train the NLP model, ensuring it can accurately parse and extract relevant information.

Preprocessing: Clean and preprocess the resume data to remove noise, standardize formats, and ensure consistency for effective parsing.

NLP Model Training: Develop and train the NLP model using machine learning algorithms to understand language nuances, extract key information, and categorize resumes based on criteria such as skills, experience, and qualifications.

System Integration: Integrate the NLP-powered parsing system with existing recruitment software or Applicant Tracking Systems (ATS) to streamline the resume screening process.

Testing and Validation: Conduct thorough testing to ensure the accuracy and reliability of the parsing system, validating its performance against a diverse set of resumes and real-world scenarios.

Customization: Customize parsing rules and criteria based on specific job requirements, industries, and organizational preferences to enhance accuracy and relevance.

Deployment: Deploy the NLP-powered RPS into production, ensuring seamless operation and compatibility with existing recruitment workflows.

Monitoring and Maintenance: Continuously monitor the performance of the parsing system, addressing any issues or discrepancies that may arise, and periodically update the NLP model to improve accuracy and adaptability.

METHODOLOGY:

There are several techniques and algorithms are available to solve NLP based problems. As these problems are based on Deep Learning concepts, Python Language is preferred. Libraries available in Python language such as nltk and spacy are used to extract text/information from documents. For text cleaning, Regular Expression (RE) library is used. NLTK and Spacy libraries are used for Natural Language Processing (NLP) related tasks like eliminating stop words, extracting root words, POS, NER. Pre-processing data is a difficult task. As text preprocessing is the initial stage of any NLP project, text pre-processing is done in order to prepare the text data.

The following are some of the pre-processing steps:

- Removing Stop words
- Lower casing
- Tokenization
- Lemmatization Steps in Parsing:

a) Extract the text from pdf

b) Extract Text from Doc files

c) Detect the file extension

d) Extract the entities

e) Extract the email

f) Extract Name from Resume

g) Extract Mobile Number

h) Extract Skills from the resume

i) Extract Technical Skills

j) Extract Education and Year

k) Extract Experience

After parsing analysis is done on extracted fields and stored in database which can be used by recruiter's visualization and decision process also it will be used for job seeker's suggestions.

7 RESEARCH AND ANALYSIS

- Research and analysis for a resume parsing system using natural language processing (NLP) is a multifaceted process that encompasses various key activities.
- This includes exploring existing solutions in the market, gathering user requirements from stakeholders such as recruiters and HR professionals, delving into the technological

landscape of NLP techniques and frameworks, collecting and annotating a diverse dataset of resumes for training and evaluation, assessing model performance metrics, ensuring compliance with data privacy regulations, conducting cost-benefit analysis to evaluate the feasibility and potential benefits of the system, and identifying and mitigating potential risks associated with development and deployment.

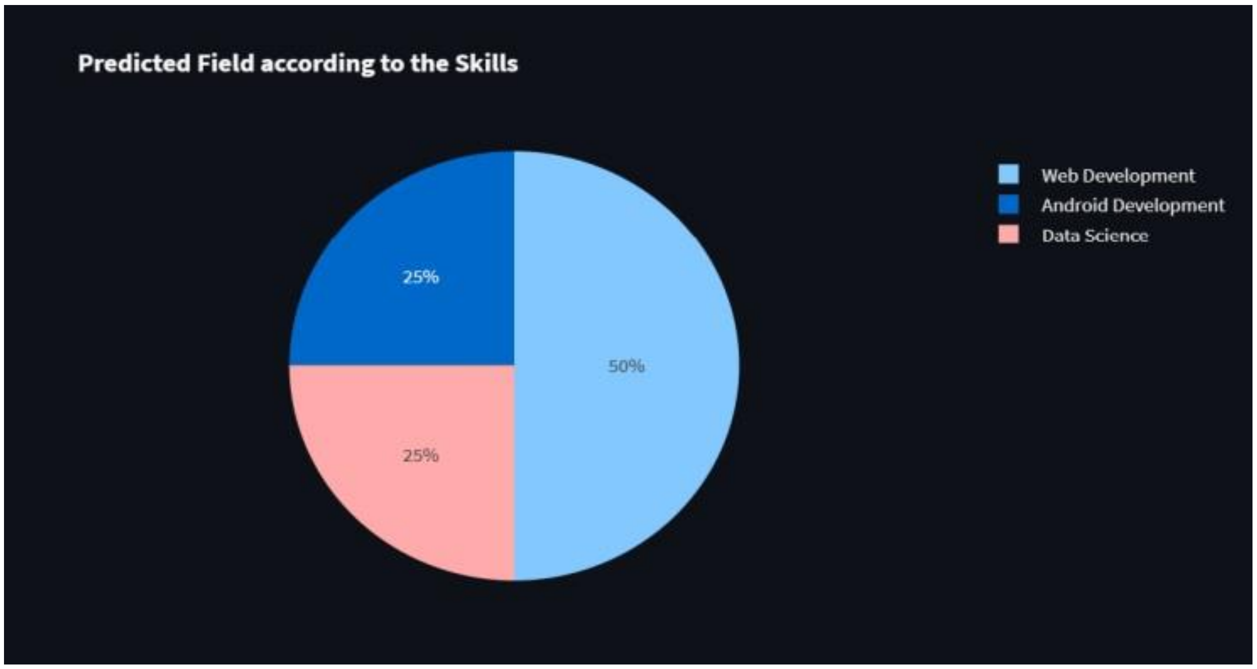
- Through this thorough research and analysis, organizations can develop a robust and effective resume parsing system that streamlines the recruitment process, improves efficiency, and meets the needs of both users and regulatory standards.

