Resume Rating Tool

Summary: Help Recruiters to shortlist a resume
Created: 13th July, 2021
Owner: Anil Gujiri

The objective of this project is to design a resume rating tool that should be able to "rate" an applicant based on the contents of the resume primarily his skills on the scale of 1 -5, 5 being the best.

Background

Before a resume is crafted, usually a job description will be posted, and then the applicant will (likely) tailor their resume to suit the job description. Depending on the size of the company, the HR will then filter the resume based on keywords that they expect to see, with keywords being either general to the company or provided by the recruitment evaluators (typically the project team). Our approach uses keyword matching, where a set of fixed keywords are matched with the content of the resume. Some of the common keyword for the role of "Data Scientist" can be "Machine Learning, Deep Learning, Python, Regression, Classification, Clustering" etc.

Problem

Below is the list of problems occurred when building the project:

1. Not enough publicly available dataset
2. Parameters on which the resume is to be screened is not defined
3. Number of features are way more than the data itself as the skills for various profiles combined are very high
4. High chances of overfitting

Requirements and Phases

Phases	Steps
Phase 1: Collecting of dataset	Run a UI path script to fetch Candidate Profile in the form of pdf from LinkedIn
	Publicly available data set from Kaggle
Phase 2: Defining the Rating parameters	Defining the various keywords based on the job profile
Phase 3: Building and testing our model	Multiclass classification to rate a resume on the scale of 1 – 5

Phase 1: Collection of data

This can be divided into 2 parts:
1. Available dataset on Kaggle
2. Churning resumes from LinkedIn

Available dataset on Kaggle:
https://www.kaggle.com/gauravduttakiit/resume-dataset

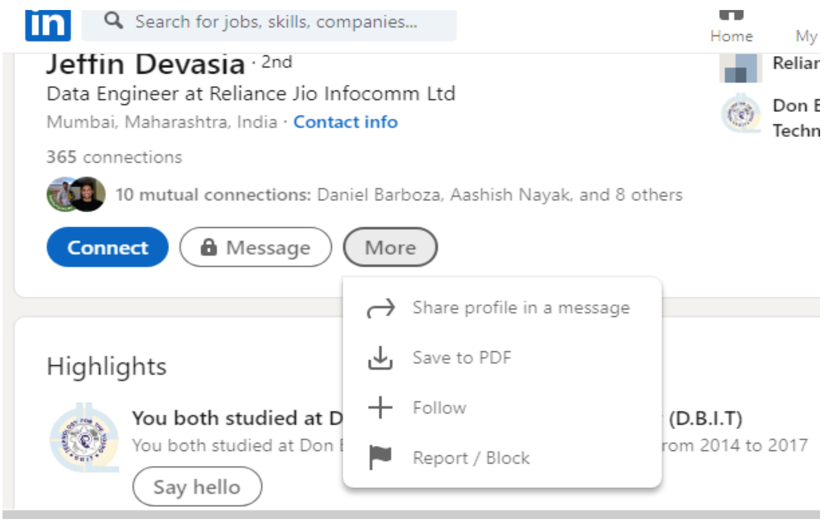
	Category	Resume
0	Data Science	Skills * Programming Languages: Python (pandas...
1	Data Science	Education Details \r\nMay 2013 to May 2017 B.E...
2	Data Science	Areas of Interest Deep Learning, Control Syste...
3	Data Science	Skills â¢ R â¢ Python â¢ SAP HANA â¢ Table...
4	Data Science	Education Details \r\n MCA YMCAUST, Faridab...

There are only two columns we have in the data. Below is the definition of each column.

Category: Type of Job Resume fits for.
Resume: Resume of candidates

This is a straight forward data available in the csv format.

