Approach to Resume Ranking

Preliminaries

University Rankings

QS Rankings from http://www.topuniversities.com/university-rankings

Provides a list of top 800 universities in the world.

The site also provides a list of top 200 universities filtered subject wise.

Over 25 subjects covered across Natural Sciences, Engineering, Life Sciences, and Humanities etc.

There is also a ranking based on which universities have the best faculty. Top 400 lists for different areas like Engineering, Arts and Humanities, Natural Sciences etc.

Extract the ranking lists from topuniversities.com and create the following databases:

General list of top 800 universities in the world.

Subject specific ranking lists e.g. Math, Computer Science, Biology, Material Sciences etc.

Create lists for subject wise rankings based on faculty.

Position

Partition the set of academic ranks or position into the following equivalent classes:

1. Teaching Assistant (and its variations)
2. Lecturer (and its variations)
3. Associate Professor (and its variations)
4. Assistant Professor (and its variations)
5. Professor (and its variations)

6. Head of Department

7. Chancellor/Vice Chancellor

Higher the class index, higher the score received.

Degrees

Obtain set of degrees and abbreviations from www.abbreviations.com/acronyms/degrees/

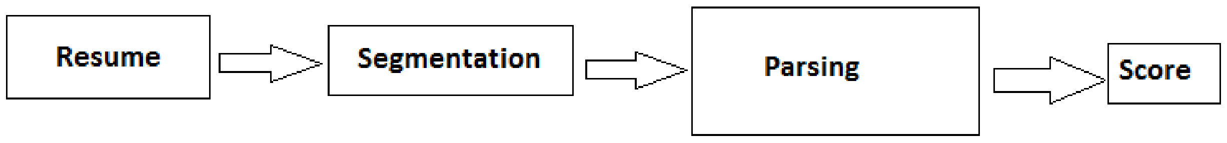
Create a DegreeList which is split into three categories:

1. Bachelor level

2. Masters Level

3. Doctoral Level

3 will receive the highest score, followed by 2 and then 1.



Segmentation

We have used **pdfminer** to convert a pdf to text. We then use **Beautiful Soup** to parse the xml structure of the resume.

The first step is to divide the resume into the following sections:

1. Education
2. Work Experience
3. Publications

In any resume we can assume that all headings are written in using the same format i.e. font size, typeface, bold/italics etc. This doesn't mean that the format will be same across resumes but within each resume the headings like "Education", "Publications" etc. will follow the same format. By using document parsers we can extract the format of each and every paragraph in the resume. We will then create a HashMap from this with the format as the key and a list of all paragraphs following that format as the value. So the headings list will look similar to [Objective, Education, Technical Skills, Work Experience, Projects]. Now we will also have a lot of other lists which correspond to non-heading content. We need to implement a learning algorithm to understand which list contains the headings. We can also take into account that the headings list will contain paragraphs not more than two or three words in length. But mainly, we need to consider the words within the lists for training our learning algorithm.
Once we understand which list contains the headings, we can assign each heading to our own predefined class. For example a heading like Work Experience will be mapped to the Experience label and the Qualifications heading will be mapped to the Education label. Once we know that a heading corresponds to a particular class then we get the text between that and the next heading as the text which belongs to that class. For example, we can say that the text between the Qualifications heading and the Technical Skills heading belongs to the Education class as Qualification has been mapped to the education class. Now we can parse it in a way particular to the education class.

Parsing

After segmentation we implement a parsing algorithm for each section to search for relevant information.

Since we are coding in python, we will make use of the python interface for the Stanford CoreNLP library. We're using this library to perform Named Entity Recognition (NER).

Education section 'E'

Here we assume that all Named Entities (NE) in this section will be names of universities

Parse through all words/phrases in E:

    If the word is in DegreeList:

```
        Add word to dl
    Else If the phrase is a NE:
        Add it to ul
    Else
        Add it to fl
```

dl[i] and ul[i] correspond to the degree and the university at which the candidate earned the degree.
Now parse through the words/phrases in 'fl' that lie in between max(Pos(dl[i-1]),Pos(ul[i-1])) and min(Pos(dl[i+1]),Pos(ul[i+1])). This will tell us the field of study wrt to the degree. If we find the string 'Biology' the degree was in the field of Biology.

[Pos(x) gives the position of string x.]

Work Experience section 'W'

We parse through W and make a list 'w' of all the NE. This would be a list of organizations the candidate has worked at.

We find out the duration of the job using regular expression matching.

```
Parse through all words/phrases in E:
    If the phrase matching RegEx is found:
        Add the number to d
    Else If the phrase is a NE:
        Add it to w
    Else
        Add it to 't'
```

This implies the candidate worked at organization w[i] for d[i] years

Now parse through the words/phrases in ‘t’ that lie in between max(Pos(d[i-1]),Pos(w[i-1])) and min(Pos(d[i+1]),Pos(w[i+1])). This will tell us the nature of the job.

[Pos(x) gives the position of string x.]

A statistical classifier will be used to ascertain whether w[i] is a university or a company.

If it’s a university we look for phrases like “Associate Professor”, “Research Assistant” etc.

If it’s a company we run a separate algorithm to find out the nature of the job eg. Systems Analyst, Head of R&D etc.

Publications

To find the number of times a research publication has been cited.

Suppose the research paper is titled T and authored by A and B (Candidate being A) and was published in journal J.

Use the **mechanize** library available for python to perform an advanced search on Google Scholar with the optimal combination of the parameters A, T, J, B.

From the html code of the results page extract the number of citations.

From the data provided by the Journal of Impact Factors (.csv file) search for J in the database and find its impact factor.

Use the number of citations and impact factor to give a score to the candidate.

Challenges

Our initial implementation will be as stated above. However there are problems. The toughest part of the project is efficient and accurate parsing ie the task of information extraction. There cannot be an exhaustive list of ‘Positions’ or ‘Nature of of Jobs’. The challenge is to create a system which is not solely dependent on phrase matching. This problem is AI complete and we continue to work towards finding a better solution.

Techniques we are considering:

Wrapper induction:- Wrapper induction is useful to extract information from structured dataset. We can learn wrapper for resume dataset with some labeled resume examples and use wrapper induction to extract information.

Tokenization:- Train statistical classifier to tokenize resume into different segments and sub-segments like person detail section, qualification section etc. Use machine learning algorithm like CRF++, HMM etc to tokenize resume into different section.