iBeyond

Team Members:

Divya Karthikeyan

Bharathkumar Gunasekaran

Haroon Rasheed Paul Mohammed

Advisor:

Prof. Marti Hearst

Table of Contents

Acknowledgements

We would like to express our profound gratitude to Professor Marti Hearst for her exemplary guidance, mentoring and constant encouragement throughout the course of our project.

We would like to take this opportunity to express our heartfelt thankfulness to Jike Chong for all the valuable support and guidance and to Huangming Xie for his encouragement.

Our sincere thanks to our data source Simply Hired Inc.

We would like to also like to extend our gratitude to every one of our survey participants.

Lastly we are grateful to our families and friends for their everlasting support and encouragement

1. Executive Summary

1.1 The Problem

If finding the right candidate for the job is difficult for the recruiter sitting on top of a lot of potential candidate resumes, finding the right job among the myriad opportunities that exist out there is even more difficult for the job seeker. As job seekers ourselves, we often are at a crossroads and have the following questions in mind.

- Which jobs we are qualified for?
- What skills are the most important when I apply for a particular job?
- At what job do I have a better chance of succeeding at?

1.2 The Solution

iBeyond, is a tool that would help the job-seeker get answers to the above questions. The recruitment industry is awash with data. We intend to leverage the data in conjunction with Machine learning, Data mining and Natural Language Processing methodologies and techniques to tackle the "match-making" process: recommend jobs that the candidate might be interested in. Candidate matching systems today use methods that involve keyword scanning of the resume and comparing resume to the job description. The more modern systems also go the extent of generating a score for the candidates based on his or her credentials.

We have a very different approach. Unencumbered by just the job description, iBeyond leverages current/prior employee data to decide if a candidate would be a good fit for the job. This approach is both innovative and challenging.

2. Basic architecture

Content based recommender systems, like the one we intend to build, have three important steps: Data-Preprocessing, training the model and providing recommendations. Data-preprocessing is the step which involves transforming potentially unstructured data into structured format that can be used in training the model. Steps involved include typical information retrieval tasks: parsing unstructured data to extract features, feature selection, and feature extraction. The transformed data usually has a feature vector and a class label that act as the input to the supervised learning algorithm while training the model. When a new item for which the class label is to be predicted comes in, the model gives out a list of recommendations that the new user might be interested in. This is based on the same set of key features that are extracted from the new user's input. The basic architecture for iBeyond is as follows.

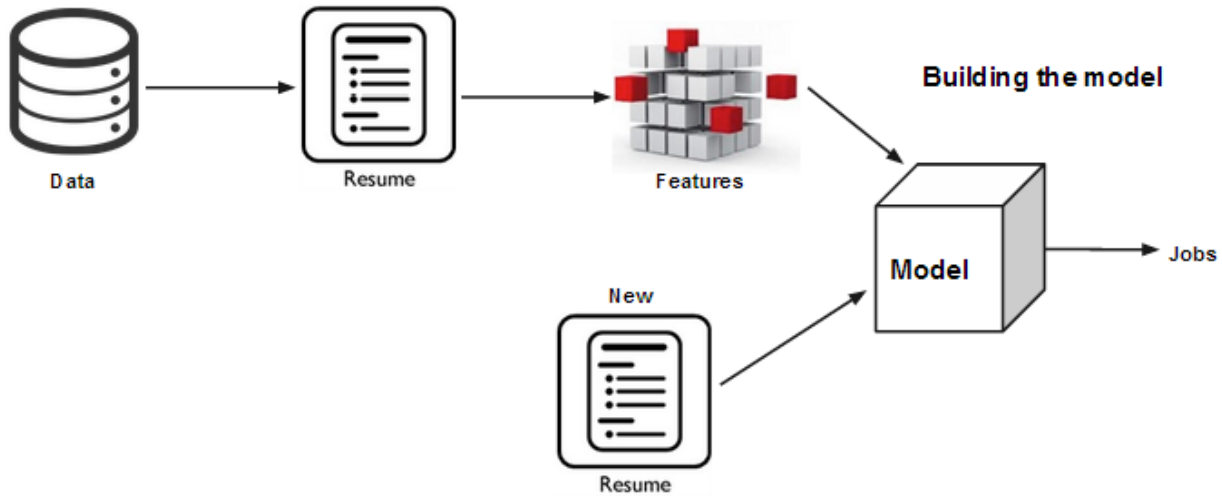


FIGURE 1 – BASIC ARCHITECTURE

3. Data

Source: Simply Hired Inc.

We received access to anonymized candidate resume data in XML format. The data encompassed resumes from diverse industries ranging from Nursing, Trucking to

Software. For the scope of this project, we had to limit to one specific industry. Hailing from the Information School at Berkeley with a background in Software Engineering, we felt most connected to the Software industry. We also had easier access to fellow students from the same industry to help validate our results. We extracted resume data specific to our field of interest using software related keywords.

