How I used NLP (Spacy) to screen Data Science Resume

Position your Data Science Resume better through NLP

Venkat Raman · Follow
Published in TDS Archive · 6 min read · Jan 14, 2019

Image Source: pixabay

Resume building is very tricky. A candidate has many dilemmas,

> · *whether to state a project at length or just mention the bare minimum*
>
> · *whether to mention many skills or just mention his/her core competency skill*
>
> · *whether to mention many programming languages or just cite a few*
>
> · *whether to restrict the resume to 2 pages or 1 page*

These dilemmas are equally hard for Data Scientists looking for a change or even for aspiring Data Scientist.

Now before you wonder where this article is heading, let me give you the reason of writing this article.

The Context

A friend of mine has his own Data Science consultancy. He recently bagged a good project which required him to hire 2 Data Scientist. He had put a job posting on LinkedIn and to his surprise he received close to 200 resumes. When I met him in person he remarked, "*if only there was a way to select the best resumes out of this lot in a faster manner than manually going through all resume one by one*".

I had been working on few NLP projects both as part of work and as a hobby for past 2 years. I decided to take a shot at my friend's problem. I told my friend perhaps we can solve this problem or at least bring down the time of manual scanning through some NLP techniques.

The Exact Requirement

My friend wanted a person with Deep Learning as his/her core competency along with other Machine Learning algorithms know how. The other candidate was required to have more of Big Data or Data Engineering skill set like experience in Scala, AWS, Dockers, Kubernetes etc.

The Approach

Once I understood what my friend was ideally looking for in a candidate, I devised an approach on how to solve this. Here is what I had listed down as an approach

> *· Have a dictionary or table which has all the various skill sets categorized i.e. if there are words like keras, tensorflow, CNN, RNN then put them under one column titled 'Deep Learning'.*
>
> *· Have an NLP algorithm that parses the whole resume and basically search for the words mentioned in the dictionary or table*
>
> *· The next step is to count the occurrence of the words under various category i.e. something like below for every candidate*

Candidate X

Deep Learning	NLP	Big Data	Machine Learning	Data Engineering
6	0	1	6	0

The above candidate would be a good match for the 'Deep Learning Data Scientist' that my friend is looking for.

> *· Represent the above information in a visual way so that it becomes easy for us to choose the candidate*

The Research

Now that I had finalized on what my approach was going to be, the next big hurdle was how to accomplish what I had just stated.

NLP Part — Spacy

I was on a look out for a library that kind of does 'phrase/word matching'. My search requirement was satisfied by Spacy. Spacy has a feature called 'Phrase Matcher'. You can read more about it here.

Reading the Resume

There are many off the shelf packages which help in reading the resume. Luckily all the resume that my friend had got was of the PDF format. So, I decided to explore PDF packages like PDFminer or PyPDF2. I chose PyPDF2.

Language: Python

Data Visualization: Matplotlib

The Code and Explanations

The complete Code

Here is a Gist link to the full code.

Now that we have the whole code, I would like emphasize on two things

The Keywords csv

The keywords csv is referred in the code line 44 as 'template_new.csv'

You can replace it with a DB of your choice (and make required changes in the code) but for simplicity I chose the good ol excel sheet (csv).

The words under each category can be bespoke, here is the list of words that I used to do the phrase matching against the resumes.

