Project: CViA – Curriculum Vitae Analyzer

Background Information

A CV is a representation of a candidate's profile, skill sets and achievements. Matching CV across a job description to identify potential candidates to interview is a tedious job. In this project you would be solving this problem by designing and implementing ***resume parser and analyzer*** which will help recruiters to process resumes by extracting data in a meaningful way given a certain job description. The application is aimed for recruiters to efficiently manage electronic resume documents sent via the internet.

Information Categories for Parsing

In order to build a good parser you need to identify the information which might be relevant for recruiters to look for. It is not straight-forward to come up with an extensive list of fields in which a person's information can be said to be represented completely. Here, we just try to provide an initial list to get you started. You are encouraged to find other information as well.

First Name
Last Name
Headline/Objective
Location
Industry
    It might make sense to have a pre-defined list of industries the person can fall into. You could look at https://developer.linkedin.com/docs/reference/industry-codes as an example.
Summary
Positions Held/Work Experience
    The positions can each have the following fields -
        Title of the position
        Summary of the position
        Start Data
        End Data
        Company
            A company can again have the following fields -
                Name
                Type (Public or Private)
                Industry (See **5** above.)
                Ticker (Stock Market Ticker Symbol for public companies)
Profile Picture
Email-address
Associations
    List of various associations the candidate is a part of.
Interests
Publications
    Title
    Publisher Name
    List of other co-authors if any
    Publication Date
    Public URL for the publication
    Summary

Patents
- Title
- Summary
- Patent or application number
- Application or Granted
- Issuing Body
- List of Inventors
- Date of filing
- Data of patent grant
- URL to patent details

Languages
- Name
- Proficient Level
  - The list can include
    - elementary
    - limited-working
    - professional-working
    - full-professional
    - native-or-bilingual

Skills

Certifications
- Name
- Certification’s issuing body
- License number for the certification
- Start Date
- End Date

Educations
- School Name
- Field of Study
- Start Date
- End Date
- Degree Received (if any)
- Activities
- Other details/notes

Courses

Volunteer experience
- Role Performed as volunteer
- Organization
- Cause or reason for volunteering

Recommendations
- Summary
- Recommender

Date of birth

Honours and Awards

Phone Numbers
- Phone Type: e.g.., home, work and mobile.
- Phone number.

Main Address

Social Media Accounts

Keyword Matching and its Limitations

The simplest use of the information parsed from a CV is to match it against keywords that occur in job requirements. This works well, but only up to a limit. It works well for a list of skills like 'Java', 'Objective C', 'C#' because the occurrence of these keywords in a person's CV means that the person has experience using these programming languages (in this case).

But consider a skill like 'mobile development'. Can the occurrence of a keyword like 'mobile' guarantee that candidate is a mobile developer? Most CVs contain 'mobile' associated with a phone number as well. So, it is clear that simple keyword matching is not sufficient. Your application would clearly benefit from some context information. For example, you can search for a group of numbers in the vicinity of 'mobile' to see if it corresponds to a mobile number or something else.

Job Matching and Ranking

After you have parsed a resume and identified a good algorithm to extract the correct information from the parse, next job is to match a given resume and assign it some score. You are expected to come up with your own metrics to identify which CVs are the best match for the job. You application should be able to run on a batch of resumes for a given job description and should be able to assign a meaningful score which must be indicative of candidates aptness for that job in some manner.

To get a feel of the use case, put yourself into the shoes of a recruiter. You roll out a job description and you get 100 candidates. You do not have time to go through all 100 CVs. **From those 100 candidates that applied for that job, how would you use the information you parsed above to identify the top 3 that you need to interview?**
Example job descriptions - https://www.holmusk.com/careers

An Example Scenario

Imagine you are a recruiter at Holmusk and you have created an advertisement for an iOS developer position -

Responsibilities -
Work on, improve and develop our front-end client with some new features and an impressive user interface.
Work with the backend team to come up with APIs and sync data across platform.
Work across teams in a project based way and be involved in feature creations, product enhancements and experiments throughout the entire development lifecycle, from databases over backend and APIs to frontend and user experience
Be part of a strong team that passionately believes in what it does
Get exceptional career opportunities

Minimum Requirements -
Bachelor degree in computer science or a related field
At least 1.5 year of relevant work experience with iOS application development, published apps or significant demonstrable equivalent expertise.
Programming experience in Swift.
Ability to follow necessary software engineering practices - documentation, testing, etc.
Strong foundation in UIKit and CoreAnimation including UIKitDynamics and custom CATransitions.
Experience with multithreaded programming and concurrency in iOS
Strong foundation in Autolayout and Size Classes
Experience in architecting and optimizing persistent storage solutions in iOS