Churning the resume using RPA from LinkedIn:



in
Search for jobs, skills, companies...
Home
My
Jeffin Devasia · 2nd
Data Engineer at Reliance Jio Infocomm Ltd
Mumbai, Maharashtra, India · Contact info
365 connections
10 mutual connections: Daniel Barboza, Aashish Nayak, and 8 others
Connect
Message
More
Share profile in a message
Save to PDF
Follow
Report / Block
Highlights
You both studied at D
You both studied at Don
(D.B.I.T)
rom 2014 to 2017
Say hello
Reliar
Don E
Techn

We needed **UIPATH** which is an RPA tool to do the task of automating our download from LinkedIn
We need to read our data from this pdf using a pre-trained general Named Entity Recognition (NER) model from SpaCy was used to perform entity detection. We used **tika** to parse our data and convert it into a Spacy doc. We also used docx to read word file resume. After the docs are parsed, I made a whole corpus out of each resume, keeping only those words in the corpus which matches my predefined keywords in the skills.txt files. Here we limited our skill set to just 50 to avoid **curse of dimensionality**. Both the datasets were combined and the total dataset were having 1000 records so **Oversampling** was done to increase our size of data set. This concludes our data preparation. Code can be found in **Data_Preparation.ipynb.**

Phase 2: Defining the Rating parameters

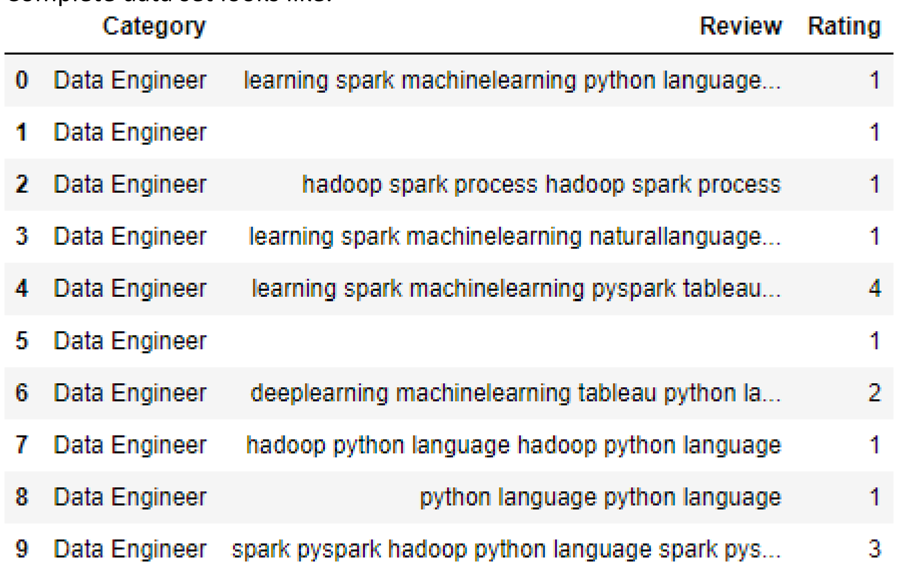
Here a simple approach of rating the resume more which has mor e umber of keywords is used. Less number of keywords, poor is the rating of the profile.

Keywords can be:

1. Skills of the Candidate
2. Degree he is holding
3. Total work Experience

For simplicity, we have only skills in our keyword corpus, but there is no restriction on training our model for the other two.

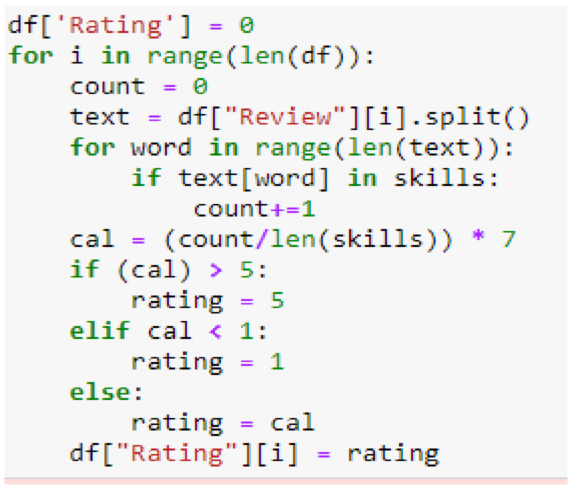
Complete data set looks like:



