

# Semantic Job Recommendation System

Project 11

## Team 58

Vishal Gupta  
Avabratha  
201403002  
UG3

Mahak Gambhir  
  
201303002  
UG4

Harshendra  
  
201505520  
PG1

**Mentored by: Soumyajit Ganguly**

# Introduction

**Information retrieval (IR)** is the activity of obtaining information resources relevant to an information need from a collection of information resources. **Information extraction (IE)**, on the other hand, is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents which often entails processing human language texts by means of natural language processing (NLP).

As can be perceived, these techniques have wide ranging applications, from analysing social media content and summarizing sports activities to other, more commercial applications like automation of customer service chats and search engines. More recently, a new need has emerged. Huge companies, with billions of dollars of turnovers, need a steady supply of new recruits who are competent enough to handle the tasks given to them efficiently and add to the worth of the company. For the company, this means rummaging through thousands of applications and selecting the best suited person for each of the positions available. Thus, a lot of resources are wasted in engaging more resources! This might not seem like a problem for small and medium size companies, but when the applications are in excess of thousands, and from people spread all over the world, it becomes important to streamline the process.

And it is here that we come in the picture! We use our knowledge of information retrieval and extraction to achieve the following:

- Extract important personal as well as professional details from the candidate resumes, presenting a more structured view of the provided information.
- We accept a 'job description', and then scan through the index of all the candidate resumes to dig out the 10 best candidates for the job.
- Besides, we also accept a resume, and present other resumes most similar to the given one. This helps if you know exactly what kind of person you need for the job.
- Moreover, we support 'field queries', whereby you may narrow down your search to incorporate specific details.

We believe that all these features will make the job of finding the right person for the right job considerably easy and fast. We also intend to add features like taking into account the specific details of the things mentioned, like the internship durations and qualities, academic growth, etc. We also intend to make a semi-structured '*summary*' for each person, highlighting the salient features of his/her resume.

## Related Work

Although we are not the first ones to tackle this problem, we find that there many, very diverse solutions that have been presented over time, all of which seem to work pretty well. This encouraged us to think freely and come up with some entirely unrelated approach, rather than trying to find bottlenecks in previous works. Nevertheless, undoubtedly there were some brilliant works on the subject which we came across and were inspired by. A recent paper, called “PROSPECT: a system for screening candidates for recruitment”[1], provided some key insights into how a resume should be handled and what part should be focused on. We was this paper which made us think that segregation of the resume into different sections may help a lot and present us with good results. Other than this, the “Skill Finder”[2] system details were very helpful, especially in terms of suggesting the use of NLP techniques like Named Entity Recognition (NER) and keyword identification. Other related works which we helped to some extend are mentioned in the references.

## Our Approach

Our first task was to read the resumes for processing. Since the de facto format for resumes is essentially the portable document format (PDF), we have considered our input to be a set of resumes in this format. The first step hence is to convert these into a format more amenable for further use. Apache PDFBox is an open source tool available, which converts pdf files into text files, which can easily be worked on in any environment. But we go one step further to suit our programming environment, which is python, and while doing so, also break down the text into semantic sections.

## Infobox and Jsons

For this, we had to understand the structure of a typical resume. It usually begins with the name of the person and some personal details, like contact, address and email. This is followed by education details, projects, skills, experience and often interests, research and courses/training. We segregate all these distinctly identified sections of the document and use it in two ways. The first is to create a json file corresponding to each document, with all the sections as different fields, allowing us an easy import in python in the form of a dictionary and hence enabling individual reference to each section. The second is to create an ‘infobox’ for that person. This is a short description of the person, containing the contact details as well as the salient features of his resume. This is supposed to be telling enough to allow one to have a fair estimate of the person’s level of competence, as well as provide information to contact him in case you feel the need.

