Potential Candidate Selection Using Information Extraction and Skyline Queries

Chapter · March 2020

DOI: 10.1007/978-3-030-43192-1_58

3 authors, including:

Farzana Yasmin
University of Houston
3 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Potential Candidate Selection using Information Extraction and Skyline Queries

Farzana Yasmin, Mohammad Imtiaz Nur, and Mohammad Shamsul Arefin
Computer Science and Engineering, Chittagong University of Engineering & Technology, Chattogram-4349, Bangladesh

farzanayasminefu@cuet.ac.bd
m.imtiaznur@gmail.com
sarefin@cuet.ac.bd

Abstract. Information extraction is a mechanism for devising an automatic method for text management. In the case of candidate recruitment, nowadays different companies ask the applicants to submit their applications or resumes in the form of electronic documents. In general, there are huge numbers of resumes dropped and therefore the volume of the documents increases. Extracting information and choosing the best candidates from all these documents manually are very difficult and time-consuming. In order to make the recruitment process easier for the companies, we have developed a framework that takes the resumes of candidates as well as the priorities of the employer as input, extract information of the candidates using Natural Language Processing (NLP) from the resumes, filter the candidates according to predefined rules and return the list of dominant candidates using skyline filtering.

Keywords: Information Extraction, natural language processing, machine learning, skyline query, candidate selection.

1 Introduction

Information extraction (IE) infers the process of automatically gisting of information in a structured way from unstructured and/or semi-structured machine-readable documents [1]. Nowadays huge volume of documents are found online and offline. Extracting information from these vast volumes of data manually is time consuming.

Recruitment is the process of searching and selecting best candidates for filling the vacant positions of an organization. Recruitment process requires planning, requirements setup strategy, searching candidates, screening the candidates according to the requirements and evaluation of the candidates. These steps are usually conducted by the Human Resource (HR) department of any company.

Whenever there is a job opening for the vacant positions, large amount of applications are dropped. Searching and screening the best candidates from these applicants after assessing the abilities and qualifications manually takes huge amount of time, cost and effort of the HR department as the volume of data are big. If we can develop an efficient system for extracting information from the resumes of the applicants and process these information in an automated way, it will ease the work of the HR management. An automated system for choosing the potential candidates that best suit the position's requirements can increase the efficiency of the HR agencies greatly. Therefore, in order to make the recruitment process easy, effective and automated, we have developed a framework of potential candidate selecting system by choosing a domain of document information extraction i.e. the CV/resume documents. This development task involves the information extraction based on natural language processing i.e. tokenization, named entity recognizer (NER) and utilizes skyline query processing which works well in filtering the non-dominating objects from database and also makes a new addition to this domain. So the objectives of the system development can be summerized as follows:- 1) To design an efficient information extraction system from documents like curriculum vitae, 2) To generate scores on different features based on extracted information, 3) To perform appropriate filtering of information using skyline queries and 4) To generate proper ranking system for candidate selection.

The rest of the paper is presented as follows: In Section II related works of the candidate ranking system development has been portrayed. The system architecture and design is elaborated in Section III. Section IV represents the implementation of our work with some experimental results. And finally, a conclusion over the work has been drawn in section V.

2 Related Work

D. Celik [2] proposed an information extraction system for candidate selection where the information extraction was based on ontology. The proposed methodology used Ontology-based Resume Parser(ORP) to convert English and Turkish documents into ontological format and constructed seven reference ontologies to extract the information and categorize them into one of these ontologies. Though the methodology worked good on information extraction but it did not describe any score generation mechanism to rank the candidates. R. Farkas et. al. [3] worked on a method of extracting information for career portal where the information of applicants' are stored in a uniform data structure named HR-XML format. They used a CV parser to automatically extract data from the CV. In [4], the authors used a hybrid cascade model for information extraction from CVs. In the first pass, the proposed method segments resume using Hidden Markov Model. The second pass uses HMM and SVM to extract further detailed information. The cascaded pipeline suffers from error propagation i.e. errors from first step are passed in the second pass and the precision and recall value of the second pass decreases