4. Data pre-processing

The resume data at our disposal was in xml format. One of our resumes reconstructed in the format is as shown below.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<ResDoc>
    <resume canonversion='2' dateversion='2' present='734579' xml:space='preserve'>
        <contact>
            <name>
                <givenname>Haroon Rasheed</givenname>
                <surname>Paul Mohamed</surname>
            </name>
            <phone>(123) 456-7890</phone>
            <email>example@example.com</email>
        </contact>
        <education>Education
            <school id='7'>
                <institution>University of California</institution>
                <address>
                    <city>Berkeley</city>,
                    <state abbrev='CA'>CA</state>
                </address>
                <degree level='16'>Master of Information Management and Systems</degree>
            </school>
        </education>
        <experience end='present' start='729757'>Experience
            <job end='present'>2013 - Present
                <employer>CalCentral, University of California, Berkeley</employer>
                <title>Software Engineer</title>
                <description>Part of the development team for CalCentral, UC Berkeley's new online portal.</description>
            </job>
            <job>2013 - 2013
                <employer>EMC Corporation</employer>
                <title>Software Developer Intern</title>
                <description>Developed an application using HTML, JQuery, and Python to test the newly implemented xPlore REST services.</description>
            </job>
        </experience>
        <skills>Skills
            C++, Python, HTML, JavaScript, jQuery, CSS, PHP, SQL, XML, XQuery, R, Git
            Learning AngularJS
        </skills>
    </resume>
</ResDoc>
```

The tags are standardized and self-explanatory. As the first step, all personally identifiable information contained in the <contact> tag were removed from the resume data. The <experience> tag contains the employment history of the candidate. The most recent job has the attribute 'present' to indicate current employment. The <title> within that tag is used as the class label for that resume and all other details current employment details are removed from the resume. 80% of the available data was used for training and the remaining 20% for testing. From the training set, 80% is used to