	Category	Review	Rating
0	Data Engineer	learning spark machinelearning python language...	1
1	Data Engineer		1
2	Data Engineer	hadoop spark process hadoop spark process	1
3	Data Engineer	learning spark machinelearning naturallanguage...	1
4	Data Engineer	learning spark machinelearning pyspark tableau...	4
5	Data Engineer		1
6	Data Engineer	deeplearning machinelearning tableau python la...	2
7	Data Engineer	hadoop python language hadoop python language	1
8	Data Engineer	python language python language	1
9	Data Engineer	spark pyspark hadoop python language spark pys...	3

We then converted our Category and Review in 1's and 0's so that it can be provided to our machine learning model. In general scenario TFIDF vectorizer would do a great done as we are not only considering for most frequent words but also the rare word. This is because our data corpus will have lots of feature and it will be very difficult to find most of them in a single resume. This case is perfectly handled by TFIDF.
For our case as we have only 50 features so we can consider BOW.

Code :



```
df['Rating'] = 0
for i in range(len(df)):
    count = 0
    text = df["Review"][i].split()
    for word in range(len(text)):
        if text[word] in skills:
            count+=1
    cal = (count/len(skills)) * 7
    if (cal) > 5:
        rating = 5
    elif cal < 1:
        rating = 1
    else:
        rating = cal
    df["Rating"][i] = rating
```

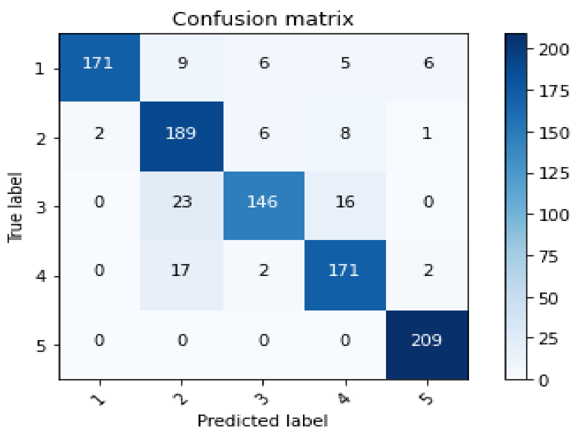
Phase 3: Model Training

This is a simple use case of multiclass classification where our target outputs are 1,2,3,4,5. % being the best. Generally, for such cases a Bayesian model or a simple logistic Regression will work very well.

1. We have use Naïve Bayes as the base model and selected the alpha after cross validation.

```
Alpha: 0.0, Score : 0.8825757575757576
Alpha: 0.1, Score : 0.8775252525252525
Alpha: 0.2, Score : 0.8737373737373737
Alpha: 0.30000000000000004, Score : 0.8712121212121212
Alpha: 0.4, Score : 0.8712121212121212
Alpha: 0.5, Score : 0.8712121212121212
Alpha: 0.6000000000000001, Score : 0.8699494949494949
Alpha: 0.7000000000000001, Score : 0.8699494949494949
Alpha: 0.8, Score : 0.8699494949494949
Alpha: 0.9, Score : 0.8699494949494949
```

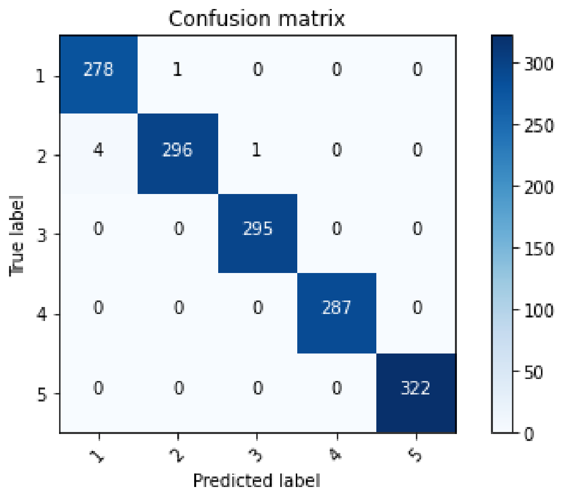
```
accuracy:   0.896
Confusion matrix, without normalization
```



As we can see just by using the naïve bayes we are getting an accuracy equal to 90%. Also our cross validation score is around 87% so it seems not overfitted.