subsequently. Information is extracted from resumes using basic techniques of NLP like word parsing, chunking, reg ex parser in [5]. Information like name, email, phone, address, education qualification and experience are extracted using pattern matching in this work. Some other online resume parsers are found in [6] & [7]. There also have been developed some works using skyline queries. [8], [9] & [10] describes some algorithms for processing skyline queries with their implementation. S. Patil et. al. [11] developed a method for learning to rank resumes with the help of SVM rank algorithm. In [12], X. Yi et. al. applied a Structured Relevance Model to select resumes for a given post or to choose the best jobs for a given candidate based on their CV. In [13] job narration are transformed into queries and lookup in database is performed. The top-ranked candidates gets selected automatically from these queries. Some authors exploit additional information like social media information along with information gained directly from resumes in [14] & [15]. Another form of candidate selection was proposed by S. Kumari et. al. [16] where candidate selection was done by using Naïve Bayes algorithm for classifying the candidate profiles. They also considered employers importance criteria. No description given of how the information extraction are done. Also it requires GPRS connection every time as it is online based. In [17], CVs are filled in a predefined format and the scoring and ranking process is based on Analytic Hierarchy Process (AHP).

Though many works have been developed for candidate ranking, the use of skyline query in this sceneraio is relatively new approach and we have implemented this novel approach in our framework.

3 System Architecture and Design

The proposed framework works in 4 modules according to Fig. 1: Document processing module, Query Execution Module, Analysis & Output module and Storage module. The details of the system architecture are described below [Reviewer Comment 1,2,3]:

3.1 Processing Module

Document Input. First we will need to input the resumes in the interface for a specific job id. After documents are being fed to the system in processing module, information extraction process begins and we used a NLP module named spaCy [18] for the rest of the processing steps. Suppose, we have fed the resumes of Fig. 2 in the system.

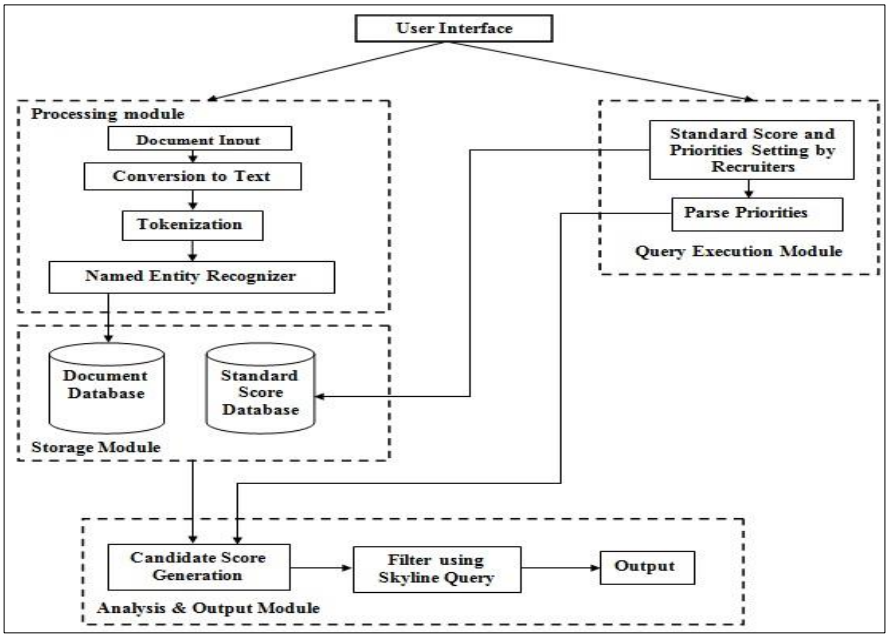


Fig. 1. System Architecture of Potential Candidate Selection

Adam Wang (Male)

XXXX Company of Bejing,
Beijing City, 100007
1364-110-XXX
wangXXX@hotmail.com

Education Background

From Sept. 2000 to Apr. 2003, I got master degree from University of XXX in computer software engineering.

From Sept. 1996 to July. 2000, I got bachelor degree from School of XXX and major in computer science and technology.

Experience

From March 2003 to now, working on Human Face Recognition System in XXXX Company of Beijing

From June 2001 to March 2003, working on Content-Based Intelligent Image Retrieval System in Research Center of XXX Company

From Sept. 2000 to May 2001, working on Intelligent Highway Distress Detection System in National Lab. Of XXX University

Interests

Reading, music, and jogging

a)

ABC

Cell: +880 1680671851
E-mail: abc@gmail.com,

RESEARCH INTEREST

Data Mining, Artificial Intelligence, Machine Learning, Algorithm Design, Data Structure.

