|  |  |
| --- | --- |
| #Resume Phrase Matcher code | |
|  |  |
|  |  |
|  | #importing all required libraries |
|  |  |
|  | import PyPDF2 |
|  | import os |
|  | from os import listdir |
|  | from os.path import isfile, join |
|  | from io import StringIO |
|  | import pandas as pd |
|  | from collections import Counter |
|  | import en\_core\_web\_sm |
|  | nlp = en\_core\_web\_sm.load() |
|  | from spacy.matcher import PhraseMatcher |
|  |  |
|  | #Function to read resumes from the folder one by one |
|  | mypath='D:/NLP\_Resume/Candidate Resume' #enter your path here where you saved the resumes |
|  | onlyfiles = [os.path.join(mypath, f) for f in os.listdir(mypath) if os.path.isfile(os.path.join(mypath, f))] |
|  |  |
|  | def pdfextract(file): |
|  | fileReader = PyPDF2.PdfFileReader(open(file,'rb')) |
|  | countpage = fileReader.getNumPages() |
|  | count = 0 |
|  | text = [] |
|  | while count < countpage: |
|  | pageObj = fileReader.getPage(count) |
|  | count +=1 |
|  | t = pageObj.extractText() |
|  | print (t) |
|  | text.append(t) |
|  | return text |
|  |  |
|  | #function to read resume ends |
|  |  |
|  |  |
|  | #function that does phrase matching and builds a candidate profile |
|  | def create\_profile(file): |
|  | text = pdfextract(file) |
|  | text = str(text) |
|  | text = text.replace("\\n", "") |
|  | text = text.lower() |
|  | #below is the csv where we have all the keywords, you can customize your own |
|  | keyword\_dict = pd.read\_csv('D:/NLP\_Resume/resume/template\_new.csv') |
|  | stats\_words = [nlp(text) for text in keyword\_dict['Statistics'].dropna(axis = 0)] |
|  | NLP\_words = [nlp(text) for text in keyword\_dict['NLP'].dropna(axis = 0)] |
|  | ML\_words = [nlp(text) for text in keyword\_dict['Machine Learning'].dropna(axis = 0)] |
|  | DL\_words = [nlp(text) for text in keyword\_dict['Deep Learning'].dropna(axis = 0)] |
|  | R\_words = [nlp(text) for text in keyword\_dict['R Language'].dropna(axis = 0)] |
|  | python\_words = [nlp(text) for text in keyword\_dict['Python Language'].dropna(axis = 0)] |
|  | Data\_Engineering\_words = [nlp(text) for text in keyword\_dict['Data Engineering'].dropna(axis = 0)] |
|  |  |
|  | matcher = PhraseMatcher(nlp.vocab) |
|  | matcher.add('Stats', None, \*stats\_words) |
|  | matcher.add('NLP', None, \*NLP\_words) |
|  | matcher.add('ML', None, \*ML\_words) |
|  | matcher.add('DL', None, \*DL\_words) |
|  | matcher.add('R', None, \*R\_words) |
|  | matcher.add('Python', None, \*python\_words) |
|  | matcher.add('DE', None, \*Data\_Engineering\_words) |
|  | doc = nlp(text) |
|  |  |
|  | d = [] |
|  | matches = matcher(doc) |
|  | for match\_id, start, end in matches: |
|  | rule\_id = nlp.vocab.strings[match\_id]  # get the unicode ID, i.e. 'COLOR' |
|  | span = doc[start : end]  # get the matched slice of the doc |
|  | d.append((rule\_id, span.text)) |
|  | keywords = "\n".join(f'{i[0]} {i[1]} ({j})' for i,j in Counter(d).items()) |
|  |  |
|  | ## convertimg string of keywords to dataframe |
|  | df = pd.read\_csv(StringIO(keywords),names = ['Keywords\_List']) |
|  | df1 = pd.DataFrame(df.Keywords\_List.str.split(' ',1).tolist(),columns = ['Subject','Keyword']) |
|  | df2 = pd.DataFrame(df1.Keyword.str.split('(',1).tolist(),columns = ['Keyword', 'Count']) |
|  | df3 = pd.concat([df1['Subject'],df2['Keyword'], df2['Count']], axis =1) |
|  | df3['Count'] = df3['Count'].apply(lambda x: x.rstrip(")")) |
|  |  |
|  | base = os.path.basename(file) |
|  | filename = os.path.splitext(base)[0] |
|  |  |
|  | name = filename.split('\_') |
|  | name2 = name[0] |
|  | name2 = name2.lower() |
|  | ## converting str to dataframe |
|  | name3 = pd.read\_csv(StringIO(name2),names = ['Candidate Name']) |
|  |  |
|  | dataf = pd.concat([name3['Candidate Name'], df3['Subject'], df3['Keyword'], df3['Count']], axis = 1) |
|  | dataf['Candidate Name'].fillna(dataf['Candidate Name'].iloc[0], inplace = True) |
|  |  |
|  | return(dataf) |
|  |  |
|  | #function ends |
|  |  |
|  | #code to execute/call the above functions |
|  |  |
|  | final\_database=pd.DataFrame() |
|  | i = 0 |
|  | while i < len(onlyfiles): |
|  | file = onlyfiles[i] |
|  | dat = create\_profile(file) |
|  | final\_database = final\_database.append(dat) |
|  | i +=1 |
|  | print(final\_database) |
|  |  |
|  |  |
|  | #code to count words under each category and visulaize it through Matplotlib |
|  |  |
|  | final\_database2 = final\_database['Keyword'].groupby([final\_database['Candidate Name'], final\_database['Subject']]).count().unstack() |
|  | final\_database2.reset\_index(inplace = True) |
|  | final\_database2.fillna(0,inplace=True) |
|  | new\_data = final\_database2.iloc[:,1:] |
|  | new\_data.index = final\_database2['Candidate Name'] |
|  | #execute the below line if you want to see the candidate profile in a csv format |
|  | #sample2=new\_data.to\_csv('sample.csv') |
|  | import matplotlib.pyplot as plt |
|  | plt.rcParams.update({'font.size': 10}) |
|  | ax = new\_data.plot.barh(title="Resume keywords by category", legend=False, figsize=(25,7), stacked=True) |
|  | labels = [] |
|  | for j in new\_data.columns: |
|  | for i in new\_data.index: |
|  | label = str(j)+": " + str(new\_data.loc[i][j]) |
|  | labels.append(label) |
|  | patches = ax.patches |
|  | for label, rect in zip(labels, patches): |
|  | width = rect.get\_width() |
|  | if width > 0: |
|  | x = rect.get\_x() |
|  | y = rect.get\_y() |
|  | height = rect.get\_height() |
|  | ax.text(x + width/2., y + height/2., label, ha='center', va='center') |
|  | plt.show() |