train the model and the remaining 10% was used for validation during training. The data in XML format was converted to plain text documents.

4.1 Normalization

The most challenging problem with data pre-processing was normalizing our data.

- Class Labels – Job titles
    - Our initial analysis of the data revealed that the job titles had many variations. In a sample of about 1000 resumes, there were 800 unique job titles. For example the title Senior Software Engineer could be represented as Sr. Software Engineer, Snr Software Engineer, and Senior Software Eng. etc. Using the data in the current state, we would end up having multiple class labels for the same job title. Hence on analyzing the data and exploring the different manifestations of the titles we came up with a script that
        1) Mapped abbreviations such as asst, admin, sr, jr, tech to their full form.
        2) Removed unnecessary tabs, commas, special characters.
        3) Equated Director of Operations to Operations Director after removing the common morphological and inflectional endings using the porter stemmer.
- University/Institution names
    - University/Institution names in our data also had many variations which required cleaning/normalization. This was an essential step since we had planned on using the university names in two work flows.
        - Use University name as a feature
        - Leverage education information to generate professional network for the candidate.
    - Standard university names and their corresponding aliases were extracted from Freebase. Using this information along with n-gram analysis and basic text processing we created a lookup.

Normalization of this type required human intervention from time to time to perform quality control.

4.2 Feature Selection

Classifying each resume into one of the job titles was the goal. Using a supervised learning algorithm, our model had to be trained for the same. We stated some initial hypothesis that helped us choose our set of features initially:

