AUTOMATED RESUME SCREENING SYSTEM

A

Project Report

Submitted

in fulfillment

for the award of the Degree of

Bachelor of Technology
In
Computer Science & Engineering

Supervisor
Prof. S.C. Jain

Submitted By
Jai Janyani (14/230)
Kartik Agarwal(14/231)
Abhishek Sharma(14/208)

Department of Computer Science & Engineering
Rajasthan Technical University, Kota
Session (2017-18)

CERTIFICATE

This is to certify that Jai Janyani, Kartik Agarwal, Abhishek Sharma of VIII Semester, B. Tech (Computer Science & Engineering) “2017-2018”, has completed a project titled “AUTOMATED RESUME SCREENING SYSTEM” in fulfillment for the award of the degree of Bachelor of Technology under Rajasthan Technical University.

Mr. Vikas Panthi

Assistant Professor

CANDIDATE’S DECLARATION

I hereby declare that the work, which is being presented in the Project, entitled “AUTOMATED RESUME SCREENING SYSTEM” in fulfillment for the award of Degree of “Bachelor of Technology” in Department of Computer Science & Engineering with Specialization in Computer Engineering, and submitted to the Department of Computer Science &Engineering, University College of Engineering,Rajasthan Technical University is a record of my own investigations carried under the Guidance of Mr. Vikas Panthi, Department of Computer Science & Engineering. I have not submitted the matter presented in this Report anywhere for the award of any other Degree.

Candidate Name	Roll No.	Signature of Candidate
Jai Janyani	14/230	
Kartik Agarwal	14/231	
Abhishek Sharma	14/208	

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and teachers. We would like to extend my sincere thanks to all of them.

We am highly indebted to Mr. Vikas Panthi for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our gratitude towards our parents & faculties of CS department for their kind co-operation and encouragement which help us in completion of this project.

Our thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

ABSTRACT

Designing an automated system to extract information from unstructured resumes and transform that information to structured format. And ranking those resumes based on the information extracted, according to the skill sets of the candidate and based on the job description of the company.

Today, online recruiting web sites such as Monster and Indeed.com have become one of the main channels for people to find jobs. These web platforms have provided their services for more than ten years, and have saved a lot of time and money for both job seekers and organizations who want to hire people. However, traditional information retrieval techniques may not be appropriate for users. The reason is because the number of results returned to a job seeker may be huge, so job seekers are required to spend a significant amount of time reading and reviewing their options.One popular approach to resolve this difficulty for users are recommender systems,which is a technology that has been studied for a long time.

In this thesis we have made an effort to propose a personalized job-resume matching system, which could help job seekers to find appropriate jobs more easily. We used a information extraction library to extract models from resumes and job descriptions. We devised a new statistical-based ontology similarity measure to compare the resume models and the job models. Since the most appropriate jobs will be returned first, the users of the system may get a better result than current job finding web sites.

Using NLP(Natural Language Processing) and ML(Machine Learning) to rank the resumes according to the given constraint, this intelligent system ranks the resume of any format according to the given constraints or the following requirement provided by the client company. We will basically take the bulk of input resume from the client company and that client company will also provide the job description and the constraints according to which the resume should be ranked by our system.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

Chapter 1 Introduction

1.1 Project Specification

Finding the right person for the right job has never been an easy feat for companies, whose value is very often to a large degree derived from their manpower. With the increased mobility of job seekers in recent years, more and more jobs are seeing rapidly growing pools of potential candidates, requiring respective recruiters to wade through hundreds if not thousands of CVs to find the perfect match.

When there is a large number of candidates, the automation can also include a more challenging task: scoring and ranking candidates' CVs according to their match to a job posting (represented concretely as a query). This task is usually implicitly performed by a search engine's retrieval model, which computes ranking scores based on the text overlap of the CVs with the query keywords. While text similarity might reflect good matches for some jobs, it is easy to find examples where this naive approach fails (e.g. a warehouse manager should not appear on top of a list for assistant warehouse manager). Improvement could be obtained through data-driven approaches, however, in the domain of recruitment it is very costly to compile and annotate enough data so that supervised learning would be possible

This thesis takes on a data-driven approach but avoids the hand labelling of training data by using a different kind of information: We use a dataset with recruiters' hiring decisions to learn a ranking model, which can improve on the initial search ranking. For this purpose we first introduce field relevance models for CVs, which are generally unsupervised models that can take advantage of implicit domain knowledge in a large collection of CVs. We demonstrate in an experiment that such a model has the potential to improve the recall of retrieval when applied in a simple query expansion set-up.

To improve the ranking of CVs we take on a feature-oriented, learning-to-rank approach (LTR), i.e. we propose a number of features for our problem domain, forming the basis for a machine learning algorithm whose resulting model is a ranking model. In particular, we also propose features which can be computed based on the field relevance models.

The main contributions of this thesis can be summarised as follows:

• We propose an unsupervised model that can take advantage of implicit knowledge to summarize every resume and extract information from it.

• We adopt a feature-oriented view of retrieval and describe a number of features that can be used in a learning-to-rank framework for the ranking of CVs.

• We used K Nearest Neighbours and Content based filtering algorithm for the ranking of CVs according to job description

1.2 Background & Motivation

Hiring the right person for the right job is a common challenge faced by all companies. Especially for positions with a large number of applicants the search for the right candidate(s) can feel like looking for a needle in the haystack. In these situations traditional methods of recruitment can be too expensive and time-consuming to be a viable option. Hence, not surprisingly, recruitment technology that can facilitate this process are in high demand. E.g. using a (searchable) database of candidates and a search engine a recruiter can preselect a small number of suitable candidates from a much larger pool so as to assess them in further recruitment procedure. It should be noted that the goal of such software is not to replace the “human” in human resources but to make the decision process smoother for the recruiter. For this purpose an increasing number of software packages provide means for executing recurring tasks automatically. Both CVs and job postings can be automatically parsed and relevant information is extracted and stored in databases. Easy-to-use interfaces are provided for maintaining the quality of extracted information (e.g. for manual corrections) and for keeping track of typical HR processes involving vacancies, candidates and interviews(so-called applicant tracking systems or ATS). With the growing importance of social media, more and more companies nowadays also offer “social recruiting” capabilities, which can tap into an even larger, more global pool of qualified candidates through social media platforms such as LinkedIn and Xing. Thus, in order to take full advantage of the bigger candidate pools, it is crucial to apply smart search and ranking strategies such that good candidates are indeed placed on top and do not disappear in the crowd.