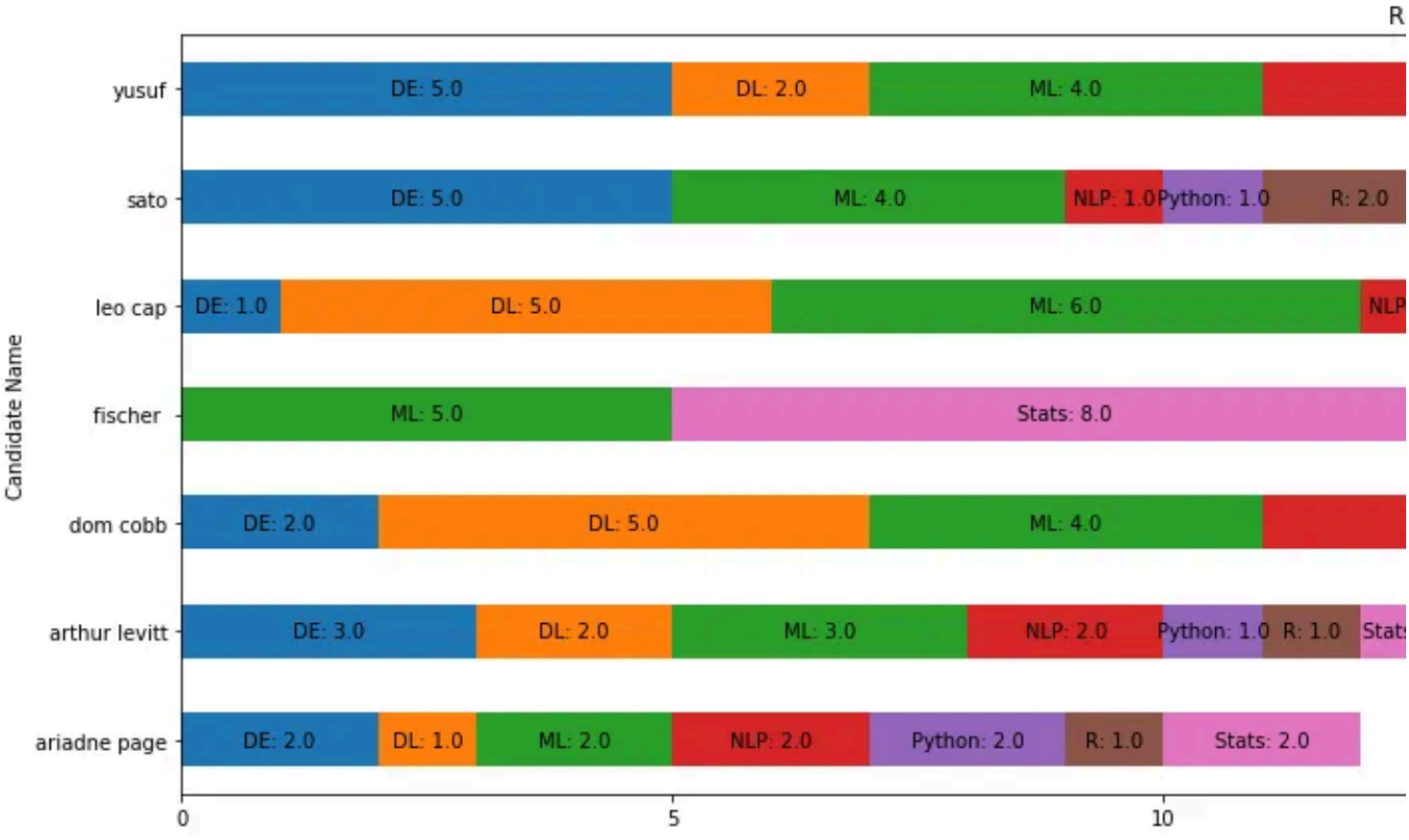
Statistics	Machine Learning	Deep Learning	R Language	Python Language	NLP	Data Engineering
statistical models	linear regression	neural network	r	python	nlp	aws
statistical modeling	logistic regression	keras	ggplot	flask	natural language processing	ec2
probability	K means	theano	shiny	django	topic modeling	amazon redshift
normal distribution	random forest	face detection	cran	pandas	lda	s3
poisson distribution	xgboost	neural networks	dplyr	numpy	named entity recognition	docker
survival models	svm	convolutional neural network (cnn)	tidyr	scikitlearn	pos tagging	kubernetes
hypothesis testing	naive bayes	recurrent neural network(RNN)	lubridate	sklearn	word2vec	scala
bayesian inference	pca	object detection	knitr	matplotlib	word embedding	teradata
factor analysis	decision trees	yolo		scipy	lsi	google big query
forecasting	svd	gpu		bokeh	spacy	aws lambda
markov chain	ensemble models	cuda		statsmodel	gensim	aws emr
monte carlo	boltzman machine	tensorflow			nltk	hive
		lstm			nmf	hadoop
		gan			doc2vec	sql
		opencv			cbow	
					bag of words	
					skip gram	
					bert	
					sentiment analysis	
					chat bot	

The Candidate — Keywords table

In line 114 of the code, the execution of the line produces a csv file, this csv file shows the candidates' keyword category counts (the real names of the candidates have been masked) Here is how it looks.

	Keyword Category						
Candidate Names	DE	DL	ML	NLP	Python	R	Stats
ariadne page	2	1	2	2	2	1	2
arthur levitt	3	2	3	2	1	1	1
dom cobb	2	5	4	10	5	1	0
fischer	0	0	5	0	0	0	8
leo cap	1	5	6	1	2	1	0
sato	5	0	4	1	1	2	2
yusuf	5	2	4	5	1	2	2

This may not be intuitive hence I have resorted to the data visualization through matplotlib as depicted below



Here DE stands for Data

From the chart it looks like Dom Cobb and Fischer appear more like specialists while others seem like generalists !!

Was the whole exercise beneficial?

My friend was really surprised with results achieved and it saved him a lot of time. Not to mention he did shortlist around 15 resumes out of nearly 200 resumes just by running the code.

Here is how the whole exercise was useful

Automatic reading of resume

Instead of manually opening each and every resume, the code automatically opens the resumes and parses the content. If this were to be done manually it would take a lot of time.

Phrase matching and categorization

If we were to manually read all the resume, it would be very difficult to say whether a person has expertise in Machine learning or Data engineering because we are not keeping a count of the phrases while reading. The code on

the other hand just hunts for the keywords ,keeps a tab on the occurrence and the categorizes them.

The Data Visualization

The Data Visualization is a very important aspect here. It speeds up the decision making process in the following ways

> *We get to know which candidate has more keywords under a particular category, there by letting us infer that he/she might have extensive experience in that category or he/she could be a generalist.*
>
> *We can do a relative comparison of candidates with respect to each other, there by helping us filter out the candidates that don't meet our requirement.*

How you can use the code

Data Scientist seeking job change / Aspiring Data Scientist:

Chances are that many companies are already using codes like above to do initial screening of candidates. It is hence advisable to tailor make your resume for the particular job requirement with the required keywords.

A typical Data Scientist has two options either position himself/herself as a generalist or come across as an expert in one area say 'NLP'. Based on the job requirement, a Data Scientist can run this code against his/her resume and get to know which keywords are appearing more and whether he/she looks like a 'Generalist' or 'Expert'. Based on the output you can further tweak your resume to position yourself accordingly.