8 EVALUATION OF OUTCOMES/RESULTS:

- Resume parsing system using natural language processing (NLP) is essential for assessing its effectiveness and impact.

- This evaluation encompasses several key aspects, including the performance of NLP models in accurately extracting information from resumes, user feedback and satisfaction regarding system usability and effectiveness, efficiency gains and time savings achieved through automation, the system's impact on the overall recruitment process including metrics like time-to-hire and candidate quality, compliance with data privacy regulations and security standards, cost-benefit analysis to determine the system's return on investment (ROI), and identification of areas for improvement based on evaluation findings to ensure ongoing optimization and alignment with organizational goals.
- By systematically evaluating these aspects, organizations can gauge the success of their resume parsing system, make informed decisions for future enhancements, and continuously improve recruitment processes.

9 GRAPH/FINDINGS:



This shows the pictorial representation of applicants based on their skills.

10 BIBILIOGRAPHY

- Ronan Collobert, Jason Weston L´eon Bottou ,Michael Karlen, Koray Kavukcuoglu, Pavel Kuksa “Natural Language Processing

from Scratch” arxiv.

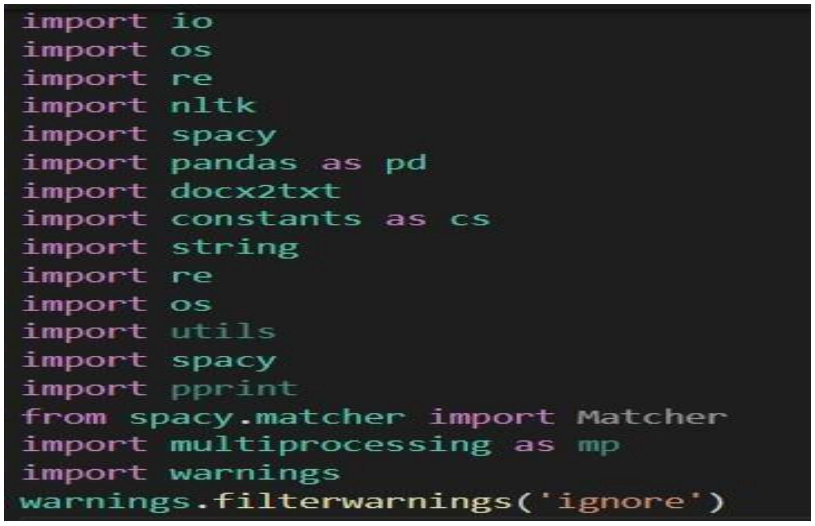
- Shicheng Zu, Xiulai Wang and Seth Darren “Resume Information Extraction with A Novel Text Block Segmentation Algorithm”.

- Chengguang Gan, Tatsunori Mori “A Few Shot
  Approach to Resume Information Extraction via Prompts”
- Agrawal, R., 2021. analyticsvidhya. Available at:
  https://www.analyticsvidhya.com/blog/2021/06/must-
  knowntechniquesfortextpreprocessinginnlp/#:~:text=Text%20prep
  rocessing%20is%20a%20method,text%20in%20a%2 0diff erent%20case. [Accessed 2022]
- Bhatia, V., Rawat, P., Kumar, A. & Shah, R. R., 2019. End-to-End Resume Parsing and Finding Candidates for a Job Description using BERT. arxiv.
- Chavan, J., 2020. medium. Available at:
  https://medium.com/@jeevanchavan143/nlptoke
  nizationstemming lemmatization-bag-of-
  wordstf-idf-pos-7650f83c60be [Accessed 2022].