EDUCATIONAL QUALIFICATIONS

- **B.Sc. in CSE**- October 2015
  CUET, Bangladesh.
  Result: CGPA 3.81 (out of 4.00)
  Class Position: 2nd out of 113 students

WORK EXPERIENCES

- **Chittagong University of Engineering & Technology, Chittagong-4349, Bangladesh**
  - Joined as ***"Lecturer"*** as on 2016
  - Duration : 2016- Present

(b)

Ishraq Rayeed Ahmed
+880 1717 342569, ishraqrayeed@gmail.com, www.irahmed.wordpress.com

Research Interests

Pavement Materials and Design, Urban and Public Transportation System, Traffic Emissions and Air Quality, Transportation Safety, Intelligent Transportation System

Education

Bachelor of Science, Civil Engineering, March 2016
Bangladesh University of Engineering & Technology at Dhaka, Bangladesh
CGPA: 3.24/4.00

Technical Skills

Scientific Computing & Simulation Tools: MATLAB, R Project, ArcGIS, VisSim, EPAnet
Structural Design Software: AutoCAD, SAP, ETABS, GRASP
Programming Languages: C++, Python
Graphics Design Software: Adobe Photoshop, Adobe Illustrator, 3D Studio Max

Publications

• Ahmed, I.R; Mondal, A.R; Noor, A.U, *"Assessment Of Pedestrian Perception Towards Pedestrian Crossing Facilities In Dhaka Metropolitan City: A Study Based On Observation and Survey"*, International Conference on Recent Innovation in Civil Engineering for Sustainable Development (IICSD-2015), DUET, Dhaka.

(c)

Fig. 2. (a), (b), (c) Sample Resumes

Conversion to Text. The standard format of resumes for our system is considered English resumes in PDF format. At first we need to convert the pdf into plain text using UTF-8 encoding. UTF-8 is a compromise character encoding that can be as compact as ASCII but can also contain any Unicode characters. UTF stands for Unicode Transformation Format and '8' means it uses 8-bit blocks to represent a character. The number of blocks needed to represent a character varies from 1 to 4 [19].

Tokenization. After conversion to text, now we have our necessary text file. We start reading the text file and tokenize the whole document. Tokenization is done using the language rule i.e. removing the white space, checking the exception rules like punctuation checking, abbreviation rules etc.

Named Entity Recognition. Named entity recognition (NER) is the most important task to do next. The success of the extraction process mainly depends on the accurately recognized entities from a resume. The subtask of information extraction that seeks to locate and classify named entity mentions in unstructured text into pre-defined categories such as the person names, organizations, email, phone, address, time, quantities, numeric values, etc. can be defined as Named entity recognition [20]. A statistical model is used to classify our desired entities in a standard resume. The NER training model is designed using incremental parsing and residual CNNs. In case of training our model (Fig. 3.) with the desired annotation, we used resumes in JSON format. At first we have to manually annotate our training data in JSON format. Then we load or build the NER model. For training the NER model with our custom entities, now we add the labels for each annotations. Then we shuffle and loop over our training examples. At each word

the model makes a prediction. It then consults the annotations to see whether it was right. If it was wrong, it makes adjustment of the weight so that the correct action will score higher next time. Then we save the model and test it to make sure the entities in the test data are recognized correctly. After the validation of the trained NER model, now we use this model to extract the values of the entities trained from the resumes. The recognized entity values are stored in a row of a table for each candidate in the storage module.

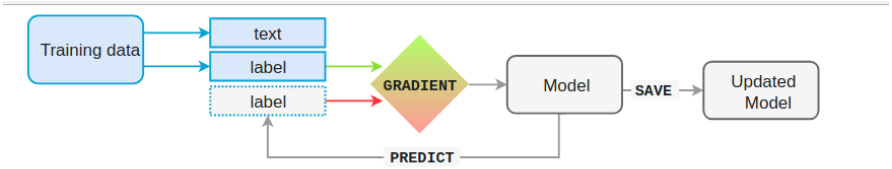


Fig. 3. spaCy's NER model training process (Source: [21])

3.2 Query Execution Module

Standard Score and Requirements Setting by the Recruiter. In the UI, employers set the standard scores required to evaluate the abilities of the candidate according to the job criteria. Each criterion gets a value and a weight for a specific keyword. The weight represents the relative importance or priorities of the specific criteria and value represents the variations of the score of each criteria. Keyword gives the matching criteria i.e. which information to be satisfied for scoring. These standard scores are stored in the storage module as a lookup table. Suppose, for software engineer position, the employer sets the following values and weights in the table for each criteria.