The goal of this thesis is to extend the search and ranking component of an existing commercial recruitment software package. In particular, we target the ranking component of a CV search engine, which is responsible for scoring and ranking candidates’ CVs for queries (either issued by users or automatically translated from job postings). This existing software offers some basic functionalities, which form the foundation of many common recruitment processes:

• The automatic CV parsing extracts relevant information such as name, address, skills and previous work experience from original CVs (e.g. given as PDF or DOC documents) and transforms them into a searchable semi-structured format.

• The search engine indexes parsed CVs and enables searching with semi-structured queries as well as through search facets and tag clouds. CVs are assigned a relevance score w.r.t. the query by the search engine and are ranked accordingly.

• Automatic vacancy parsing extracts relevant information from vacancies such as the title of the advertised position, skill requirements and other job-opening-related keywords.

• The query generation component automatically generates semi structured search queries for finding matching candidates in the document collection.

The CV and vacancy parsing models are machine-learned models, which are trained to detect relevant phrases and sections in CVs and vacancies and can infer what kind of information the given phrase represents. Knowing the "meaning" of relevant parts of a CV allows more sophisticated search and filtering options, e.g. by searching only in the skills section or filtering candidates by their years of experience.

The workflow that we are mainly interested in involves the query generation component, which uses the information obtained from the vacancy parsing model and generates a query according to a predefined template. This kind of query is generally longer than user defined queries and contains a lot more information. An example query is given in Listing 1, which contains both terms that should match in specific fields and terms which can match anywhere in the CV (so-called fulltext terms).

Thus, these queries provide a good basis for finding candidates that match the original job posting well with very little human effort. Based on the search terms in the generated query, the search engine in our current system computes a score for each candidate's CV, according to which a final ranking is created. The focus of our work is to extend this current ranking system by learning a model that can re-rank an already ranked list of candidates according to some notion of suitability for a given query. Concretely, the initial ranking is performed by the search engine's TFIDF based retrieval model. Our "re-ranking" model should manipulate the ranking of this preselected list to ensure that the best candidates are placed on top. Furthermore, we want to gain some understanding of the aspects that play a role in the learning and the ranking procedure and how they relate to possible notions of suitability /relevance.

This task is challenging because we face a logical (i.e. given the already existing set-up) but highly competitive baseline provided by the search ranking. Previous user feedback suggests that the retrieval model used by the search engine already captures some notion of relevance that has a correspondence to the suitability of candidates. The approach that this work follows to tackle this challenge is one of machine learning, i.e. we want to learn a model from data without having to craft a ranking function or ranking rules explicitly. This approach requires a suitable dataset from which a ranking model can be learned. The details of the datasets that are used in our project are given in the next section.

The practical motivation for our learning approach is the availability of a relatively large dataset which contains real-world job ads, original CVs of the applicants for these jobs as well as information about which candidates were hired in the end of the recruitment process. We will refer to this dataset as the hiring decisions. Another, much smaller dataset contains human relevance judgements for a number of job ads and CVs of people who did not necessarily apply for the given job(in this thesis referred to as the relevance assessments).

1.3 Objective and scope

The major objective of our system is to take the current resume ranking system to other level and makes it more flexible for both the entity.

1) Candidates, who has been hired.

2) Client company, who is hiring the candidates.

Candidates, who has been hired : Candidates who are searching for jobs after been graduated. Out of those, major number of candidates are so much desperate that they are ready to work on any post irrelevant to their skill set and ability.

The main reason behind this unemployment is like a cancer to our society, if a guyis not got place after been passed out for 1yr, society include relatives starting blaming that guy. Inspite of this reason the candidate are ready to work in any condition, on any post. So they don't have to face those situation.

Where our system help such candidates to get hired by such a company or an organisation who really worth their ability and their skill sets. Where our algorithm will work in such a way that with the help of the previous result and previous ranking constraints, it will try to optimize the current result, which we called it Machine Learning.

This will make sure that the relevant candidate is been hired for that particular vacancy. You can say best possible candidate.

Client company, who is hiring the candidates : Like I am the owner of a particular organisation, obviously my aim would be to create such a team which is the best team in the world. It is like, if there is a vacancy of a java developer in my organisation. So, I won't prefer to hire a python developer and then make him learn Java. That will be pretty useless and time consuming for both that candidate and for the organisation too.

Where our system help the organisation to make out the best possible candidates list according to their given constraints and requirement for that particular vacancy.

This kind of approach, will help our hiring sector to improve like anything and make it more efficient as the relevant person is getting a relevant job. So there would be no regrets for both the entities, client company and that hired candidate. Hence satisfaction will be achieved.

As we know Indian I.T sector is second largest candidate recruiting sector of our country. It contribute about 7.5% to our Gross Domestic Product(G.D.P) Our Proposed system is initially concerned with the I.T sector of our country. It is mainly going to deal the Indian I.T industry but if you talk about the pro version of our system it can be extended to various other commercial sector where, intake and elimination are in bulk like for Govermental Jobs.

1.4 Recommender System

Job searching, which has been the focus of some commercial job finding web sites and research papers is not a new topic in information retrieval. Usually scholars called them Job Recommender Systems (JRS), because most of them used technologies from recommender systems. Wei et al classified Recommender Systems into four categories[48] : Collaborative Filtering, Content-based filtering, Knowledge based and Hybrid approaches. Some of these techniques had been applied into JRS; Zheng et al. [44] and AlOtaibi et al. [3] summarized the categories of existing online recruiting platforms and listed the advantages and disadvantages of technicalapproaches in different JRSs. The categories include:

1. **Content-based Recommendation (CBR)** -
   The principle of a content-based recommendation is to suggest items that have similar content information to the corresponding users, like Prospect [43].
2. **Collaborative Filtering Recommendation (CFR)** –
   Collaborative filtering recommendation finds similar users who have the same taste with the target user and recommends items based on what the similar users, like CASPER [35].
3. **Knowledge-based Recommendation (KBR) –**
   In the knowledge-based recommendation, rules and patterns obtained from the functional knowledge of how aspecific item meets the requirement of a particular user, are used for recommending items, like Proactive [24].
4. **Hybrid recommender systems** combine two or more recommendation techniques to gain better performance, and overcome the drawbacks of any individual one. Usually, collaborative filtering is combined with some other technique in an attempt to avoid the ramp-up problem.