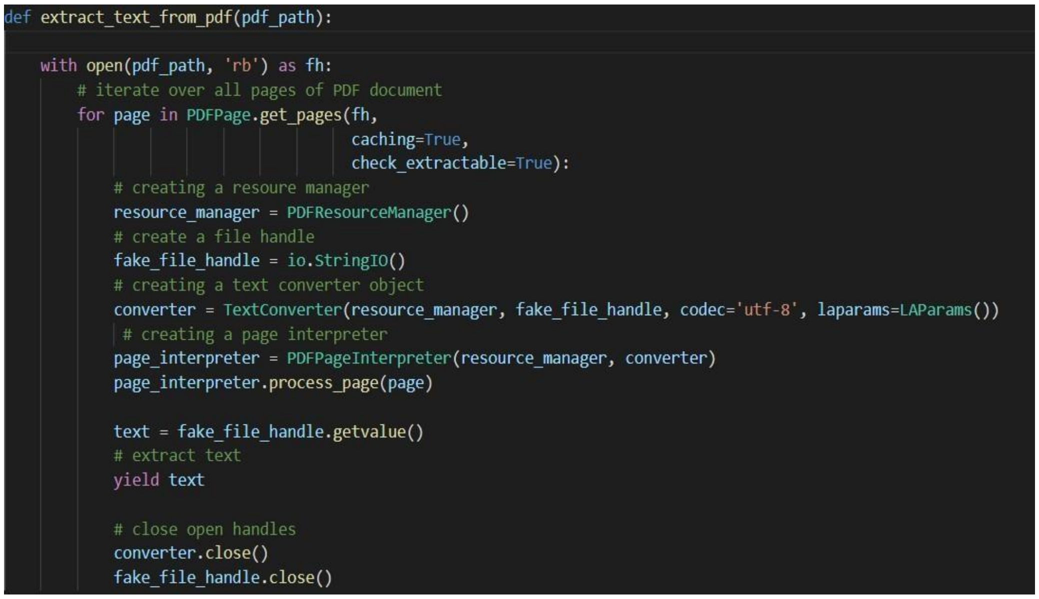
- Chen, J., Gao,. L. & Tang, Z., 2016. Information Extraction from Resume Documents in PDF. Research Gate. 64
- D., 2021. analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/06/t extpreprocessing-innlp-with-pythoncodes/ [Accessed 2022]. [9] Kopparapu, S.. K., 2015. Automatic Extraction of Usable Information from Unstructured Resumes to Aid Search.

ieeexplore.

