[RFC] Automating Resume Screening

Created: July 13, 2021
Current Version: 1.0.0
Target Version: 1.1.0
PRD: Link to PRD if applicable

Status: WIP | In-Review | Approved | Obsolete

Owner: bawa.mandeep0001@gmail.com
Contributors: bawa.mandeep0001@gmail.com

Automating resume screening process for recruitment team using Machine learning models.

Background

The cost of hiring an employee is arguably one of the most expensive parts of running a business. Hiring a right candidate for the job, especially in recruitment of a startup, requires a lot of skills, patience, time and money. According to a blog on toggl.com, small business owners spend around 40% of their working hours on tasks that do not generate income, such as hiring. Even big companies need a dedicated HR team for the hiring process but that too comes at a price.

On one hand we need a large number of applicants to apply for the job, on the other we need a smaller number of applicants to move forward to the next process. Dealing with the large number of applications with a small recruitment team would be a challenge for a startup and established companies.

Proposal

One option in this circumstance is to have a software implemented solution that can filter the top applicants from the group of collected resumes for the job position, thus easing the task for the recruitment team.

For this, it is necessary to parse the resume and have some model which can rate the resume given its data.

Dataset

Open source kaggle dataset on resume was used for this problem. Dataset has 228 files in docx format.

Main ingredients of the standard resume

- Personal Details
    - Name
    - Email
    - Phone No.
    - Country
- Education
    - Degree
    - University
- Past companies
- Past designations
- Experience
- Skills set
- Hobbies

Below are the steps to be followed to get to final model

1. Parsing Resume

It is very important to have a good parser which can capture most of the information from the docx format resume files. For this, we have used the pip's latest library resume-parser which is accompanied with thousands of university names, skills and designations. On the backend, this library uses nltk, tika to extract information from the resume.

This library is still incomplete as it lacks certain checks while extracting experience from the resume. To solve this issue, I need to add certain try-except conditions and also a digits extractor function which helps to calculate experience for almost all the resumes.

We got json files in directory Resume-json/* after parsing resumes using main.py

Note: Without the addition of this function and manually added checks, rating would not work properly

2. Data Preprocessing

After parsing the files with resume parser, json files were preprocessed to give clean data.

This step required each json file to go through stemming, tokenization, converting to lowercase. Removal of stop words, punctuation was not required as we were not using

text as our direct input for our model. Json files were combined to one resume_data.csv file

3. Feature Extraction

This is the crucial step for our model building process.

I. Experience

Doing some statistical analysis on the experience, I found that certain values in the experience column were having very large values which may be due to error while extracting experience. To get rid of these values I used z-score < 2 to get relevant data.

II. Designation ranking

After collecting designation from all the rows, I rated 369 designation on a scale of 1 to 5.

Rating done as

5 - Senior positions
4 - Project managers
3 - Developer positions
2 - Relevant skill
1 - Normal skill

After rating all the designations, a binary file (desig_rating.pkl) was generated. This binary file was used to get a total designation rank for a user.

III. Designation count

This is the number of designations extracted by resume-parser.

IV. Skill count

This is the number of skills extracted by resume-parser

Note: Designation ranking can be based on the job requirements giving required designation more weightage than others. Similarly skills can also be rated according to job position.

More Features

Similar ranking can be given to universities, degrees and past companies to get more features. But considering the time bounds, the model was limited to four features.

Note: One hot encoding the skills could give us a large number of features, but as our dataset is small, this would result in a curse of dimensionality and therefore overfitting.

4. Generating Labels

To train a model we require a training dataset with labels for supervised training. So far, we have got only features.

Dataframe was sorted with primary index as experience and secondary index as designation rating, designation count and skill count.

A new column percentile rank was generated for this sorted data. Further scaling and ceiling was done to get labels in the 1-5 range. Now these will act as the final rating for a resume.

5. Train Test Split

After getting the labels train test split was done to get 80% training data and 20% test data.

6. Models

We started with very basic models like linear regression as the data would be linearly separable based on our feature extraction.

Same was evident from our results from feature importance values

```
Feature 0, Score 0.26974
Feature 1, Score 0.00280
Feature 2, Score -0.02863
Feature 3, Score -0.00106
```

Random forest regression and many weak learners were also tried for this data. But due to the small amount of data, they didn't perform better than the linear regression model.