Rafter et al. began to use Automated Collaborative Filtering (ACF) in their Job Recommender System, “CASPER” [35]. All these factors are viewed as measure of relevance among users. The system recommend jobs in two steps: First, the system finds a set of users related to the target user; second, the jobs that related users liked will be recommend to the target user. The system use cluster-based collaborative filtering strategy.

CASPER also allows users to search jobs by a query which is a combination of some fields: like location, salary, skill and so on. The system uses such query to find jobs, and the returned jobs are ranked with the collaborative filtering algorithm.In their paper, the authors do not give a detailed description on how to detect the related fields they need and how to the transfer semi-structured job description to the structured data.

The shortages of collaborative filtering: First, since the number of search results is huge, and the results are sorted randomly, the probability of two similar usersreviewing the same jobs is low, which causes the sparsity problem of collaborativefiltering. Second,because recommended jobs are from others users’ search results, since the quality of current searching result are low, the quality of recommendation cannot be high.

1.5 A Short Introduction to Learning to Rank

Learning to rank refers to machine learning techniques for training the model in a ranking task. Learning to rank is useful for many applications in Information Retrieval, Natural Language Processing, and Data Mining. Intensive stud-ies have been conducted on the problem and significant progress has been made. This short paper gives an introduction to learning to rank, and it specifically explains the fundamental problems, existing approaches, and future work of learning to rank

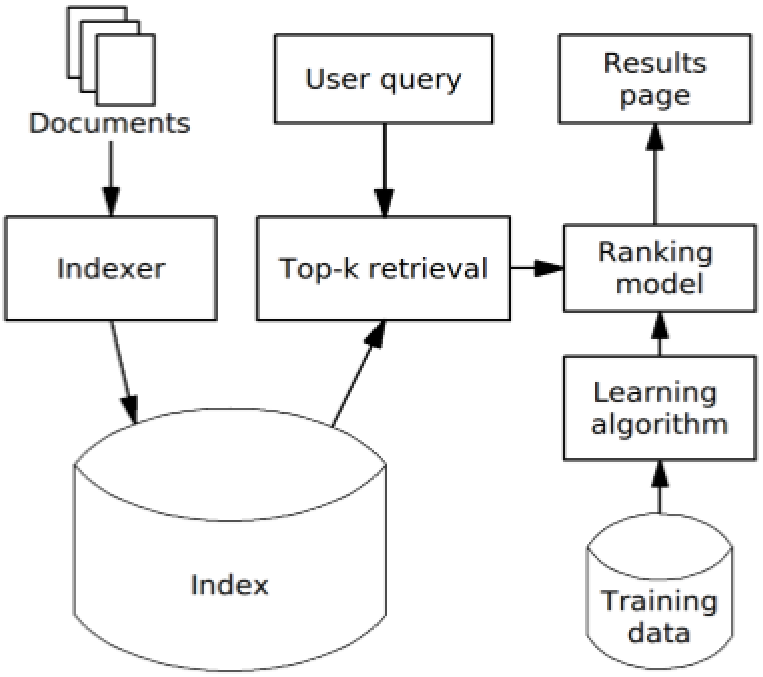

Fig 1: Overview of Ranking System

1.6.1 Weaknesses

1. Prior systems needed lot of human efforts and time.
2. Cost of hiring is high.
3. Potential candidate may loose the opportunity because of ambigious keyword matching.
4. Resumes needed to be in specific format.

1.6.2 How to overcome

1. Use of NLP to read resumes allow candidates the freedom to choose any format that's avalible to them.
2. Machine learning is used to rank candidates in accordiance to requirements Which reduces the efforts of sorting thousands of resumes.
3. Use of NLP can be used to get mean out of ambigious data.
4. Five benifits of A.I. -Goes Beyond Key Words, Fast and Accurate, Perfect For the New World of Social Recruiting ,Customizes to your Needs, Gets Smarter

Chapter 2 Case Study on talent acquisition

2.1 First Generation Hiring Systems

In this System the Hiring team would publish their vacancies and invite applicants. Methods of publishing were newspaper, television and mouth. The interested candidates would then apply by sending there resumes. These resumes were then received and sorted by the hiring team and shortlisted candidates were called for further rounds of interviews. The whole process would take lot of time and human efforts to find right candidate suitable for their job roles.

2.2 Second Generation Hiring Systems

As the industries have grown, there hiring needs has rapidly grown. To serve this hiring needs certain consultancy units have came into existence. They offered a solution in which the candidate has to upload their information in a particular format and submit it to the agency. Then these agencies would search the candidates based on certain keywords. These agencies were middle level organizations between the candidate and company. These systems were not flexible as the candidate has to upload there resume in a particular formats, and these formats changed from system to system.

2.3 Third Generation Hiring Systems

This is our proposed system, which allow the candidates to upload their resumes in flexible format. These resumes are then analyzed by our system, indexed and stored in a specific format. This makes our search process easy. The analyzing system works on the algorithm that uses Natural Language Processing, sub domain of Artificial Intelligence. It reads the resumes and understands the natural language/format created by the candidate and transforms it into a specific format. This acquired knowledge is stored in the knowledge base. The system acquires more information about candidate from his social profiles like Linkedin and Github and updates the knowledge base. Ranking

Attributes are:

1) Current Compensation

2) Expected Compensation

3) Education

4) Specialization

5) Location

6) Earliest Start Date

7) Work Gap

8) Total Experience

9) Relevant Experience

10) Communication

11) Current Employer

12) Stability

13) Education Gap

Chapter 3 Requirement Analysis

The tutorial is intended to be accessible for enthusiasts, engineers, and data scientists at all skill levels. The only skills that you will need are a basic understanding of Python and enough knowledge of the command line to setup a project.