- Resumes of managerial level candidates tend to contain more words and be longer in comparison to a programmer or software engineer.
- There are specific ngrams that are common to candidates in the same job level
- Candidates from specific specialization and universities tend to choose similar job roles.

The step prior to creating the learning algorithm was transforming the resumes to a data-structures that can be used as input. Dimensionality reduction using feature selection was used to eliminate less significant features from the dataset. After removing stop words and stemming words, the feature set that we created contained the top unigrams along with TF-IDF (Term Frequency – Inverse Document Frequency) value. Another feature set contained top bigrams, in which each pair of adjacent words was considered as a feature with the appropriately scaled TF-IDF value. We iterated through a number of different combinations of the top unigrams and top bigrams to check for improvement in the model accuracy. We also used a brute force approach to assess how our model performed using different combinations of the features from the resume such as the

1. Unigrams
    a. Presence of words
    b. TF IDF of words
2. Document length
3. Average word length
4. Bigrams

5. POS tags
6. Skills

5. Training the model

During the course of this project, we experimented with two multiclass classifiers:

1. Naïve Bayes
2. Support Vector Machines (SVM)

We started training the model using Naïve Bayes classifier. The following features proved not to be very good representation of the document: POS tags, Document length, Average word length, TFIDF of unigrams. Features that increased the accuracy, but by a small negligible margin: Bigrams. Hence we proceeded with only the frequency of unigrams in the documents and the resulting accuracy was close to 26% for 20 different classes.

Although this helped us determine the most informative features, an SVM classifier usually performs better for very high-dimensional data. For the same 20 class labels an SVM classifier gave us an accuracy close to 46%. Joachims [3] lists some of the reasons why SVM work best for text categorization, which are worth mentioning.

- SVM can handle a large number of features because they protect against overfitting, which does not depend on the number of features.