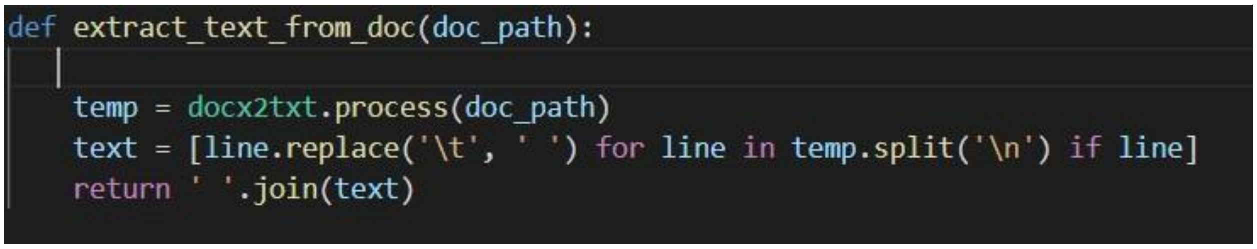
11 APPENDIX



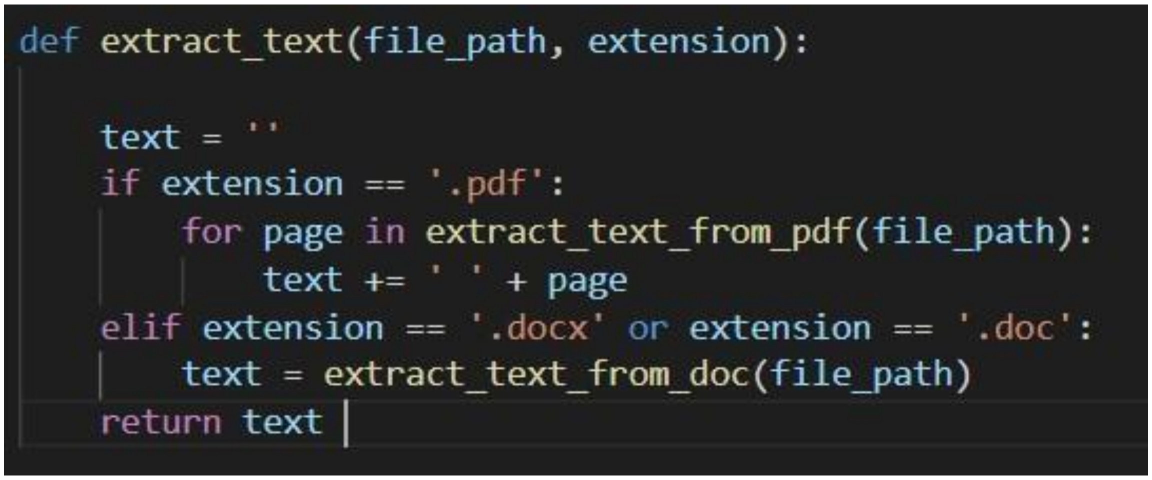
```
import io
import os
import re
import nltk
import spacy
import pandas as pd
import docx2txt
import constants as cs
import string
import re
import os
import utils
import spacy
import pprint
from spacy.matcher import Matcher
import multiprocessing as mp
import warnings
warnings.filterwarnings('ignore')
```



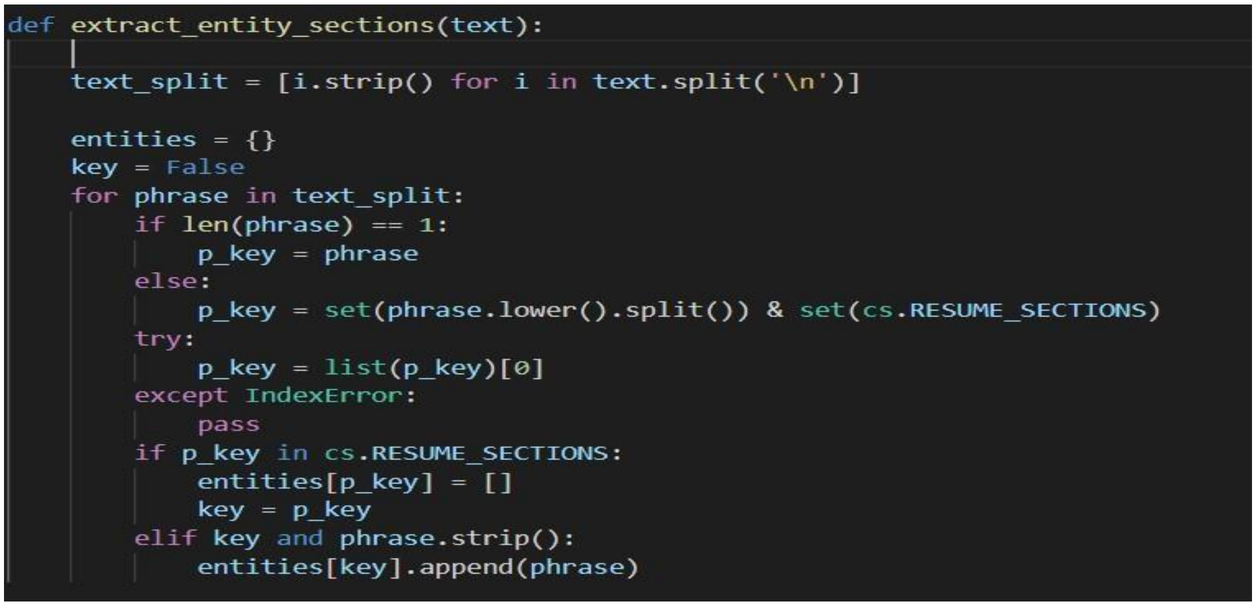
```python
def extract_text_from_pdf(pdf_path):

    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)
            page_interpreter.process_page(page)

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

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



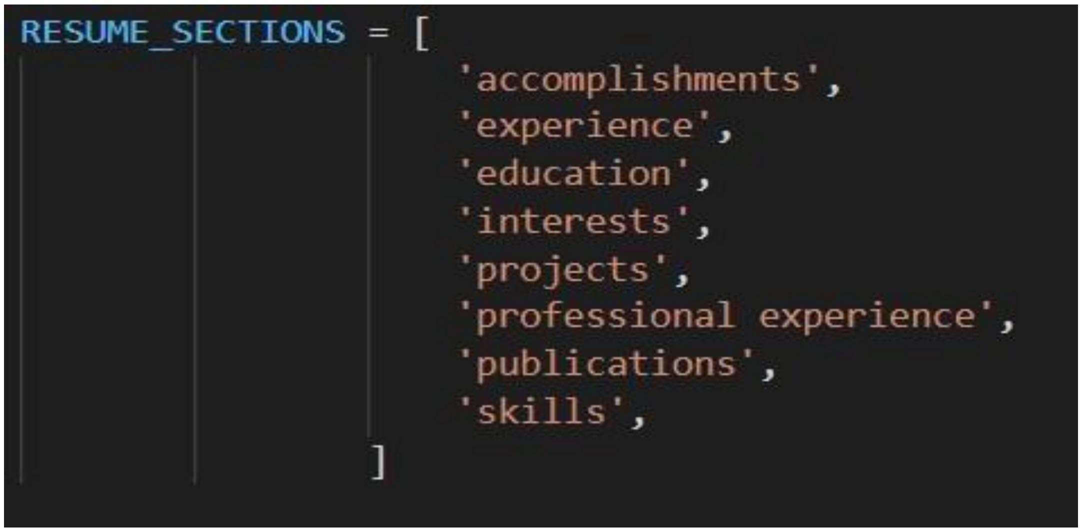
```python
def extract_text_from_doc(doc_path):