3.1 Software Requiremtents

The software reuirements in this project include:

• Python

• NLTK

• Machine Learning

Python:
Python is used for creating backbone structure. Python is intended to be a highly readable language. It is designed to have an uncluttered visual layout, it uses whitespace indentation, rather than curly braces or keywords. Python has a large standard library, commonly cited as one of Python's greatest strengths.

WebCrawlers: **BeautifulSoap (Python Package)**
BeautifulSoap is an application framework for crawling web sites and extracting structured data which can be used for a wide range of useful applications, like data mining, information processing or historical archival. Even though BeautifulSoap was originally designed for web scraping, it can also be used to extract data using APIs (such as Amazon Associates Web Services) or as a general purpose web crawler. BeautifulSoap is controlled through the BeautifulSoap command-line tool, to be referred here as the "BeautifulSoap tool" to differentiate it from the sub-commands, which we just call "commands" or "BeautifulSoap commands". The BeautifulSoap tool provides several commands, for multiple purposes, and each one accepts a different set of arguments and options.

Natural Language Processing Tool : **Natural Language Toolkit (NLTK) (Python Package)**
NLTK was originally created in 2001 as part of a computational linguistics course in the Department of Computer and Information Science at the University of Pennsylvania. Since then it has been developed and expanded with the help of dozens of contributors. It has now been adopted in courses in dozens of universities, and serves as the basis of many research projects. NLTK was designed with four primary goals in mind:
Simplicity : To provide an intuitive framework along with substantial building blocks, giving users a practical knowledge of NLP without getting bogged down in the tedious house-keeping usually associated with processing annotated language data .

Consistency : To provide a uniform framework with consistent interfaces and data structures, and easily guessable method names .
Extensibility : To provide a structure into which new software modules can be easily accommodated, including alternative implementations and competing approaches to the same task.
Modularity : To provide components that can be used independently without needing to understand the rest of the toolkit.A significant fraction of any NLP syllabus deals with algorithms and data structures. On their own these can be rather dry, but NLTK brings them to life with the help of interactive graphical user interfaces that make it possible to view algorithms step-by-step. Most NLTK components include a demonstration that performs an interesting task without requiring any special input from the user. An effective way to deliver the materials is through interactive presentation of the examples in this book, entering them in a Python session, observing what they do, and modifying them to explore some empirical or theoretical issue.

Machine Learning tool : Scikit-learn (Python Package)
It is a Python module integrating classic machine learning algorithms in the tightly-knit scientific Python world (numpy, scipy, matplotlib). It aims to provide simple and efficient solutions to learning problems, accessible to everybody and reusable in various contexts: machine-learning as a versatile tool for science and engineering.
In general, a learning problem considers a set of n samples of data and try to predict properties of unknown data. If each sample is more than a single number, and for instance a multidimensional entry (aka multivariate data), is it said to have several attributes, or features.
We can separate learning problems in a few large categories:
• Supervised learning , in which the data comes with additional attributes that we want to predict .This problem can be either: –classification: samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data. An example of classification problem would be the digit recognition example, in which the aim is to assign each input vector to one of a finite number of discrete categories. – regression: if the desired output consists of one or more continuous variables, then the task is called regression. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight.
• Unsupervised learning , in which the training data consists of a set of input vectors x without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or thee dimensions for the purpose of visualization.

3.2 Supportive Operating Systems :
The supported Operating Systems for client include:
• Windows 7 onwards
• Linux any flavour.

Chapter 4 System Overview and Design

4.1 Project Design

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization. Once the software requirements have been analyzed and specified the software design involves three technical activities design, coding, implementation and testing that are required to build and verify the software. The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made..

The system uses information extraction technique to parse job descriptions and resumes, and it gets information such as skills, job titles and education background. The information is used to create the models of job openings and job seekers. A domain specific ontology is used to construct the knowledge base, which includes the taxonomies that support resume-job matching.

The models of resume includes job seekers specialties, working experience and education background, and all the fields are extracted from their resumes. The job models are extracted from job descriptions, and they have the same information fields as the resume models. When a job seeker searches the jobs by their resume, the system calculates the similarity between the resume model and job models, then gives every job model a similarity value.

Our system follows the three tier architecture . First tier consist of GUI, Processing block and the Database.

GUI: The GUI(Graphical User Interface) in our project deals with the interface for the user where the user will submit his resumes and job description in any format(pdf, doc, docx,ect.).The GUI provides a platform for the user to communicate with the database. It acts as a connector as well as communicator which connects the database and helps in transfer of data between the GUI and the database.

Processing block: Processing block is the block where the actual processing of our project is done. This block connects the gui to the database i.e. it acts as a connector as well as communicator which connects the database and helps in transfer of data between the gui and the database. Its main function is to take input from resumes, convert them to text and parse it to store the information in database. After storing this information this system will give output using web application.

Database: Database tier is the tier used for the storage of data. This tier contains all the data that is need for the processing of the whole project. The data in this tier is related to the student information gathered form his/her resumes.

4.2 System Architecture

Figure shows the architecture of the whole system, which includes such modules:

1. The Web Crawler can access and download resume's from indeed.com everyday.

2. The Job Parser can parse job description given by recruiter, extract the information and create the job model.

3. The Resume Parser is much like the Job Parser; it parses the resumes,convert all resume files to text file , summarize every resume content and creates the resume models.

4. All the job models and resume model are stored in the database.

5. After that machine learning techniquues are applied to rank the resume's acoording to job description.

Information Extraction is the task of automatically extracting structured information such as entities, relationships between entities, and attributes describing entities from unstructured sources [42]. The IE framework in our system uses six stages in order to extract the information from job descriptions: HTML parsing,segmenting, preprocessing, tokenizing, labeling and pattern matching, which is show in Figure .