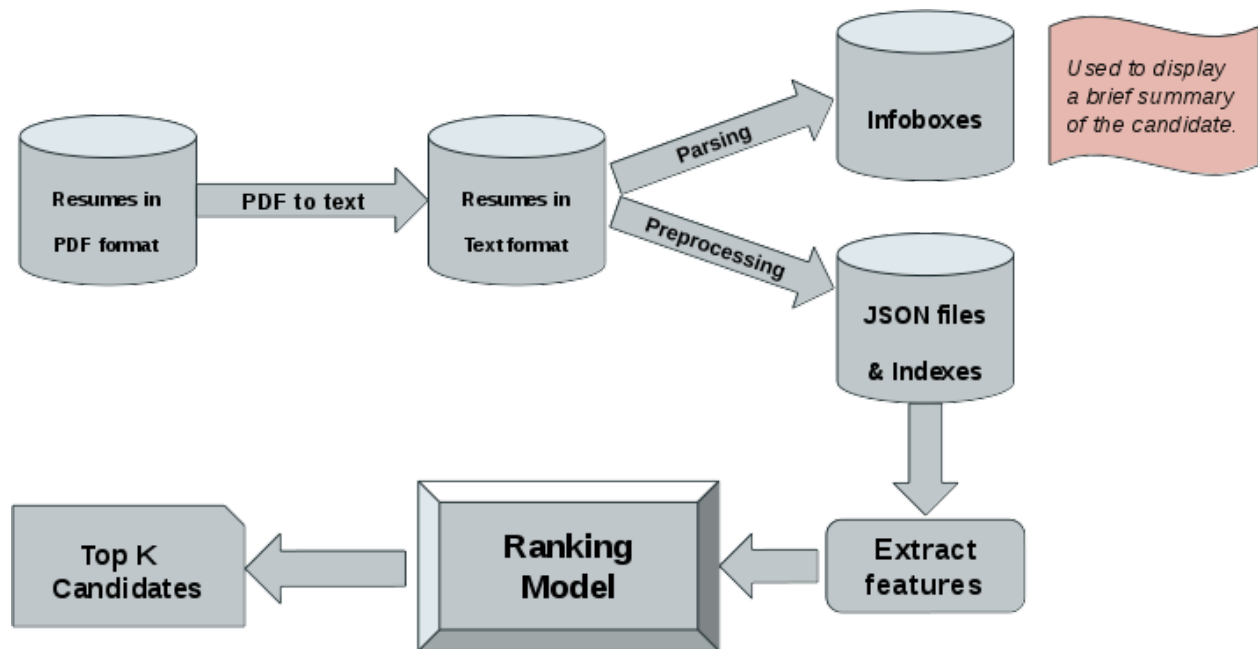
A slew of techniques have been employed to meet this end. Firstly, for the name, we had to use named entity recognition, because we could think of no other way we could think of to segregate the name of the person from a pretty cluttered text file. Though nltk also offers a NER Tagger, we opted for the Stanford NER Tagger, which gives much better accuracy, although at the cost of being a little slower. But since this indexing is required to be done only once, and the difference in time is not all that much, it seemed wiser to root for accuracy rather the speed. For identifying contact details, we used regular expressions, which was pretty effective give the necessarily structured nature of these pieces of information. Next, to identify the different sections of the resume, we use simple keywords as identifications for the beginning and end of different topics. This way, we exploit the semi-structured nature of the document.

## Document Matching and Ranking

Now that we have the data in a json file, we can move on to the retrieval of the most relevant resumes. Many papers we looked through were successful not because they were using some very complex technique, but simply because they were pruning the data carefully before applying a simple and basic ranking algorithm on it. And this is exactly what we did. Since we have separated the data into sections, we can now not only give different sections different weightages, but also give every query term the user enters a different relevance score with respect to each of these sections. For example, if I want to search for people having studied in Harvard, I would give the term more relevance for being in the 'education' section rather than in the 'experience' section which may only indicate an internship there.

