**RESUME SHORTLISTER**

**Overview**

We have made a Resume Shortlister Project, an AI application for screening and shortlisting Resumes and candidatures of individuals using Natural Language Processing and heuristics to calculate the final score of each candidate. The candidates with the highest scores can then be called for further interview rounds.

**Approach**

1. We will use Natural Language Processing to solve the problem at

hand. First, we created a CSV file, which acts as a dictionary and has the various skill sets categorized. The skills have been categorized into a total of 8 fields. They are:

● Statistics

● Machine Learning

● Deep Learning

● R language

● Python Language

● Natural Language Processing

● Data Engineering

● Web Development

2. These fields are then further categorized according to the various

terms that appear in their respective domain. So for example, there are terms like keras, tensorflow which are grouped under Deep learning whereas terms like spacy, lda are added in the NLP category.

This is the CSV file containing all the skill sets.

3. After that, an NLP Algorithm is applied that goes through the resumes one by

one, parses them and looks for the words mentioned in the dictionary. 4. The algorithm then counts the occurrences of the words under each category

and then finally represents the above count in a visual way by plotting a graph and a showing a table for each candidate. 5. A graph is plotted which represents the candidates on the y- axis and their skill

counts on the x-axis. 6. A scoring mechanism is used which helps us to filter out the candidates for the

profile we have made the application for, by giving more weight to that particular field and assigning less weight to the others. Eg. For shortlisting candidates for an AI profile, more weightage is given to fields like Deep Learning, NLP, Machine learning compared to Statistics and Web Development.

**Technology used Language: Python Visualization : Matplotlib**

The following dependencies were used in the program :

1. PyPDF2 2. Numpy 3. Pandas 4. Spacy 5. En-core-web-lg

**Benefits**

**1. Automatically reading the Resume -** If we were to manually open each and

every resume, it would take a lot of time. The code saves us from this scenario by automatically opening each and every resume and parsing the content.

***# Function to read resumes from the folder one by one*** *mypath = '/home/karanpal/NLP\_Resume/candidateResume' 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*

**2. Phrase Matching -** Instead of manually searching for whether a candidate has

the desired skills or not, the code hunts for the keywords, keeps a count of their occurrence and categorizes them.

*matcher = PhraseMatcher(nlp.vocab) matcher.add('Stats', None, \*stats\_words)*

**3. Data Visualization -** It is the most important aspect as it helps us to speed up

the process . The graph plotted by the algorithm helps us to decide which candidate has more keywords under each category, thus implying that they may be good in that domain and thus helps to make the selection procedure much faster.

*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()*

**Results** The code helps us to quickly shortlist the resumes with a few seconds and thus saves a lot of time.