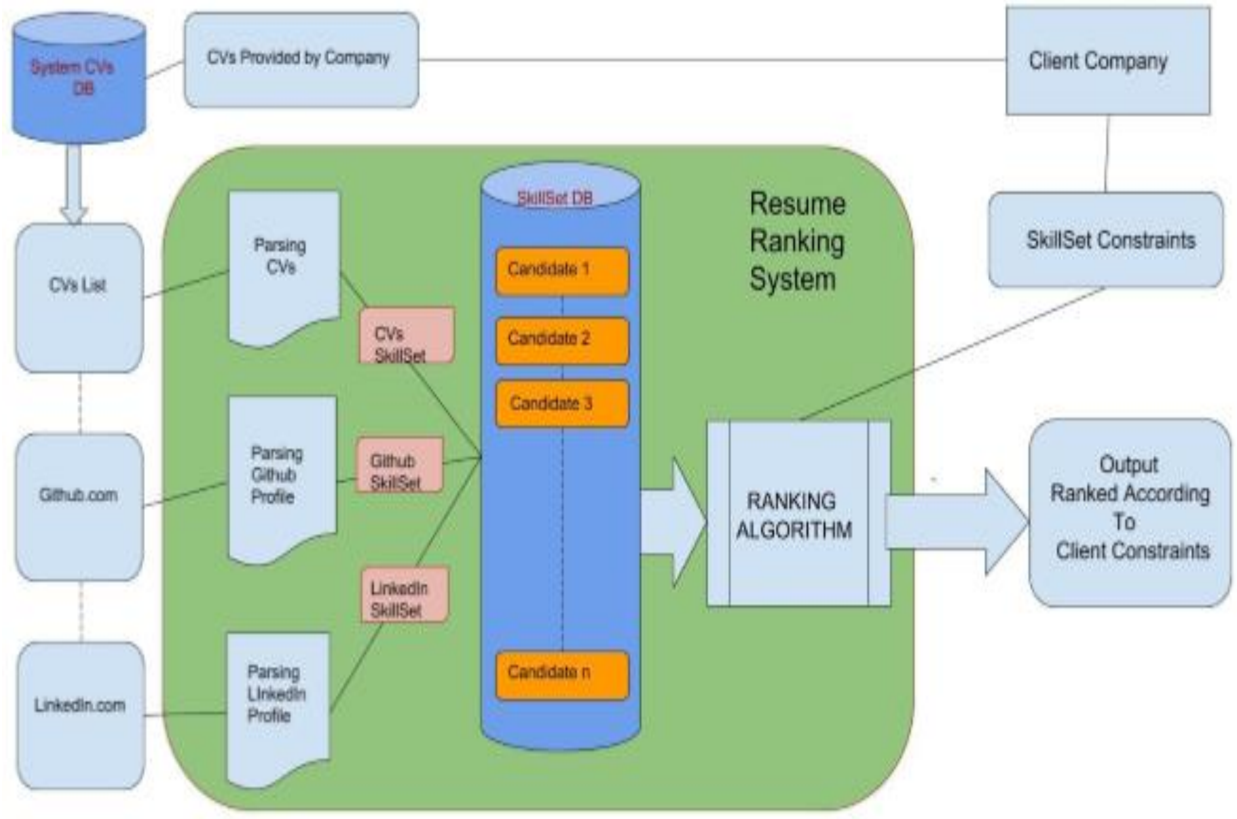

Fig 2: System Architecture

Chapter 5 Implementation

5.1 Working

1) The HTML Parsing will parse the web pages that contain job descriptions, which are obtained from web crawler. The parser uses HTML tag template to extract attributes of the jobs, like job title, location, company name, content and so on. A job will be saved as a record with these attributes in the database. In the record, the content field contains the text part of the job description, which will be processed inlater stages.

2) In the segmenting stage, the content field of the job description is be separated into paragraphs according HTML tags. Then paragraphs are separated into sentences by either HTML tags or punctuation, and after this step, all HTML tags will be removed.

3) Web pages of job description are created in different character sets, (e.g. UTF8 and ISO 8859-1), and almost always contain some unreadable characters. In theprepossessing stage, characters in the sentences are converted to ASCII characters,unreadable characters will be deleted, and some punctuation will be replaced by spaces (e.g. / and -).

4) Now resume are downloaded by webcrawler or one can directly upload resume through GUI interface.

5) Every resume file is then converted to a text file and data of each resume is summarized for fast processing. Summarization is performed to remove stopwords and special characters. This is done using NLTK corpus dictionary of stopwords.

6) In the tokenizing stage, the sentences of job description will be tokenized into arrays of tokens by NLTK [5] and TFIDF Vectorizer.This formed array is named as Item-Feature matrix.

7) Now each resume is taken and tokenized using TFIDF Vectorizer. A new 2-D array is formed in which each row contains tokenized items of single resume. Column attributes in this array is features of item feature matrix. This array is named as User-Feature matrix

8) In the pattern matching and ranking stage, the KNN algorithm and Content Based Filtering algorithm is used to match the user features with the features of job description.

9) After this resume are ranked in decending order seperately by each algorithm and the output is shown on GUI interface.

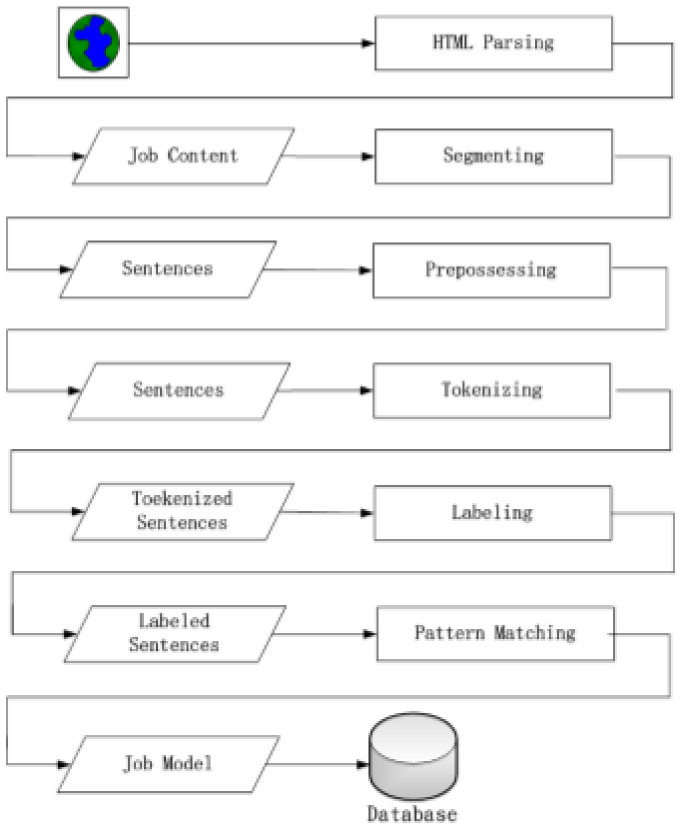

Figure 3: Job Description Process Pipeline

We will describe some implementation details here. The whole system is implemented in Python and uses some third party libraries and frameworks. We used Flask, a lightweight web framework, to build the web application. We used pdf to text file parser, Python Lex-Yacc (PLY) as the token regular expression compiler, and Beautiful Soup as the HTML parser. All the jobs and resumes retrieved by the Web Crawler are stored database. For the natural language processing procedure,we used Natural Language Toolkit (NLTK), a natural language processing library,to extract and tokenize the sentences.