Accuracy on train data: 87%
Accuracy on test data: 75%

Results on the test data

	precision	recall	f1-score	support
1.0	1.00	0.25	0.40	8
2.0	0.25	0.50	0.33	4
3.0	0.78	1.00	0.88	7
4.0	1.00	1.00	1.00	4
5.0	1.00	1.00	1.00	10
accuracy			0.76	33
macro avg	0.81	0.75	0.72	33
weighted avg	0.86	0.76	0.75	33

Implementation

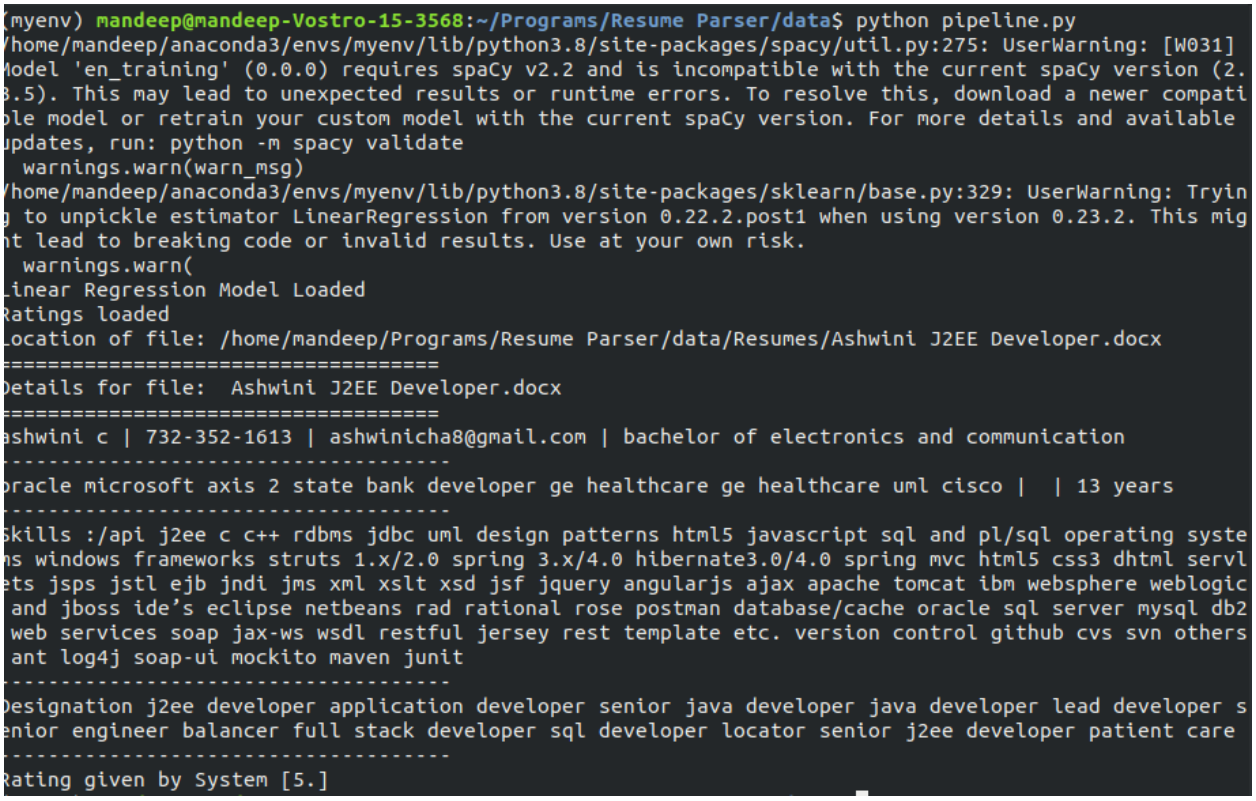
Prerequisite libraries

- tika
- nltk
- resume-parser
- numpy
- pandas
- json
- os

Replace the file ‘lib/python/site-packages/resume_parser/resumeparsing.py’ by resumeparse.py provided.

> Run the file pipeline.py
> enter the path to resume

Output (resume specific)



Link for Colab
https://colab.research.google.com/drive/1TtPbRcU4-Qtc-BLVetYWdO5EhgMZGNe8?usp=sharing