We believe that this fine tuning, when coupled with usual ranking techniques, can give way better results than complex ranking models. Hence we decided to use 'tf-idf' ranking, along with cosine similarity as a start. Though the results here itself were reasonably good, we went further to distribute the score weightage over the cosine similarity, some specifics like internship durations and CGPA, as well as user defined relevance to particularly pertinent sections.

The following diagram illustrates and summarizes the entire process:



## Challenges:

Though there were a number of resources available which we referred, there were places we had to struggle.

The first challenge was to leverage the semi-structured nature of the resume to our advantage. Thus, we had in front of us the daunting task of going through several resumes manually and analysing the commonalities and thinking of ways to exploit them. It was doing this that we got the idea of segregating the resume into different sections. But this was easier said than done.

Before settling for the techniques that we actually used, we had to experiment a lot with various things and check manually for the accuracies. This was the most timing consuming and testing part of the project.

Other than that, there were some other small impediments and stumbling blocks, like experimenting with tools to deal with file formats, choosing between stanford and nlp taggers, the choice of languages and support for IR operations, etc.

# Results:

## User Interface:



Semantic Job Recommendation Engine

Enter Query

Results

## A sample search query along with the results:



Semantic Job Recommendation Engine

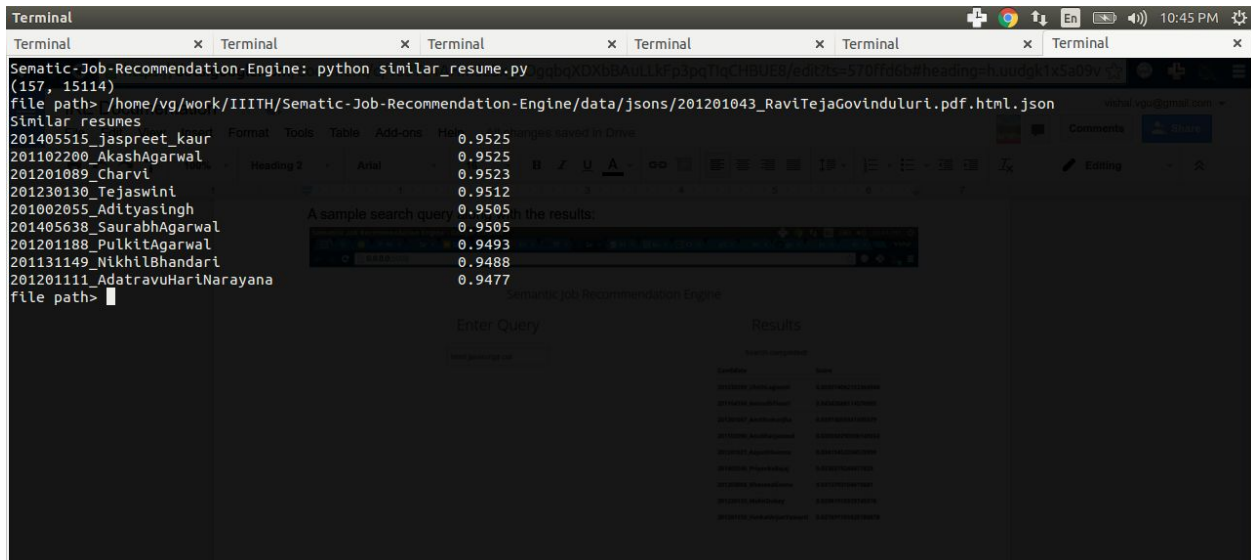
Enter Query

Results

Search completed!