5.2 Information Extraction

In this chapter we will explain how the Information Extraction (IE) module of our system extracts information from these unstructured data source. An example of job description is shown in Table. The IE framework will be introduced by example of processing the job descriptions and resumes. The Gensim, which is used as summarization tools, will be introduced as well.

Gensim : is a robust open-source vector space modeling and topic modeling toolkit implemented in Python. It uses NumPy, SciPy and optionally Cython for performance. Gensim is specifically designed to handle large text collections, using data streaming and efficient incremental algorithms, which differentiates it from most other scientific software packages that only target batch and in-memory processing. Gensim includes implementations of tf-idf, random projections, word2vec and document2vec algorithms, hierarchical Dirichlet processes (HDP), latent semantic analysis (LSA, LSI, SVD) and latent Dirichlet allocation (LDA), including distributed parallel versions

TF-IDF Vectorizer: In information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. Nowadays, tf-idf is one of the most popular term-weighting schemes; 83% of text-based recommender systems in the domain of digital libraries use tf-idf.

Variations of the tf–idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. tf–idf can be successfully used for stop-words filtering in various subject fields, including text summarization and classification.

One of the simplest ranking functions is computed by summing the tf–idf for each query term; many more sophisticated ranking functions are variants of this simple model.

5.2.1 Basic Features

Our basic set of features are based on the fields of the indexed documents and the queries: Most of these features can be described as “match” features, i.e. they denote whether the value(s) of a field given in the query matched the value(s) of the same field in the document. This information is usually returned by the search engine in the form of a score for each indexed field, which can be directly used as a feature.

5.2.2 Match Features

Match features are crucial query-document features, i.e. they depend both on the query and the document, hence, giving an indication about how well a document matches a specific query. Table shows the fields for which we have defined match features. For the fields that can have multiple values, e.g. compskills, we concretely defined several match features: 1) the match score averaged over all values given in the query, 2) the maximum match score (only relevant for the float score variant) and 3) the sum of all match scores.

In addition to query terms associated with certain fields most of our queries also contain fulltext search terms, i.e. terms that can match in any field. Hence, we also defined match features for these terms.

FIELD	MEANING IN CV	MEANING IN QUERY
Jobtitle	recent (< 5 years) job title(s)	posted job title
Jobcodes	recent job code(s)	posted job code
Job groupids	recent job group id(s)	posted job group id
Job class	recent job class(es)	posted job class
jfsector	last industry	posted industry
compskills	computer skills	required computer skills
location	city of residence	posted location
edulevel	highest education level	education requirement(s)
langskills	known languages	required languages

Table 1: Fields for which “match” features are extracted if available

5.3 Ranking Using K-Nearest Neighbour

We employ a K-Nearest Neighbor method for job description based ranking of Resumes. For each training job description, we define a feature vector and represent it in the query feature space (a Euclidean space). Given a new resume, we try to find the k closest training queries to it in terms of Euclidean distance. We then train a local ranking model online using the neighboring training queries (denoted as Nk(q)) and rank the documents of the test query using the trained local model. For local model training, we can in principle employ any existing learning to rank algorithm.

We call the corresponding algorithm 'KNN'. Figure illustrates the workings of the algorithm where the square denotes test query q, triangles denote training queries, and the large circle denotes the neighborhood of query q. The details of the algorithm are presented in Figure. Needless to say, the query features used in the method are critical to its accuracy. In this paper, we simply use the following heuristic method to derive query features and leave further investigation of the issue to future work. For each query q, we use a reference model to find its top T ranked documents, and take the mean of the feature values of the T documents as a feature of the query. For example, if one feature of the document is tf-idf, then the corresponding query feature becomes average tf-idf of top T ranked documents of the query. If there are many relevant documents, then it is very likely that the value of average tf-idf would be high.

Algorithm: KNN

Input:
(1) A test job description and the associated documents to be ranked.
(2) Training data {S qi , i = 1, ..., m}.
(3) Reference model hr .
(4) Number of nearest neighbors k.

Output: Ranked list for query q.

pre-processing:
For each training query qi , use reference model hr to find its top T ranked documents, and compute its query features from these documents.
Online training and testing:
(a) Use reference model hr to find the top T ranked documents for query q, and compute q's query dependent features from these documents.
(b) Within the training data find k nearest neighbors of q, denoted as Nk(q), with distance computed in the query feature space.
(c) Apply hq to the documents associated with query q, and obtain the ranked list.

5.4 Content-based filtering approach

Content-based filtering (CBF) is treated as information retrieval problem or machine learning problem. In information retrieval problem, the document representations have to be matched to user representations on textual similarity while, in machine learning problem, the textual content of the representations are combined as feature vectors, which are used for training a prediction algorithm There are two main tasks related to CBF recommender systems, the User profiling and the Item representation. User profiling is one of most challenging tasks in CBF recommender systems that deal with acquiring, extracting A and representing the features of users. However, the user interface can easily be created to assist users building their profiles. classified the profile information into two types:
(1) the users preferences such as item description that interest the user. There are many possible representations of this description, but the common representation is using a function to predict the possibility of user is interested in that item.
(2) The users interactions history with the recommendation system that includes saving the items that a user has viewed with information about users interaction.
Now that we have learned all these concepts, let's actually try to build a simple content based recommender based on job description. Hence, having a recommender system would help.

For binary representation, we can perform normalization by dividing the term occurrence(1/0) by the sqrt of number of attributes in the resume. Hence, for resume 1: normalized attribute = 1/sqrt(3) = 0.577.