- A Naïve Bayes classifier necessitated a lot of feature selection. This was hindering the performance of the classifier when we knew that, even though there were a lot of features ranked low on frequency count, they were potentially informative. An SVM classifier obviated the need to eliminate a lot of such low ranked features.
- The vector space created by both CountVectorizer() and DictVectorizer() was sparse for the resume data. An SVM classifier is more suitable for such data.
- Other factors that make SVM a better classifier are robustness and automatic parameter setting obviating the need for parameter tuning [3].

So we decided to move ahead with the SVM classifier with more class labels. Our current classifier is trained for close to 100 labels. The approach that we implemented to optimize our classifier was by using scikit's GridSearchCV(). As the documentation specifies '*GridSearchCV implements a "fit" method and a "predict" method like any classifier except that the parameters of the classifier used to predict is optimized by cross-validation*'. The parameters to be tested passed to GridSearchCV() gave us the best results using the LinearSVC() classifier. The candidate may not benefit from just one prediction. So we included the top 5 results from the classifier in order to highlight the best possible career paths the candidate could pursue. These titles for each resume were obtained using probability estimates from SVM.

	Job title among top 5 predictions	**Predicting the exact job title**		
Feature Set used	**Accuracy**	**Accuracy**	**Recall**	**Precision**
Unigrams	0.69	0.45	0.45	0.47
Unigrams + Bigrams	0.72	0.46	0.46	0.5
Bigrams	0.74	0.47	0.47	0.54
Trigrams	0.68	0.43	0.43	0.52

Table 1 - SVM results

6. Validation

We tested the trained model with test data from the corpus of resumes using different features as listed below and obtained model accuracy as indicated below. The test data as already mentioned was the 20% of the resume corpus along with class labels. Here the class label refers to the current job title of the corresponding resume. The result of our model was the predicted job title for the resume. Our validation included comparing these two results.