2. Model training using Logistics regression with hyper parameter tuning:

```
Confusion matrix, without normalization
```



```python
print(best_model.best_estimator_)
print("The mean accuracy of the model is:",best_model.score(X_test,y_test))
```

```
Pipeline(steps=[('classifier',
                 LogisticRegression(C=30, max_iter=1000, n_jobs=1))])
The mean accuracy of the model is: 0.9959568733153639
```

Here the accuracy is very high because we have trained on very less features. But as the features will increase, around 200 features, accuracy using the same approach will be between 90 – 95 %.

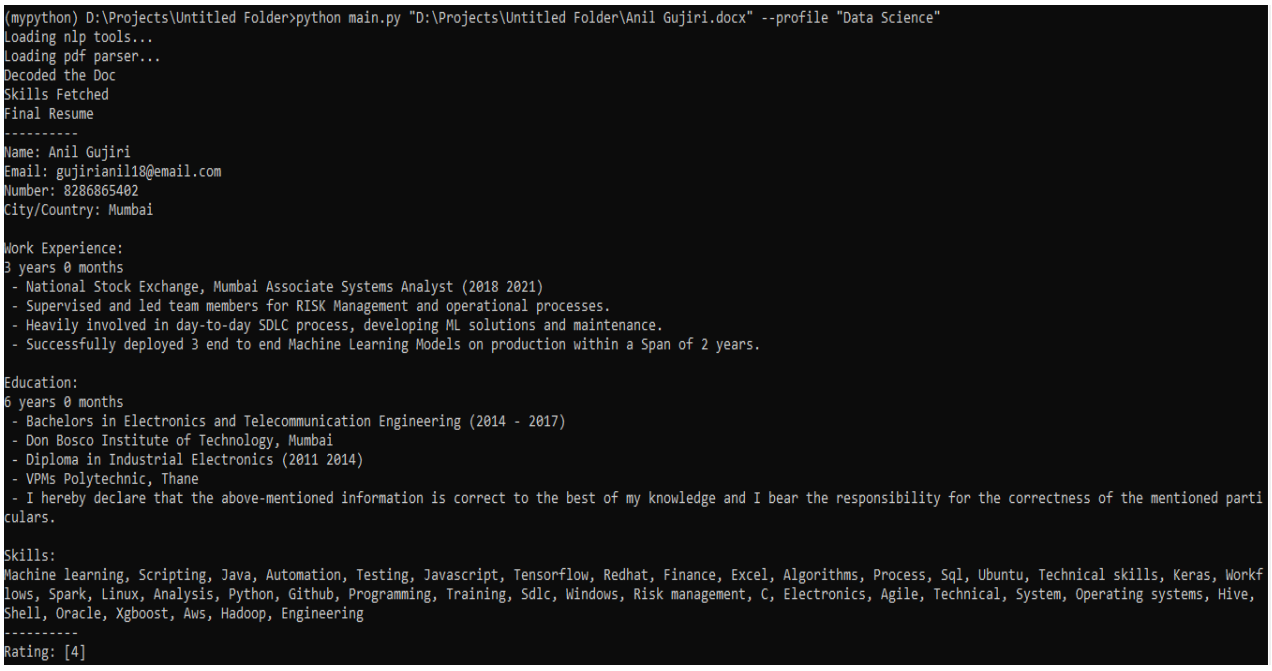
Information Extraction Implementation

For information extraction, the rules-based approach does pretty well for standard resumes, but suffers from low precision on sophisticatedly-designed resumes (irrelevant info captured). False positives are minimized by first detecting category boxes (Like "Work Experience" and "Education") and then performing relevant extractions.