A question that you must ask here is: what happened to TF? and why did we directly do normalization before calculating the TF scores? If you calculate TF without normalization, the TF scores will be $1+\log_{10} 1 = 1$ for attributes that occur and simply 0 for attributes that don't. And, thus the TF scores will also becomes 1/0.

The dot product of job description vectors and IDF vectors of resumes gives us the weighted scores of each resume. These weighted scores are again used for a dot product with the user profile vector (user 1 here)..

This concept can be applied to 'n' resume and we can find out which resume is best . Therefore, along with new resume in a week, a separate recommendation can be made to a particular job description.

In this article, we have seen two approaches to content based recommenders. They both use the TF-IDF weighting and vector space model implementations, albeit in different ways. So while the count data helped us understand the methodology of calculating the weighted scores of resume, the binary representation helped us understand how to calculate the scores for data represented as 1/0 and we also saw how ranking are generated and predictions are made based on that.

Chapter 6 Testing and Evaluation

6.1 KNN Performance

We compared the proposed KNN methods with the baselines of the single model approach (denoted as Single) and the query classification based approach (denoted as QC). For the second baseline, we implemented the query type classifier proposed in [26] to classify queries into three categories (topic distillation, name page finding, and home page finding). Then we trained one ranking model for each category. For a test query, we first applied the classifier to determine its type, and then used the corresponding ranking model to rank its associated documents.We make the following observations from these results:

(1) The better results of KNN over Single indicate that query dependent ranking does help, and an approach like KNN can indeed effectively accomplish the task.

(2) The superior results of KNN to QC indicate that an approach based on soft classification of queries like KNN is more successful than an approach based on hard classification of queries like QC.

(3) QC cannot work better than Single, mainly due to the relatively low accuracy of query classification. In fact, the accuracies of classification in terms of F1 measure are only about 60% in the two datasets.

Errors in the query classification can greatly damage the results of document ranking. This also shows that it is not easy to develop a query dependent ranking method that can beat conventional ranking methods. In contrast, the KNN methods can successfully leverage the ranking patterns of similar queries and achieve better ranking performances.

Effects of Different k Values

We tested the performances of the KNN methods with different values of parameter k, the number of nearest neighbors selected. Notice that when k = m, KNN becomes equivalent to Single, where m denotes the number of training queries. Figure 8 shows the performances of the proposed methods on Dataset 1 with different k values in terms of NDCG@5 and NDCG@10. From this figure, we can see that as k increases, the performances first increase and then decrease. More specifically,

(1) When only a small number of neighbors are used, the performances of KNN are not so good due to the insufficiency of training data.
(2) When the numbers of neighbors increase, the performances gradually improve, because of the use of more information.
(3) However, when too many neighbors are used (approaching 1500, which is equivalent to Single), the performances begin to deteriorate. This seems to indicate that query dependent ranking can really help.

6.2 Content Based Filtering Performance

The very poor performance of the word-only based approach tf-idfm has been omitted .The counter-performance of the baseline metrics LSAm on the top of the LSA representation is blamed on the poor overlap of the job announcement and CV vocabularies. The limitations of the LSA representation can be alleviated using the convolutional metrics (legend LSAu), with an improvement of circa 25%. Two main lessons are learned.

In the evaluation phase, we created a data set of 100 job descriptions that includes several kinds of jobs such as web developers, server back-end developers, mobile developers and so on. We used 5 candidate r´esum´es and retrieved the top 20 jobs. The relevance value of the job descriptions to each r´esum´e were set manually by ten human judge. We created a query q from the job descriptions, treated the text of the resumes as documents d, and applied standard ad-hoc retrieval techniques to rank theresumes. We intended to return jobs that better matched the candidates resumes at the top.

To evaluate the performance of the information extraction module, we extract sentence types through the use of sentence filters. To explain the process of our experiment, we use the sentences whose content pertains to the applicant's college degree information.

In the experiment, we selected 100 sentences from existing job descriptions, and the content of these sentences were requirements of candidate degree and major. We labeled the values for ”degree” and ”major” manually. We use some content patterns that we can identify from these sentences to match and extract the degree information. When we used 6 patterns, the accuracy of ”degree” became 94%. We also compared our pattern matching method to the conditional random fields (CRFs) model [22], which is a state of art machine learning model for sequence labeling. We used 200 labelled sentences to train the CRF model, and the features of the CRF model are words in the sentences and part of speech tags of the words. The accuracies of information extraction of the three fields with our two methods, pattern matching, and the application of the CRF model.

The fact that 80% of the relevant resumes are found in the top-200 recommended resumes is very satisfactory from the application viewpoint, as each recruiter (resp. job seeker) commonly examines several hundred CVs (resp. job announcements). This good result confirms the quality and representativity of the data. As already said however, the collaborative filtering setting does not correspond to the general setting of the job matching application, that mostly considers new job announcements and CVs.

Chapter 7 Conclusion and Future Work

7.1 Conclusion

This work has made an extensive effort to provide a system through which the resumes can be ranked with maximum efficiency. By implementing this system, the task of obtaining the most relevant resumes can be achieved which will save the recruiter time to manually select appropriate resumes, which even after processing may not be a complete fit for the profile. Since, there are multiple levels of screening involved in order to find the most relevant resumes; the accuracy of the system also improves. Thus, using this ranking technique, we can obtain the best results for obtaining the ideal resumes. This approach will automate as well as speedup the process of the HR recruiters.

In the system, job descriptions and resumes are parsed into job models and resume models by the information extraction module. When searching the jobs by a resume, similarity values between the resume model and job description models are calculated in the ontology matching module. The result is sorted by the ontology similarity scores, which are the sum of similarities of different fields multiplied by their weights.

We made such contributions in our works:

1. We developed a resume - job matching system.

2. We used a TFIDF based matching tool to extract information from unstructured data source, which is a lightweight and flexible library, and can be extended in very easy ways.

3. We developed a semi-automatic approach, which can collect technical terms from hr data sources, and by which we created a domain specific ontology for recruitment.

4. We developed statistical-based ontology similarity measure, which can measure the similarities between technical terms .

In the experiment phase, we evaluated the accuracy of information extraction.

