[PRD] Resume parsing and ranking Workflow

Summary: The aim is to create a resume ranking workflow which can rank a corpus of resumes based on a job description

Created: Aug 2, 2021

The Problem Requirements Document (PRD) begins with a brief introduction that explains the goal of the PRD. While the introduction section is read first, it should be authored last. Writing the overview last allows the author to summarize the final content of the PRD, rather than write an introduction without knowing the end result.

Background

The challenge faced by most recruiters is to get the most relevant resumes pertaining to a job application. A solution which can rank resumes based on various attributes will help them screen resumes faster and also get back to the right applicants in time to start the first conversation.

Problem

The real challenge for this problem can be divided into the following major parts

- Extracting information from resumes and job description of various formats
- Classifying the resumes into proper groups based on their content. A resume can be tagged to multiple classes
- Standardization of information across the resumes so that ranking can be done on a common baseline
- Finding the most relevant resumes which can be a function of multiple factors. Eg: content relevance of the resume, Recruiter and Resume's organisations, recency of skills etc.
- Have a large search space across the resume corpus so that search localization doesn't happen. A large set of search problems has a long unexplored tail. Exploitation vs Exploration tradeoff

Personas

- **Affected Persona 1** recruiters : For being efficient in getting the right resume
- **Affected Persona 2** hiring managers : For being efficient in writing better JDs
- **Affected Persona 3** applicants: Not get stuck in the unexplored long tail

Requirements and Phases

	Requirements
Phase 1: Build the V0.0. The first baseline model of ranking	Must be able to accept resumes and job descriptions of some major formats in the database
	Extract information like Name,Email,City,Mobile,Education, Experience, Skills
	A ranking framework to rank the resumes
Phase 2: Build V1.0. The improved algorithm with standardized data	Must able to accept resumes across multiple formats
	Extract cleaner information and standardization of information in terms of education and experience. Eg: MBA~Masters of Business Administration, Software Engineer~Software developer, may 2014<2020 *(*time standardization)*
	Classification of resumes by tags eg: Software, PM, Analyst etc and also rank them by seniority
	Ranking of skills by recency and mapping them to JD
	Build a weighted Ranking score with more attention to certain sections than others. Can be parameterized
Phase 3: Build V2.0. The ranking algorithm dependent on recruiter behavior and organization along with resumes	Rank skills in resumes by their uniqueness and recency. Identify standardized soft skills
	Build a global ranking of organizations and identify past resumes in the system and rank them by their journey
	Identify similar resumes which were earlier successful for similar roles
	Identify resumes in the long tail which are similar by randomizing
	Create the ranking algorithms which is a function of skills, seniority, uniqueness, recency and organization relevance

<Phase 1: Build the V0>

The aim is to build a baseline resume ranking model

Considerations:

Most resumes follow the standard structure of Name, Address and Contact Details followed by Experience and Education and any other skills listed at the end. Hence the approach aims to use this structural understanding to extract information

Approach:

1. Read resumes and job descriptions as input. Few input types are to be supported, PDF in this case
2. Extract email, mobile. As they follow a standard structure use regex to get them
3. For name as most names are two proper nouns aim to match a pattern on a SPACY convert text of two proper nouns and select the first such pattern as the probability of that being a name is highest

4. Similarly select the first location as the city or location for the candidate
5. Build a corpus related to educational institutions and then search them in the resume. The lines with the words are related to education
6. Similarly lines not related to them are job experience
7. Build a regex to get months and years from the text and use them as experience
8. Identify all noun chunks between education and any other section as skills
9. Use NLTK to obtain occupations in the resume
10. For the skills obtain embeddings using BERT and similarly for JD use document embeddings
11. Obtain cosine similarity between the resume skills and jd embeddings and rank them based on similarity score
12. Return all the information related to resumes, name city email mobile skills experience education etc along with the rank

<Phase 2: Build the V1>

The aim is to improve on the existing baseline in terms of information extraction standardization and algorithm

Considerations:

Information can be in different formats and there are sections of importance in both JD and resume

Approach:

