**PROJECT REPORT**

**RESUME ANALYSIS**

Parth Bapat(L-04)

Siddhi Chavan(L-15)

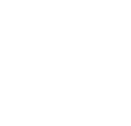
Ruta Deshpande (L-06)

Guided by: Prof. Milind Kamble

**Course: Natural Language Processing**

**2019-20 (Semester-I)**

**B. Tech**



Department of Electronics and Telecommunication Engineering

BRACT’S

Vishwakarma Institute of Technology, Pune- 411037

**INTRODUCTION:**

Corporate companies and recruitment agencies process numerous resumes daily. This is no task for humans. An automated intelligent system is required which can take out all the vital information from the unstructured resumes and transform all of them to a common structured format which can then be ranked for a specific job position. Parsed information include name, email address, social profiles, personal websites, years of work experience, years of education, education experiences, publications, certifications, volunteer experiences, keywords, etc. Resumes are difficult to parse. This is because they vary in types of information, their order, writing style, etc. Moreover, they can be written in various formats. Some of the common ones include '.txt', '.pdf', '.doc', '.docx', etc. To parse the data from different kinds of resumes effectively and efficiently, the model must not rely on the order or type of data.

**THEORY:**

Natural language processing (NLP) is a subfield of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages. While in resumes data formats that are used is not completely unstructured, it is still quite challenging to take them into structured format as there is no set-in stone rule for writing a resume. As a result, many possible ways of representing qualifications in a resume has been established so far such as chronological resume and functional resume. Beyond these two, there are many other formats and many people follow their own unique style to make their resume stand out from other ones. In the past, Resumes submitted by job seekers used to be manually analysed and judged by the employers. This method is still followed in the recent times. However, as the big companies often need to deal with hundreds of resumes each and every day, it has become very problematic and time consuming to handle such a big number of resumes one by one. The innovation in the field of Natural Language Processing along with Machine Learning has been really helpful in this case. The ability to understand unstructured written language and extract important information from it to teach the machine is exactly what is needed to analyse any written documents such as resume papers just like human being.

Tokenization:

Tokens are usually referred to as terms or words, but sometimes fabricating a type/token distinction is essential. A specimen of an array of characters in a document that is assembled as a helpful acceptable unit for processing is called a token. Whereas, the group of tokens which consists of same character sequence is called type.

POS Tagging:

In corpus linguistics, the procedure of marking up a text (corpus) which is analogous to a particular part of speech depending on both its definition and context (like how it is related to the adjoining words in a paragraph, sentence or phrase) is called part-of-speech tagging (POS tagging or POST). It is also known as grammatical tagging or word-category disambiguation.

Chunking:

Also known as shallow parsing, chunking is actually the recognition of parts of speech and short phrases. We can determine if the words are nouns, verbs, adjectives, etc by Parts of Speech tagging, but from this, we cannot get any clue about the sentence of phrase structure in the sentence. At times, some more information than parts of speech of words are useful, but the full parse tree that we would get from parsing is not needed.

**METHODOLOGY:**

1. Resumes do not have a fixed file format, and hence they can be in any file format such as .pdf or .doc or .docx. So, the main challenge is to read the resume and convert it to plain text. For these two Python modules are used: pdfminer and doc2text. These modules help extract text from pdf, .doc and .docx. file formats.
2. For extracting names from resumes a tool called [spaCy](https://spacy.io/) has been used. Spacy is a Industrial-Strength Natural Language Processing module used for text and language processing. It comes with pre-trained models for tagging, parsing and entity recognition.
3. For extracting phone numbers, we will be making use of regular expressions. Phone numbers also have multiple forms such as (+91) 1234567890 or +911234567890 or +91 123 456 7890 or +91 1234567890. Hence, we need to define a generic regular expression that can match all similar combinations of phone numbers.
4. For extracting Email IDs from resume, we can use a similar approach that we used for extracting mobile numbers. Email IDs have a fixed form i.e. an alphanumeric string should follow a @ symbol, again followed by a string, followed by a. (dot) and string at the end. Regular expressions are used to extract such expression from text.
5. Skills can be extracted using a technique called  [tokenization](https://textminingonline.com/dive-into-nltk-part-ii-sentence-tokenize-and-word-tokenize). Tokenization simply is breaking down of text into paragraphs, paragraphs into sentences, sentences into words. Hence, there are two major techniques of tokenization: Sentence Tokenization and Word Tokenization. Before implementing tokenization, a dataset against which the skills in a particular resume are compared. For this a (.csv) file with desired skillsets is created. For example, if a recruiter is looking for a candidate with skills including NLP, ML, AI then a csv file with these contents should be made.
6. The educational details like degree and the year of passing are extracted by comparing the resume text with a specified list of qualifications.

**PYTHON CODE:**

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 io

import spacy

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()

import spacy

from spacy.matcher import Matcher

# load pre-trained model

nlp = spacy.load('en\_core\_web\_sm')

# initialize matcher with a vocab

matcher = Matcher(nlp.vocab)

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('NAME', None, pattern)

matches = matcher(nlp\_text)

for match\_id, start, end in matches:

span = nlp\_text[start:end]

# print(span.text)

return span.text

import re

def extract\_mobile\_number(text):

phone = 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+))?'), text)

if phone:

number = ''.join(phone[0])

if len(number) > 10:

return '+' + number

else:

return number

def extract\_email(email):

email = re.findall("([^@|\s]+@[^@]+\.[^@|\s]+)", email)

if email:

try:

return email[0].split()[0].strip(';')

except IndexError:

return None

import pandas as pd

def extract\_skills(resume\_text):

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("skills.csv")

# extract values

skills = list(data.columns.values)

skillset = []

# check for one-grams (example: python)

for token in tokens:

if token.lower() in skills:

skillset.append(token)

# check for bi-grams and tri-grams (example: machine learning)

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])]

from nltk.corpus import stopwords

# Grad all general stop words

STOPWORDS = set(stopwords.words('english'))

# Education Degrees

EDUCATION = [

'BE','B.E.', 'B.E', 'BS', 'B.S',

'ME', 'M.E', 'M.E.', 'MS', 'M.S',

'BTECH', 'B.TECH', 'M.TECH', 'MTECH',

'SSC', 'HSC', 'CBSE', 'ICSE', 'X', 'XII'

]

def extract\_education(resume\_text):

nlp\_text = nlp(resume\_text)

# Sentence Tokenizer

nlp\_text = [sent.string.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 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{2})))'), edu[key])

if year:

education.append((key, ''.join(year[0])))

else:

education.append(key)

return education

# file\_path = 'Resume1.pdf'

file\_path = 'Resume2.pdf'

text = ''

# calling above function and extracting text

for page in extract\_text\_from\_pdf(file\_path):

text += ' ' + page

# print(text)

print('--------------Extracted information-----------------')

print('Name : ' + extract\_name(text))

print('Mobile No. : ' + extract\_mobile\_number(text))

print('Email ID : ' + extract\_email(text))

print('Education :\n', extract\_education(text))

print('Skills :\n', extract\_skills(text))

**CONCLUSION:**

Thus, vital information from the resume can be extracted using the above technique. This saves times and increases efficiency and creates an unbiased system for analysis.