We also set up a short survey and asked people what they envision their next job to be from a selected list of jobs that our model was trained on. They were asked to also upload their most recent resume. Below is a small selection of the results.

No.	Titles envisioned	Titles predicted
1	Business Consultant Consultant Data Analyst Data Scientist Developer Product Manager Project Manager Senior Business Analyst Senior Consultant Software Engineer Web Designer Web Developer	Senior Software Engineer Senior Consultant Software Engineer Developer Data Analyst
2	Designer Graphic Designer UI/UX Designer UX Researcher Web Designer	UI/UX Designer Designer Graphic Designer Marketing Assistant UI/UX Developer
3	Business Analyst Business Consultant Chief Executive Officer Data Analyst Data Scientist Information Technology Consultant	Data Analyst Senior Software Engineer Designer Business Consultant Software Engineer
4	Data Analyst Data Scientist Product Manager Software Engineer Systems Analyst	Software Engineer Senior Software Engineer Data Analyst Developer Management Consultant

Table 2 – Survey results

7. User Interface

Though the core of our product is our algorithm, it would best serve its purpose through an intuitive user interface. Our first iteration of the low-fi prototype is shown below. The user is given the option to upload his/her resume in text or pdf format as shown in Fig 1. Once they upload the resume and click “Begin Career Analysis”, the user is redirected to Fig 2 with three lists showing the Top matches for the candidate, top skills and top employers from the candidates’ professional network.

7.1 Iterative development process

All the backend code was written in python and scikit and NLTK packages. We decided to use Flask, which is a micro-framework for python and easy to setup and use.

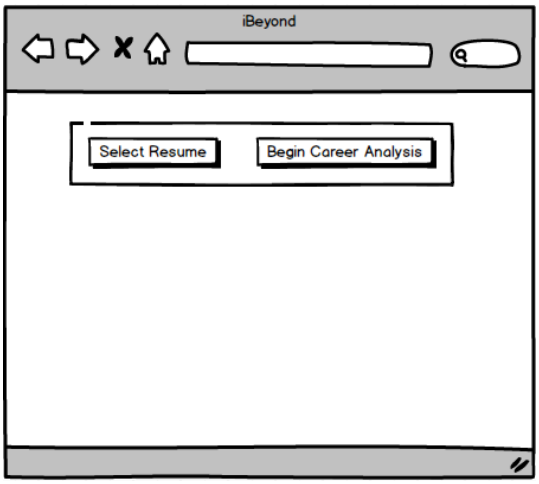


Fig 1

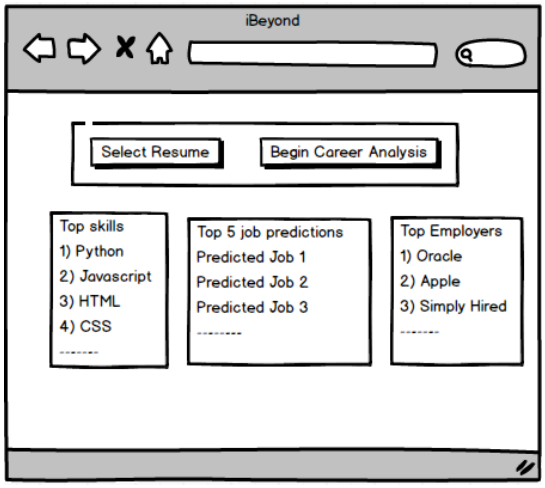


Fig 2

We used Twitter bootstrap to provide the basic scaffolding. jQuery provided us the simple interactions on the front-end. When a user uploads a Resume all the text from the resume is extracted to form an Ajax call to the Flask sever. The data is fed into the python script to create the feature vector that is the input to our classifier model. The output of the model is the top predictions of job titles. Along with the titles, the relevant data for rendering on our home page is returned as the output of the Ajax call. Having this template, we iterated through what information can be displayed on the