Problems faced

- A Named Entity Recognition (NER) model has to be trained on a large dataset containing many different CVs well-labelled for it to generalize well.
- Since labelling takes a lot of time, and since there are no well pre-trained NER models out there, non-AI methods are used
- Some CVs have very complicated document formatting (very hard to parse). Eg .Tables and photos.
- Some people have names that SpaCy's NER model has not learnt before, so it cannot detect his/her name
- Some CVs have company names that are very different from modern companies e.g. (Usaha Tegas Sdn Bhd), thus the NER model cannot detect it
- Using a non-AI approach, it's not possible to detect anything if the text converted is screwed up e.g. all the titles are all the way below the text
- For docx documents, list bullets cannot be extracted with `python-docx`, so for now a lot of irrelevant information will be shown until a custom docx parser is built.

  Example of good parsing along with Rating at the bottom.



```
(mypython) D:\Projects\Untitled Folder>python main.py "D:\Projects\Untitled Folder\Anil Gujiri.docx" --profile "Data Science"
Loading nlp tools...
Loading pdf parser...
Decoded the Doc
Skills Fetched
Final Resume
----------
Name: Anil Gujiri
Email: gujirianil18@email.com
Number: 8286865402
City/Country: Mumbai

Work Experience:
3 years 0 months
 - National Stock Exchange, Mumbai Associate Systems Analyst (2018 2021)
 - Supervised and led team members for RISK Management and operational processes.
 - Heavily involved in day-to-day SDLC process, developing ML solutions and maintenance.
 - Successfully deployed 3 end to end Machine Learning Models on production within a Span of 2 years.

Education:
6 years 0 months
 - Bachelors in Electronics and Telecommunication Engineering (2014 - 2017)
 - Don Bosco Institute of Technology, Mumbai
 - Diploma in Industrial Electronics (2011 2014)
 - VPMs Polytechnic, Thane
 - I hereby declare that the above-mentioned information is correct to the best of my knowledge and I bear the responsibility for the correctness of the mentioned parti
culars.

Skills:
Machine learning, Scripting, Java, Automation, Testing, Javascript, Tensorflow, Redhat, Finance, Excel, Algorithms, Process, Sql, Ubuntu, Technical skills, Keras, Workf
lows, Spark, Linux, Analysis, Python, Github, Programming, Training, Sdlc, Windows, Risk management, C, Electronics, Agile, Technical, System, Operating systems, Hive,
Shell, Oracle, Xgboost, Aws, Hadoop, Engineering
----------
Rating: [4]
```

Code: infoExtractor.py
Command: python main.py “PATH_TO_RESUME” –profile “Data Science”

Further Developments

1. Different model and skills set for each profile. For “Data Science” there should be a different model trained on particular skill set only.
2. Build a customised Named Entity Recognition model (NER) on large dataset containing many different CVs well-labelled for it to generalize well.
3. Give more weightage to particular keywords and then do the training. Like a Data Science should be proficient in Machine learning but not so in Big data. So for Data Scientist both ML and Big data should be present in the skills set for training but more weightage should be given to ML

Conclusion

In conclusion, for a general resume rater, such an approach could work well as a first-pass test. However, for a more robust rater that can achieve high true positives (ratings similar to what HR would give or translates to actually hiring good fits), a more supervised approach should be taken as discussed in *further developments*

For information extraction, a Named Entity Recognition approach would work much better than a rule-base approach. However, the only challenge is knowing what labels to label and labelling the labels correctly, as well as getting a dataset suitable for the company. A big company with many employees would benefit very well with such an approach. However, for smaller companies, rule-based approach still works best given its high recall (ensuring relevant information is captured).