Candidate	Score
201230200_UhithLagiseti	0.050074062153364546
201164104_AnirudhTiwari	0.04342660114576505
201201087_AmitKumarJha	0.03974059341495579
201102090_AnubhavJaiswal	0.035050290096145034
201201021_AayushSaxena	0.03415452338525999
201405540_PriyankaBajaj	0.0330375248477625
201203005_BhavanaGannu	0.0313703104615601
201230133_MohitDubey	0.02981915929745576
201201155_VenkatArjunYamarti	0.027691701625780878

## Similar Resume Search:



```
Sematic-Job-Recommendation-Engine: python similar_resume.py
(157, 15114)
file path> /home/vg/work/IIITH/Sematic-Job-Recommendation-Engine/data/jsons/201201043_RaviTejaGovinduluri.pdf.html.json
Similar resumes
201405515_jaspreet_kaur 0.9525
201102200_AkashAgarwal 0.9525
201201089_Charvi 0.9523
201230130_Tejaswini 0.9512
201002055_Adityasingh 0.9505
201405638_SaurabhAgarwal 0.9505
201201188_PulkitAgarwal 0.9493
201131149_NikhilBhandari 0.9488
201201111_AdatravuHartNarayana 0.9477
file path>
```

The background web application shows a search interface with the following data:

Enter Query	Results																				
201201043_RaviTejaGovinduluri.pdf.html.json	<table border="1"><thead><tr><th>Similar resumes</th><th>Score</th></tr></thead><tbody><tr><td>201405515_jaspreet_kaur</td><td>0.9525</td></tr><tr><td>201102200_AkashAgarwal</td><td>0.9525</td></tr><tr><td>201201089_Charvi</td><td>0.9523</td></tr><tr><td>201230130_Tejaswini</td><td>0.9512</td></tr><tr><td>201002055_Adityasingh</td><td>0.9505</td></tr><tr><td>201405638_SaurabhAgarwal</td><td>0.9505</td></tr><tr><td>201201188_PulkitAgarwal</td><td>0.9493</td></tr><tr><td>201131149_NikhilBhandari</td><td>0.9488</td></tr><tr><td>201201111_AdatravuHartNarayana</td><td>0.9477</td></tr></tbody></table>	Similar resumes	Score	201405515_jaspreet_kaur	0.9525	201102200_AkashAgarwal	0.9525	201201089_Charvi	0.9523	201230130_Tejaswini	0.9512	201002055_Adityasingh	0.9505	201405638_SaurabhAgarwal	0.9505	201201188_PulkitAgarwal	0.9493	201131149_NikhilBhandari	0.9488	201201111_AdatravuHartNarayana	0.9477
Similar resumes	Score																				
201405515_jaspreet_kaur	0.9525																				
201102200_AkashAgarwal	0.9525																				
201201089_Charvi	0.9523																				
201230130_Tejaswini	0.9512																				
201002055_Adityasingh	0.9505																				
201405638_SaurabhAgarwal	0.9505																				
201201188_PulkitAgarwal	0.9493																				
201131149_NikhilBhandari	0.9488																				
201201111_AdatravuHartNarayana	0.9477																				

## Links

- Code: <https://github.com/gvishal/Sematic-Job-Recommendation-Engine>
- Deliverables: [https://www.dropbox.com/home/IRE\\_Major\\_Project/deliverables](https://www.dropbox.com/home/IRE_Major_Project/deliverables)
- Presentation: <http://www.slideshare.net/vishalg2/semantic-job-recommendation-engine>
- Web Page: <http://mahakgambhir.github.io/Sematic-Job-Recommendation-Engine/>

## References

- [1] [Amit Singh , Catherine Rose , Karthik Visweswariah , Vijil Chenthamarakshan , Nandakishore Kambhatla, PROSPECT: a system for screening candidates for recruitment](#)
- [2] [Skill Finder: Automated Job-Resume Matching System, Thimma Reddy Kalva, Utah State University](#)
- [3] [V. Senthil Kumaran , A. Sankar, Towards an automated system for intelligent screening of candidates for recruitment using ontology mapping](#)
- [4] [Lin Chen , Richi Nayak, A Reciprocal Collaborative Method Using Relevance Feedback and Feature Importance](#)