    temp = docx2txt.process(doc_path)
    text = [line.replace('\t', ' ') for line in temp.split('\n') if line]
    return ' '.join(text)
```



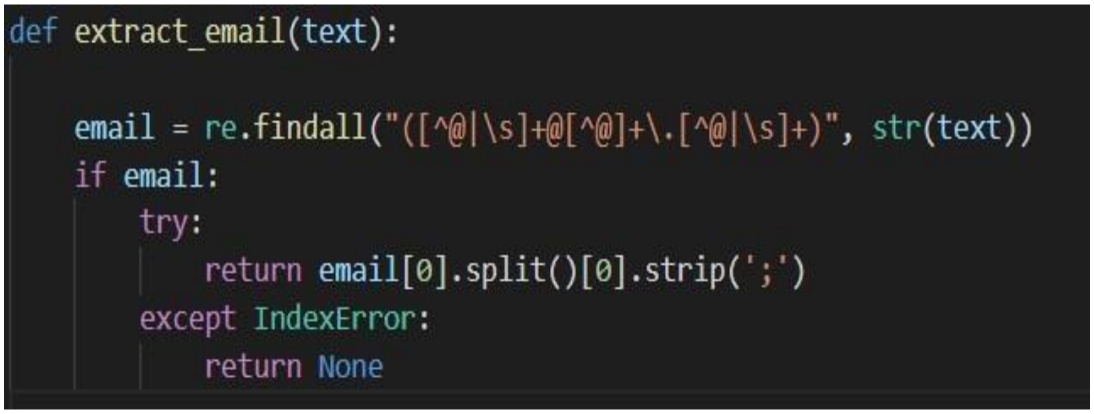
```python
def extract_text(file_path, extension):