We calculated the ontology similarity with the NDCG. Finally, we also tested the performance of resume job matching algorithm NDCG, which showed that our algorithm can achieve a better searching result than other information retrieval models like TF-IDF . We also compared our system with the commercial ranking system, and the results showed that our system can return better results.

7.2 Future Work

Finding a job is a complex process, affected by both explicit and implicit factors. Our work establishes the validity of using information extraction techniques to create a more personalized job matching system, with ample potential for improvement in the future.

First we can introduce a more complex job and resume model to improve performance of the system. In the resume model, we can consider hiring history and project experience of the job seekers. To improve the job description model, job responsibilities and company characteristics (size, dress code, etc.) should be considered as well.

Second, to improve searching speed of our system, we can reduce the number of comparison by filtering out jobs that are clearly not related to resumes. The systemcan classify the jobs into some different subsets, when searching jobs, the system only need to calculate the similarity between the resume and according subset of jobs. Resume Screening System is a content based recommendation system that is mostly focused on comparing the similarities between the resume and a relevant job description.

In future work, we could introduce a hybrid recommendation system that would take advantage of other recommendation algorithms such as Collaborative Filtering. Future work on this system would place greater consideration on job seeker's personal preference like job location, career development plan, and company background

The systems designed so far extracts all the information about the candidate only through his/her resume and after extraction it stores the information in a centralized database, finally ranking them and giving the top 50 results to the HR recruiter according to their specifications. Future advancements in this system can be as follows:

1. The profiles of the candidates can be tracked on social media sites as this will help in analyzing the personality of the candidate and whether he/she is a perfect fit for the post can be judged.

2. Analysis can be done over the past records of the candidate which will help us determine his expected tenure of work in the organization.

References

[1] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, and Raghu Ramakrishnan. Content recommendation on web portals. Communications of the ACM, 56(6):92–101, 2013.

[2] Shaha T Al-Otaibi and Mourad Ykhlef. A survey of job recommender systems. International Journal of the Physical Sciences, 7(29):5127–5142, 2012.

[3] Y. Bachrach. Human judgments in hiring decisions based on online social network profiles. In Data Science and Advanced Analytics, pages 1–10, 2015.

[4] Y. Bengio, A. C. Courville, and P. Vincent. Representation learning: A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell., 35(8):1798–1828, 2013.

[5] J. Bennett and S. Lanning. The netflix prize. In Proc Int. Conf. on Knowledge Discovery and Data Mining (KDD) Cup and Workshop, page 35, 2007.

[6] Jacob Bollinger, David Hardtke, and Ben Martin. Using social data for resume job matching. In Proceedings of the 2012 Workshop on Data-driven User Behavioral Modelling and Mining from Social Media, DUBMMSM '12, pages 27–30, 2012.

[7] Fernando Diaz, Donald Metzler, and Sihem Amer-Yahia. Relevance and ranking in online dating systems. In Proc. Int. ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '10, pages 66–73, 2010.

[8] Richard Doherty. Getting social with recruitment. Strategic HR Review, 9(6):11–15, 2010.

[9] E. Faliagka, K. Ramantas, A. Tsakalidis, and G. Tzimas. Application of machine learning algorithms to an online recruitment system. In Proc. International Conference on Internet and Web Applications and Services, 2012.

[10] Frank Färber, Tim Weitzel, and Tobias Keim. An automated recommendation approach to selection in personnel recruitment. AMCIS 2003 Proceedings, page 302, 2003.

[11] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. Commun. ACM, 30(11):964–971, November 1987.

[12] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. Domain-adversarial training of neural networks. JMLR, to appear in 2016.

[13] Rémy Kessler, Nicolas Béchet, Mathieu Roche, Juan-Manuel Torres-Moreno, and Marc El-Bèze. A hybrid approach to managing job offers and candidates. Inf. Process. Manage., 48(6).
[14] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. Computer, (8):30–37, 2009.

[15] V. Senthil Kumaran and A. Sankar. Towards an automated system for intelligent screening of candidates for recruitment using ontology mapping expert. Int. J. Metadata Semant. Ontologies, 8(1):56–64, 2013.

[16] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 9(Nov):2579–2605, 2008.

[17] Jochen Malinowski, Tobias Keim, Oliver Wendt, and Tim Weitzel. Matching people and jobs: A bilateral recommendation approach. In Proc. Annual Hawaii International Conference on System Science, volume 6, pages 137c–137c. IEEE, 2006.

[18] T Mine, T Kakuta, and A Ono. Reciprocal recommendation for job matching with bidirectional feedback. In Advanced Applied Informatics (IIAIAAI), 2013.

[19] Ioannis Paparrizos, B. Barla Cambazoglu, and Aristides Gionis. Machine learned job recommendation. In Proc. ACM Conference on Recommender Systems, RecSys ’11, 2011.

[20] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, 290(5500):2323–2326, 2000.

[21] Tuukka Ruotsalo, Giulio Jacucci, Petri Myllymäki, and Samuel Kaski. Interactive intent modeling: Information discovery beyond search. Commun. ACM, 58(1):86–92, 2014.

[22] G. Strang. Linear Algebra and its Applications. Academic Press, New York, 1980.

[23] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009:4, 2009.

[24] Xing Yi, James Allan, and W. Bruce Croft. Matching resumes and jobs based on relevance models. In Proc. SIGIR Conference on Research and Development in Information Retrieval, pages 809–810. ACM, 2007.

[26] Dekang Lin. An information-theoretic definition of similarity. In ICML, volume 98, pages 296–304, 1998.

[27] Ping Liu and Peter Dew. Using semantic web technologies to improve expertisematching within academia. Proceedings of I-Know (Graz, Austria, pages 370–378, 2004.

[28] Yao Lu, Sandy El Helou, and Denis Gillet. A recommender system for job seeking and recruiting website. In Proceedings of the 22nd international conferenceon World Wide Web companion, pages 963–966. International World Wide WebConferences Steering Committee, 2013.

[29] Jochen Malinowski, Tobias Keim, Oliver Wendt, and Tim Weitzel. Matching people and jobs: A bilateral recommendation approach. In System Sciences,2006. HICSS’06. Proceedings of the 39th Annual Hawaii International Conference on, volume 6, pages 137c–137c. IEEE, 2006.