Preferred Qualifications -
Considerable programming experience in Swift and Objective-C/C++.
Deep technical knowledge of Cocoa touch ecosystem and iOS development paradigms such as MVC, VIPER etc. (http://www.objc.io/issue-13/viper.html).
Experience writing unit and integration tests with Xcode.
Knowledge of frameworks that can enhance user experience (such as Facebook's Pop (https://github.com/facebook/pop)) MVP application design and complex, reactive touch-based User Experience.
Good Contribution to the open source community (Cocoa Control contributions would be interesting)
Strong foundation in computer science, with competencies in data structures, algorithms and software design optimized for embedded systems.
Strong experience with designing and architecting client-server based apps on iOS.
Passion for healthcare and making the world a better place

Location -
Applicants should not be barred from working in Singapore

Keywords from the job description above would include but not be limited to
Skills: Swift, ObjectiveC, C++, MVP Application design, iOS, Data Structures, etc.
Education: Bachelor (Computer Science)
Location: Singapore
Work experience time: at least 1.5 years (to be parsed from their experience section of CV)

The above is not an exhaustive list of what information can be got from the JD, but just an example to provide you an example of one way of thinking. This can be done manually for every job (if you can't come up with an automatic solution) because you, the HR manager, would not mind doing this for every posting (but it does take away from your lunch hours, so this is not preferred). Automating CV evaluation and ranking is more necessary because at a popular company like yours, candidates can run into the hundreds.

Now, one very simple and trivial way to do this is to assign a certain weightage to each section - skills, education, certifications, work experience, language, etc.. A very simplistic example of use of this technique would be to just use a linear weighting scheme . Let us say our weighting formula is
Score = 0.25*skill_score + 0.25*education_score + 0.25*work_experience_score + 0.25*language_score
But then, now you need to define scores like skill_score, education_score etc.. One simple strategy to calculate skill_score would be to just use
skill_score = num_skills_in_cv_and_job/num_skills_required_by_job.
The above is just an intersection of skills possessed by candidate and skills required by job. We also assume every skill has an equal weightage here. But again, this won’t be true in a real scenario.

Once you have the scores for each candidate for a job, you can rank the candidates.

Advanced Features

You are welcome to add your own advanced features which in your opinion would increase the usability and/or efficiency of the application, for example few suggestions could be:

1. Performance goals in terms of speed and/or accuracy
2. WebUI for submitting the resumes.
3. Some sort of control over extent of matching for the recruiter.

Deadlines

DATE / TIME	Deliverables	Weightage
Fri, 18 Sep, 5pm	**SRS of project**	**5%**
Week 9 Tue, 13 Oct, Tutorial hrs	**work-in-progress demo**	**5%**
Tue, 10 Nov, 5pm **Demo in Tutorial hrs**	**Codebase and Report**	**20% basic features + 10% advanced/unique features**

Submission Instructions for deliverable - SRS of project:

You will write a Product Backlog(requirements) for the product you want to build. Follow the format of Sample Backlog in the IVLE-Files-Project folder.

As you *sprint* through the project, you could take high priority requirements from product backlog and make *sprin*t backlogs. You ***may*** include your assumptions, limitations, technology considerations, challenges, sprint backlogs, team strengths, members' roles, at the end of product log in your submission.

This is a team task. Each member is expected to contribute equally.

Submit your Product Backlog in the folder "P1" in IVLE workbin-Student Submissions.

Name the document: "<TeamNumber>_<ProductBacklog>_P1" - e.g. "Team02_ProductBacklog_P1".

There is no submission required for the work-in-progress deliverable. You will be demonstrating the work to your tutor in week 9 during tutorial hrs . "Work in progress demo" doesn't have strict requirements . Demonstration of your work which shows you are moving reasonably with regard to your SRS is expected.

Submission guidelines for the third and final deliverable shall be announced in week 11 of the semester.

Tips and Techniques

1. The first hurdle will be to parse the CVs from formats such as PDF to text. You can use open source packages like https://github.com/euske/pdfminer
2. When doing keyword matching, consider applying stemming and lemmatization (http://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html)
3. You might find regular expressions useful.

4. If you use a rule based system for parsing CVs, do make sure to run your rules by many CVs (at least those of your project team mates). That will make sure your rules don't just work for one particular type and format of CV.
5. When matching job descriptions to CVs, you might find it useful to parse the job descriptions into fields like skills, degree requirements etc. as well so that you can match the fields to information parsed from CVs.
6. You might want to assign a score to every CV with respect to a JD(Job Description) for the purpose of ranking.
7. To make this assignment easier, you can assume the CV you are parsing has some structure at least. Use that to complete all the requirements (information extraction into fields, matching with job description, ranking, etc.) before generalizing your solution to different formats of CVs.
8. You might find named entity recognition useful (https://en.wikipedia.org/wiki/Named-entity_recognition).

Note:

You are allowed to use any programming language and/or technology stack for implementation.

General discussions related to project are welcomed at IVLE. For communications with teaching team related to project use CS3219@googlegroups.com