    text = ''
    if extension == '.pdf':
        for page in extract_text_from_pdf(file_path):
            text += ' ' + page
    elif extension == '.docx' or extension == '.doc':
        text = extract_text_from_doc(file_path)
    return text
```



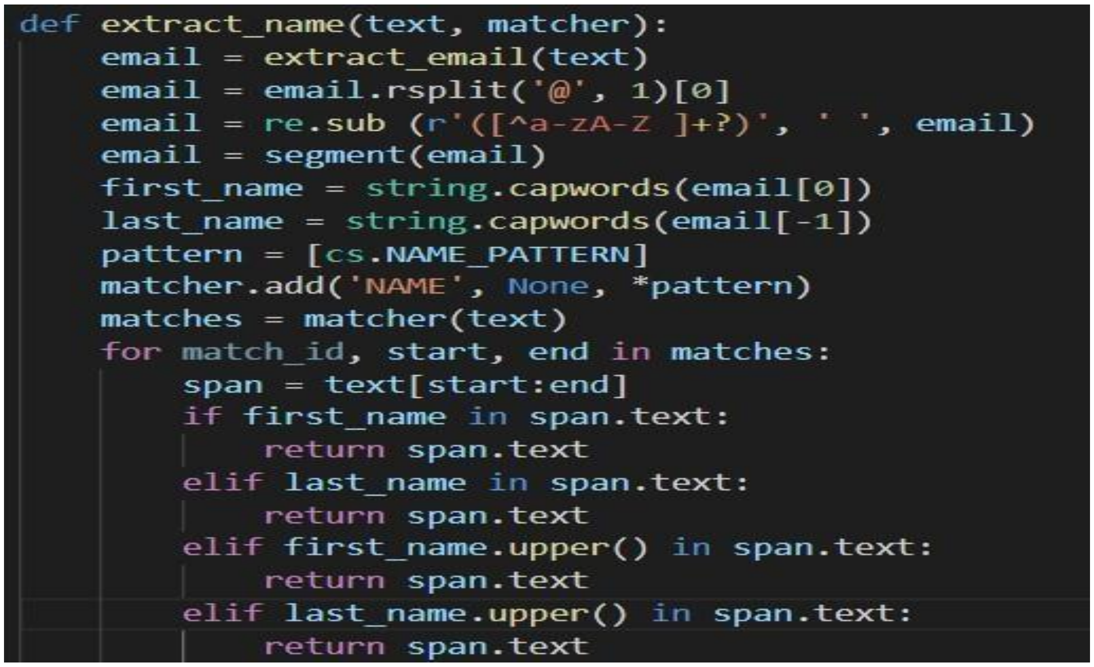
```python
def extract_entity_sections(text):

    text_split = [i.strip() for i in text.split('\n')]

    entities = {}
    key = False
    for phrase in text_split:
        if len(phrase) == 1:
            p_key = phrase
        else:
            p_key = set(phrase.lower().split()) & set(cs.RESUME_SECTIONS)
        try:
            p_key = list(p_key)[0]
        except IndexError:
            pass
        if p_key in cs.RESUME_SECTIONS:
            entities[p_key] = []
            key = p_key
        elif key and phrase.strip():
            entities[key].append(phrase)
```



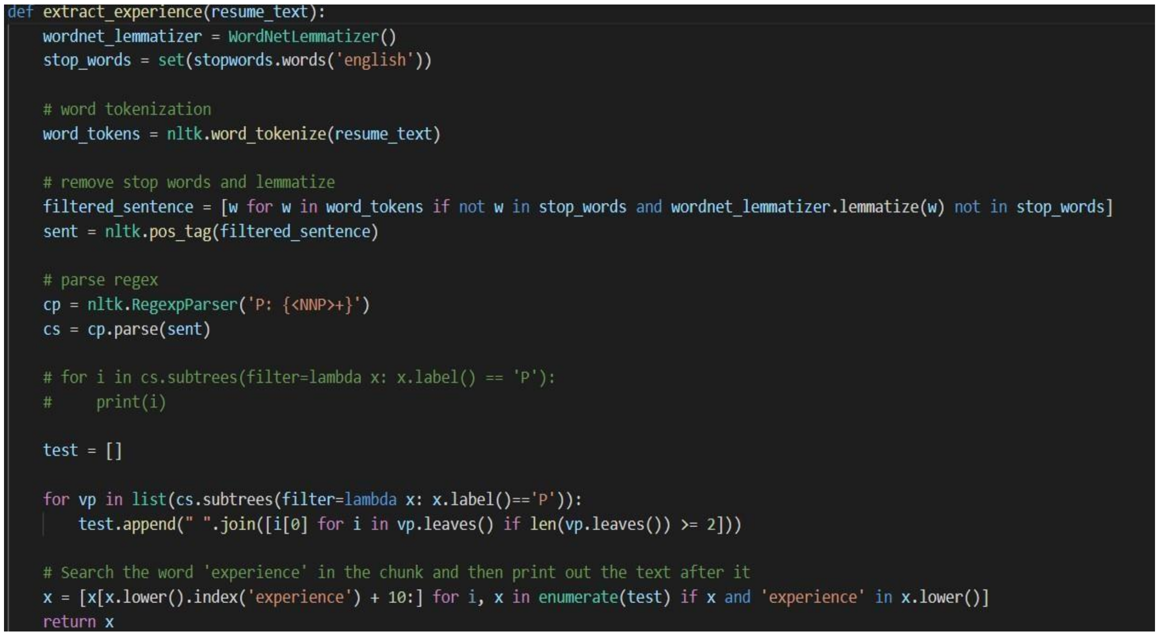
```python
RESUME_SECTIONS = [
                    'accomplishments',
                    'experience',
                    'education',
                    'interests',
                    'projects',
                    'professional experience',
                    'publications',
                    'skills',
                ]
```



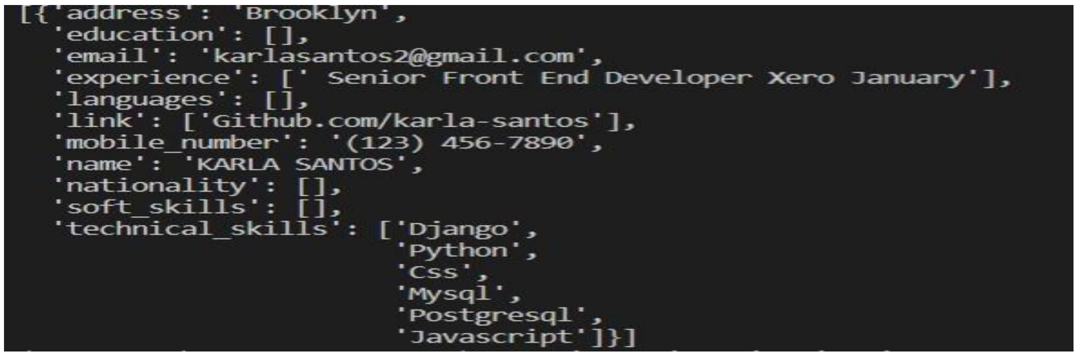
```python
def extract_email(text):