1. For name extraction apart from a proper noun sequence obtain the first person tagged in the spacy text and select the one which matches both
2. Can train a new NER recognition model by annotating various resumes using BILOU encoding of the various names and organizations present in the document. We can user BILSTM CRF or even BERT based NER Models to retrain such documents
3. The next part is to extract education qualifications and their we can do a regex search of a education qualification constants in the document and check their proximity to the education institutions in the document and use them as qualification
4. Train a multi class document classification algorithm by annotation document and their skills obtained into the required classes for us.
5. In case annotation is not available we can do a topic modelling on the skills to obtain various topics which relates to skills corpus and accordingly tag the documents
6. Also for new skills we can keep a service where in all new skills are updated and mapped to various classes so that they can searched in the documents
7. The next part is standardization of the skills etc. For that we can classify the soft skills in clusters by using their embeddings and then check cosine similarity of new skills to the cluster centroids of various soft skills groups. Else we can keep a large corpus of important soft skills words
8. For tech skills we need to maintain a master data source of skills and that can be separate service which keeps on updating skills
9. Identify the years mentioned in the experience section to get the overall experience which can be a good pointer to seniority
10. Date standardization in terms of Month Year or Year to be added to understand the sequence of experience
11. Extract required vs desired skills from the jd and map their similarity to the resumes.
12. Identify skill recency and competence by the section they are mentioned and the number of times they are mentioned in the resume
13. Complete every document object with JD similarity, seniority, skill to recency and competency matrix etc
14. Find the resumes with the similar tag of the jd and then for the corpus of resume build the weighted ranking model based on the features above. Attention can be given to certain sections to obtain relative ranking of each resume to the JD and other resumes.

<Phase 3: Build the V2>

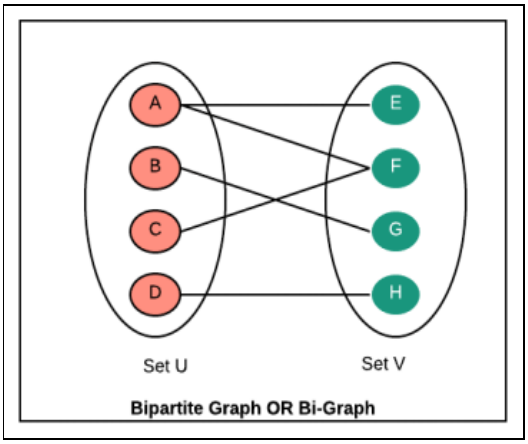
The aim is to improve on the existing baseline in algorithm

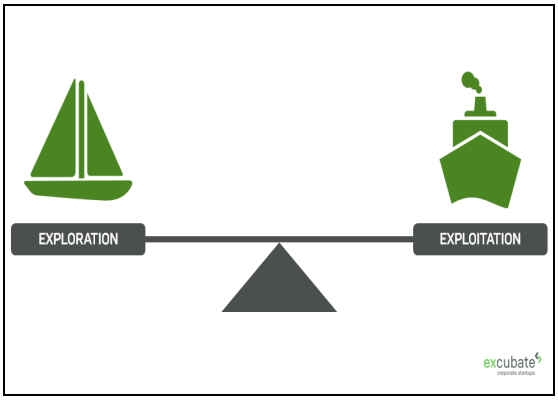
Considerations:

The major consideration is that the data extraction is perfected as well as standardized to a large extent

Approach:

1. Consider every resume and job description as part of a bipartite graph where every multiple resumes are part of a job
2. For a new job, resumes can be recommended by looking into the most relevant resumes from a similar job description.
3. All resumes can be ranked by the number of times they are referenced for similar job descriptions, which is an implementation of the page rank algorithm
4. Now when a new job description is added by the recruiter a random walk on the graph with certain steps occurs. The documents retrieved most number of times are the most relevant documents
5. There can be multiple ways to handle this random walk. User input is a collection of various parameters where the user can put importance to certain skills or organizations
6. Though earlier we classified resumes into groups of profession but all resumes belonging to a profession are not relevant to an organisation due to preference
7. We can build org clusters based on orgs of similar nature, importance and rank and then recommend resumes based on the sub section of the bipartite graph
8. This leads to another trade off issue in the recommendation system which is exploitation vs exploration. This problem can be solved by using Epsilon Greedy Strategy and then based on the user actions we can determine whether the exploration resume recommendations are working better or not. A user action can be selection of a resume part of the exploration group





Evaluation of Ranking System:

As the resume ranking is a kind of recommender system they are trickier to evaluate than - for example - a machine learning classifier. The reason is that the current recommender system influences the data that we have at hand (that we are using for training and testing). However we can use a combination and offline and online evaluation. *The Offline Evaluation* is used before deploying the model, to check its accuracy on the existing data. Usually, it's only after the model is deployed, and A/B tests that we can draw conclusions about the actual performance of our system. This second step is referred to as *Online Evaluation.* Another important thing about having a fixed offline evaluation framework, even though it's not accurate, is that we can use it as a basis to evaluate multiple recommender systems, or to fine-tune parameters related to our model, and observe how the results change (before pushing the changes to prod) .

Major Considerations for all ML Models:

1. All models should be trained on a stratified sample of the overall resume set
2. For any NER model and Resume classification models, classification metrics of F1_Score, Accuracy etc can be monitored to evaluate the model
3. For any recommendation system the last action can be predicted and prediction accuracy can be checked