home page that would provide the best user experience. At a later stage we decided to add a Network analysis page, which would be a value add to our core functionality.

8. Final Product walkthrough

This is the landing page for our system. It gives the user the option to upload his/her resume. It also has links to the 'Network' analysis page and the 'About' page.

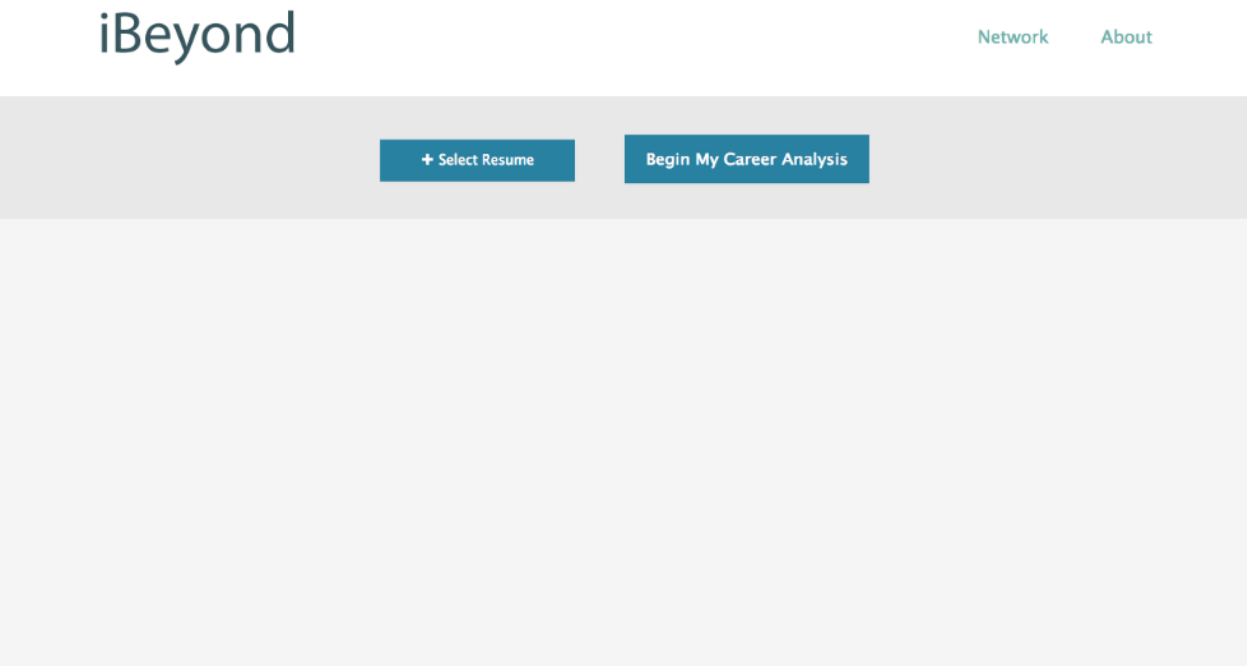


Fig 3 – Home Page

Once the user begins his/her career analysis, the below page is displayed. The center pane displays top matches for the candidate along with the match score. This score is computed based on various factors including your matching skills. The left pane helps the candidate determine the most important skills for any given title (currently restricted to about 100 titles). The Skill Search box on the right pane displays all the titles in our data for a specified skill. Candidate with niche skills can search through to explore the various job roles that would require their niche skill.

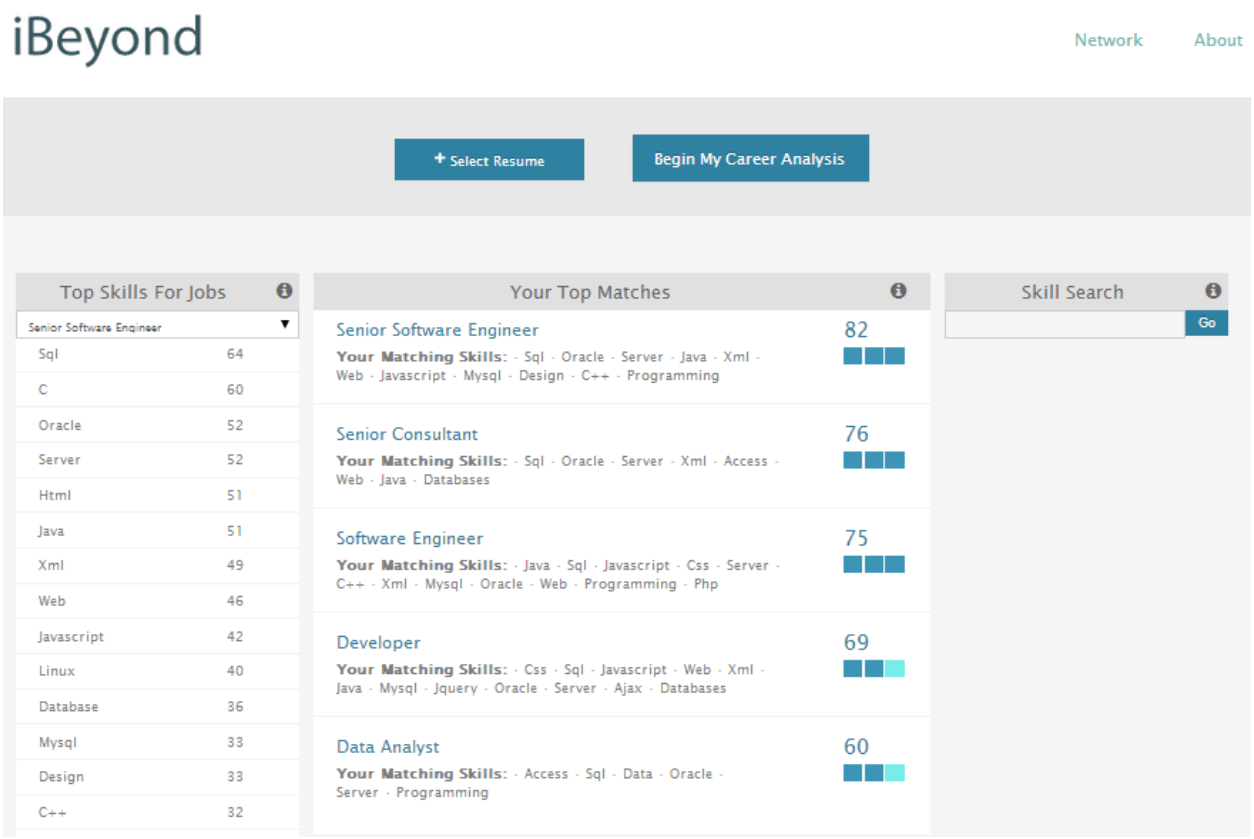


Fig 4 - Results

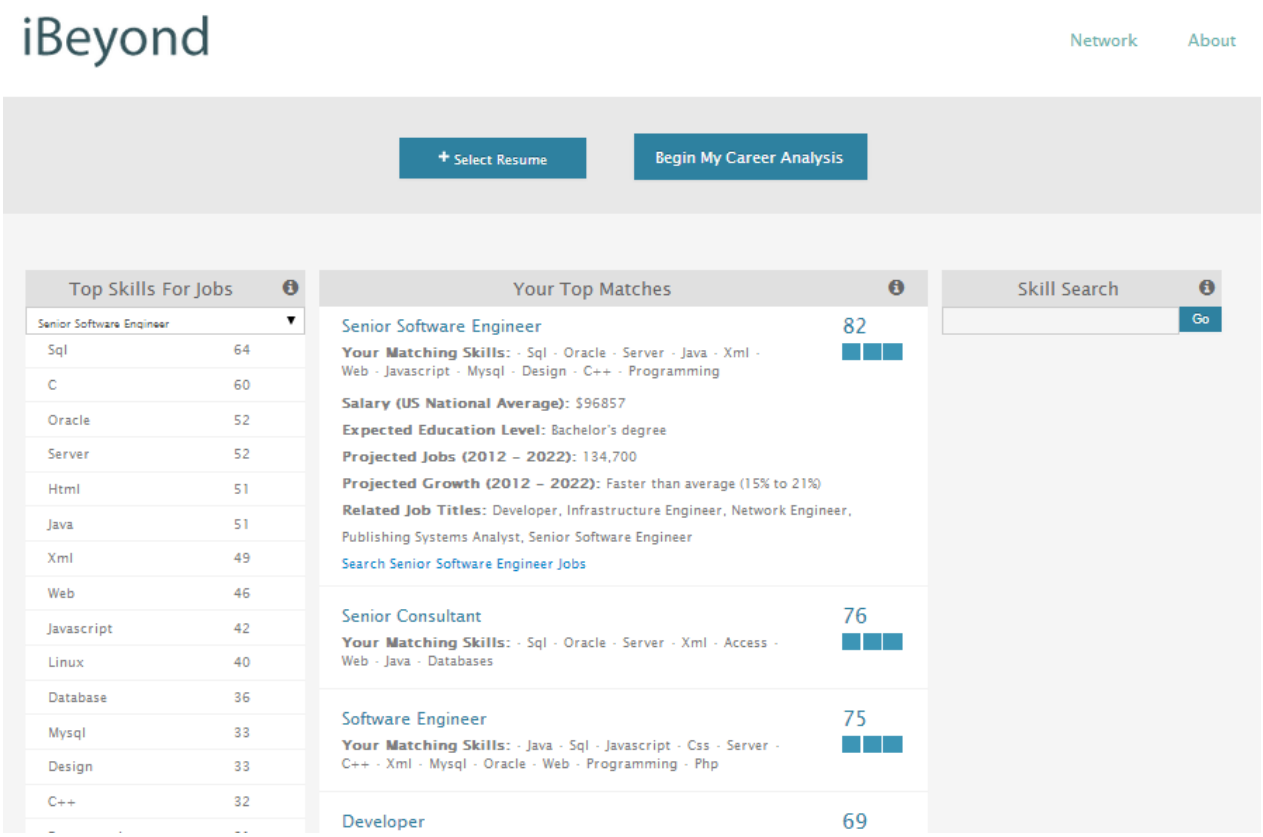


Fig 5 – Top skills expanded

On clicking any of the top matches, a card containing relevant information about the selected job title. This salary information was provided by SimplyHired and the other details come from O*NET (Occupational Information Network).

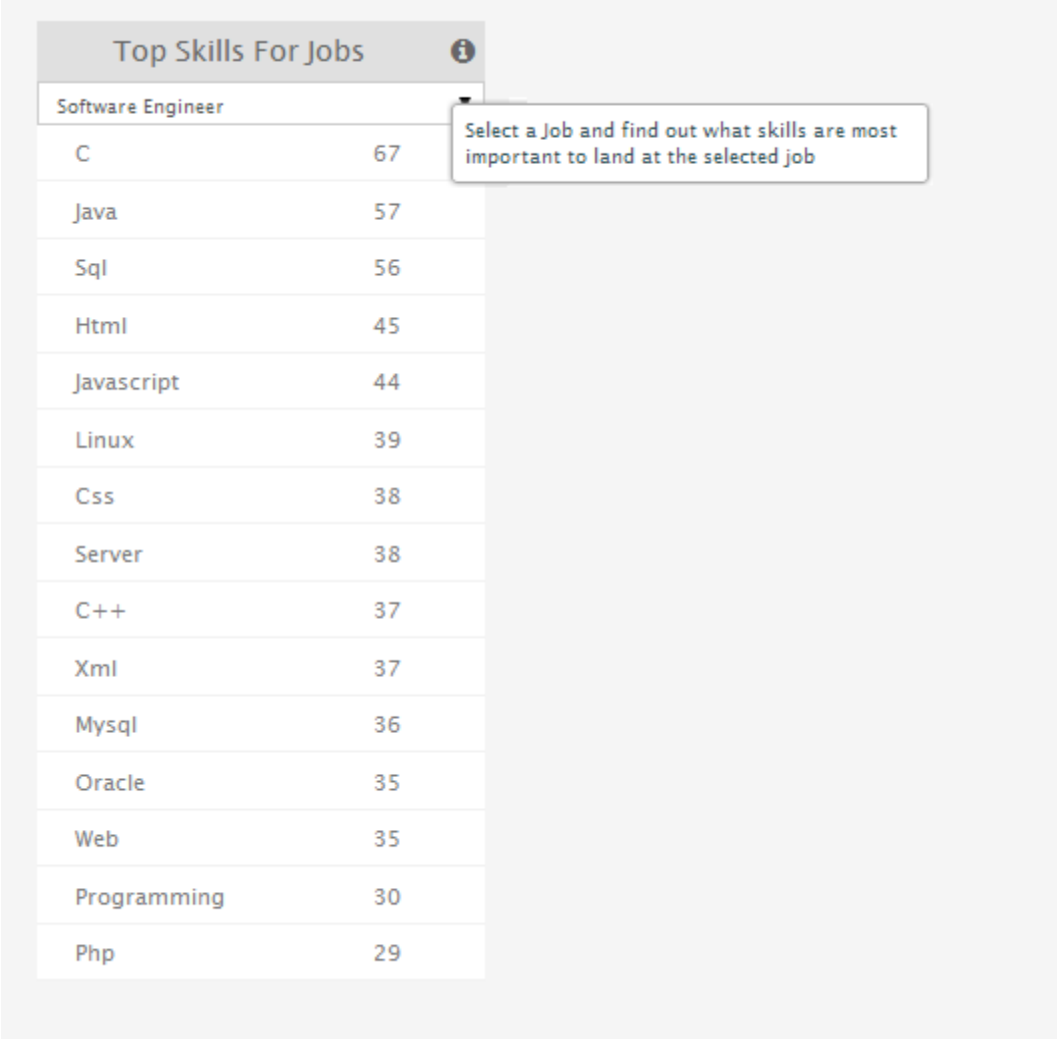


Figure 6 – Top skills for job

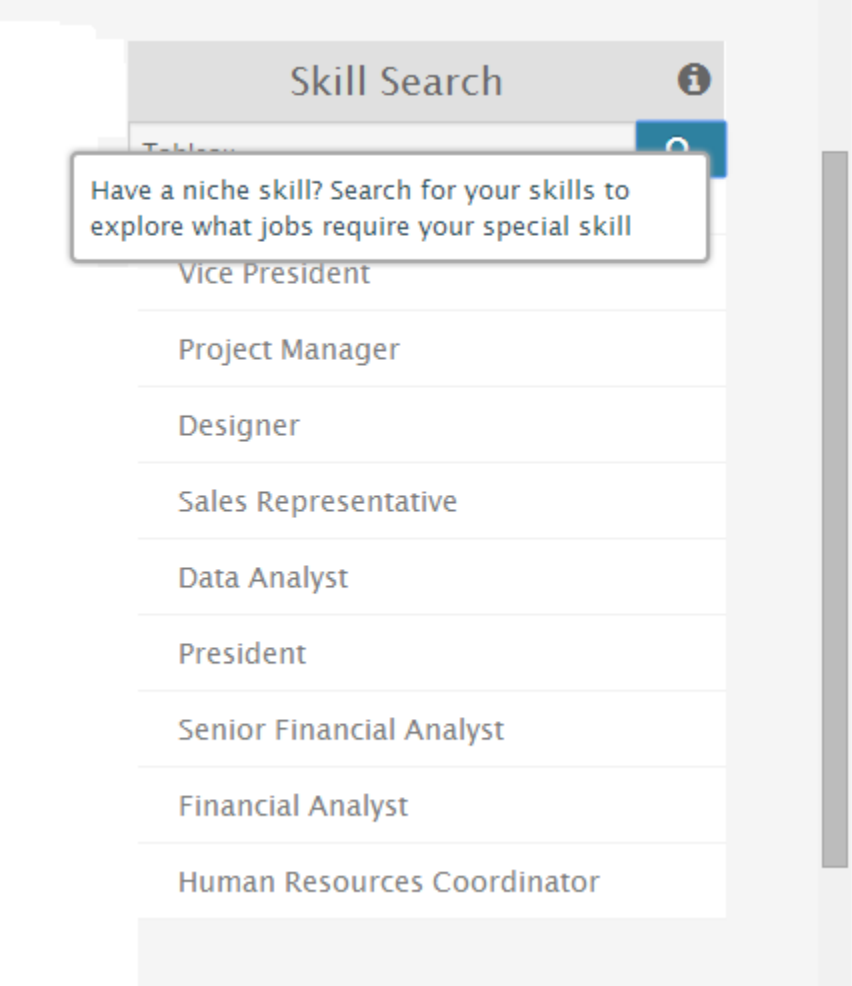


Figure 7 – Niche skill search

8.1 Candidate's Professional Network

This section provides n degree connections for the candidate. The candidate's institution is extracted from the uploaded text/pdf resume using n-gram analysis and look up against a standard institution list. Currently this is deployed as a proof of concept and hence displays only the employers of the connections. The first degree connections are computed from the candidate's educational institution / major. The second degree connections are linked via the first degree employers. A future