    email = re.findall("([^@|\s]+@[^@]+\.[^@|\s]+)", str(text))
    if email:
        try:
            return email[0].split()[0].strip(';')
        except IndexError:
            return None
```



```python
def extract_name(text, matcher):
    email = extract_email(text)
    email = email.rsplit('@', 1)[0]
    email = re.sub (r'([^a-zA-Z ]+?)', ' ', email)
    email = segment(email)
    first_name = string.capwords(email[0])
    last_name = string.capwords(email[-1])
    pattern = [cs.NAME_PATTERN]
    matcher.add('NAME', None, *pattern)
    matches = matcher(text)
    for match_id, start, end in matches:
        span = text[start:end]
        if first_name in span.text:
            return span.text
        elif last_name in span.text:
            return span.text
        elif first_name.upper() in span.text:
            return span.text
        elif last_name.upper() in span.text:
            return span.text
```



```python
def extract_experience(resume_text):
    wordnet_lemmatizer = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))

    # word tokenization
    word_tokens = nltk.word_tokenize(resume_text)

    # remove stop words and lemmatize
    filtered_sentence = [w for w in word_tokens if not w in stop_words and wordnet_lemmatizer.lemmatize(w) not in stop_words]
    sent = nltk.pos_tag(filtered_sentence)

    # parse regex
    cp = nltk.RegexpParser('P: {<NNP>+}')
    cs = cp.parse(sent)

    # for i in cs.subtrees(filter=lambda x: x.label() == 'P'):
    #     print(i)

    test = []

    for vp in list(cs.subtrees(filter=lambda x: x.label()=='P')):
        test.append(" ".join([i[0] for i in vp.leaves() if len(vp.leaves()) >= 2]))

    # Search the word 'experience' in the chunk and then print out the text after it
    x = [x[x.lower().index('experience') + 10:] for i, x in enumerate(test) if x and 'experience' in x.lower()]
    return x
```



```
[{'address': 'Brooklyn',
  'education': [],
  'email': 'karlasantos2@gmail.com',
  'experience': [' Senior Front End Developer Xero January'],
  'languages': [],
  'link': ['Github.com/karla-santos'],
  'mobile_number': '(123) 456-7890',
  'name': 'KARLA SANTOS',
  'nationality': [],
  'soft_skills': [],
  'technical_skills': ['Django',
                       'Python',
                       'Css',
                       'Mysql',
                       'Postgresql',
                       'Javascript']}]
```