implementation of the professional network will contain connection's contact information along with their employment details.

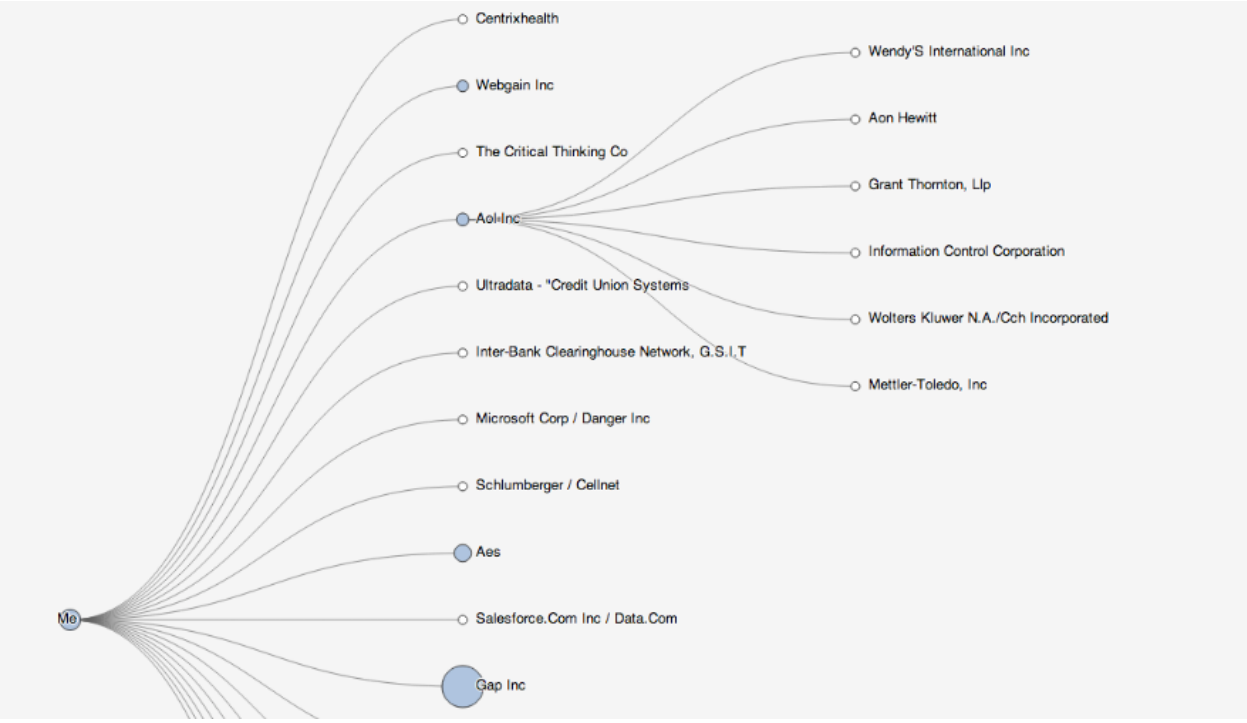


Fig 6 – A Snapshot of the professional network

9. Future work

- Compare results with traditional matching techniques.
- Explore the combination of our model with already existing models.
- Extend our model for other industries.

10. References

[1] Lops, Pasquale, Marco de Gemmis, and Giovanni Semeraro. "Content-based recommender systems: State of the art and trends." Recommender Systems Handbook. Springer US, 2011. 73-105.

[2] Underhill, David G., et al. "Enhancing Text Analysis via Dimensionality Reduction." Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on. IEEE, 2007.

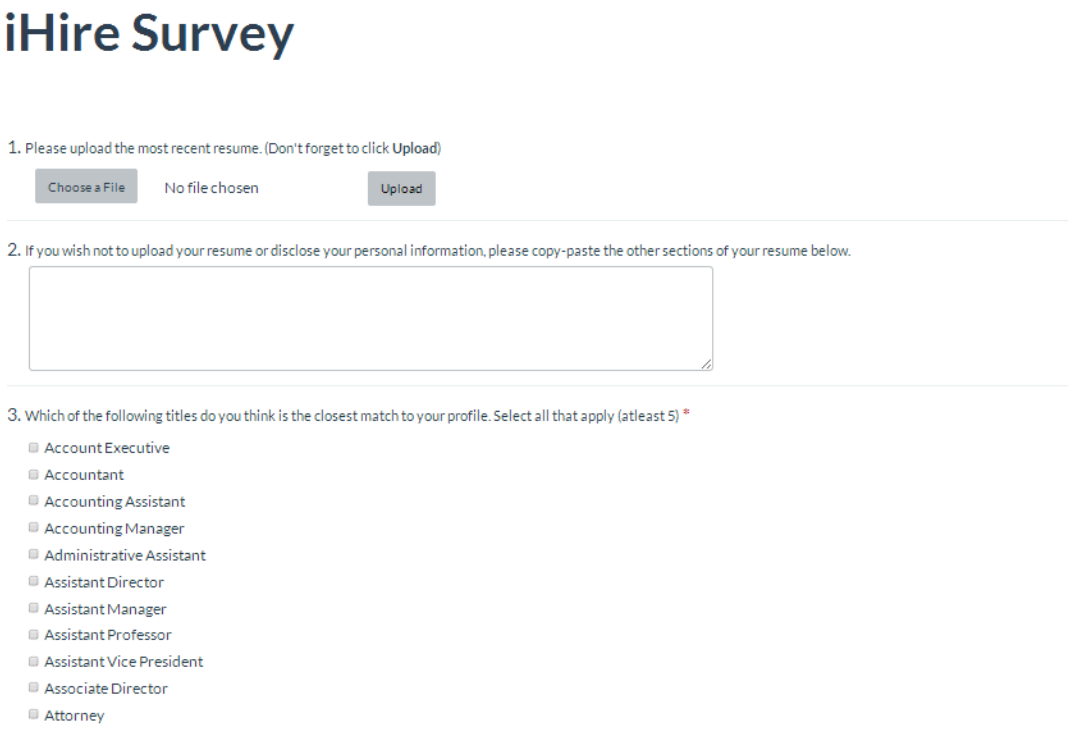
[3] Joachims, Thorsten. Text categorization with support vector machines: Learning with many relevant features. Springer Berlin Heidelberg, 1998.

11. Appendix

11.1 Source code

https://github.com/haroonrasheed333/NLPCareerTrajectory

11.2 Survey snapshot



iHire Survey

1. Please upload the most recent resume. (Don't forget to click **Upload**)

Choose a File | No file chosen | Upload

2. If you wish not to upload your resume or disclose your personal information, please copy-paste the other sections of your resume below.

3. Which of the following titles do you think is the closest match to your profile. Select all that apply (atleast 5) *

- Account Executive
- Accountant
- Accounting Assistant
- Accounting Manager
- Administrative Assistant
- Assistant Director
- Assistant Manager
- Assistant Professor
- Assistant Vice President
- Associate Director
- Attorney

- Senior Account Manager
- Senior Accountant
- Senior Associate
- Senior Business Analyst
- Senior Consultant
- Senior Director
- Senior Financial Analyst
- Senior Program Manager
- Senior Project Manager
- Senior Software Engineer
- Senior Vice President
- Software Engineer
- System Engineer
- Systems Administrator
- Systems Analyst
- Technical Writer
- UI/UX Designer
- UX Researcher
- Vice President
- Web Designer
- Web Developer

4. If you would like to know what our app predicted for you, please leave your email and we will get back to you.

Submit