Table 1. Standard Score Setting Table

Job_criteria	Keywords	Value	Weight
Skills	C++	10	5
Skills	Java	10	5
Skills	PHP	8	5
Experience	3	5	3
Experience	0	2	3
Major	CSE	10	2
Major	EEE	6	2

Parse the Requirements. The system will then parse these requirements of the employer in the query execution module.

3.3 Storage Module

Storage module stores information processed by the processing and query execution module. The extracted information table, after the entites are recognized, are stored

in the document database. The standard scores set by the recruiters in the query execution phase are stored in the score database. The total storage is required for the candidate score generation in the analysis and output module.

3.4 Analysis and Output Module

Candidate Score Generation. After parsing the requirement of the employer, the system will start the score table generation of each candidate according to the employer priority and previously set standard score for different categories. The extracted information stored in the lookup table in document database is retrieved and matched with the keywords stored in the job_info_details table. If match found, the corresponding values are calculated by multiplying the value and weight set in the standard score table. If multiple keywords are matched for a specific critria, then they are stored as aggregated sum. For example, if multiple skills match, then all the skill values are added and stored in the skill column for that candidate.

The score calculation follows the following formula (1):

$$Scores[job\_criteria] = Scores[job\_criteria] + (job\_details\ (value) * job\_details\ (weight)) \quad (1)$$

For the result and total years of experience column scoring, extracted numeric value of the applicant is matched with the sorted list of previously set keywords. İf the extracted value is greater or equal to any specified keyword, the score is calculated according to that value. For the publication column, international conference, international journal keywords are searched and matched. If found, the number of occurences are counted. If any column information contains missing value, then they are considered as zero in the score calculation. The calculated score is stored in that specific criteria column of the score table. After being scored in each criteria, now a table is generated which is score of each candidate. The sample score table for the resumes in Fig. 2 are generated as in Table 2.

Table 2. Sample Score Table

CV no.	Skills	Experience	Major	Total
1	50	15	20	85
2	0	15	20	35
3	50	6	0	56

The first candidate had the matching skill C++, experience of 3.7 years and major CSE. So the first candidate fulfills all the requirements of the specified job position and get scores according to the rules set as Table 1 i.e.

Scores[skills] = value for C++ (10) * weight of C++ (5) = 50.
Scores[Experience] = value for 3.7 (5) * weight of C++ (3) = 15.
Scores[Major] = value for CSE (10) * weight of CSE (2) = 20.

The scores of the 2nd and 3rd candidate will be calculated as the 1st candidate. The skills of 2nd candidate doesn't match the required skills and so the missing value is scored as zero. Accordingly, the 3rd candidate's major doesn't match the requirement and so he gets a zero in major field. Now if we select the Major field as mandatory, the row containing zero in this field i.e candidate 3 will be deleted.

Filter using Skyline Query. A skyline is defined as those points in a dataset those can not be worse than any other point. A point dominates other points if it is as greater or equal in all criteria and greater in at least one criteria. A study in [22] states that during the past two decades, skyline queries are applied in several multi-criteria decision support problems. Skyline query utilizes the idea of skyline operator. There are several algorithms for the implementation of skyline operator like using directly in SQL queries, block nested loop, divide and conquer, branch and bound, map reduce etc. We have used the block nested loop (BNL) method. Applying skyline queries on the score table according to employers' priorities, now the dominant applicants will be filtered. We can explain the working procedure of skyline query using Table 2. According to BNL, we compare all the data points with all other points. We keep the points that can dominate other points in all criteria and atleast in one dimension. The points dominated are discarded from the list. Those points are considered to be skyline that dominates are others or maybe a part of the skyline if they neither dominates nor dominated by others. So comparing the data points of Table 2 we find that Candidate 1 dominates the other candidates in every criteria as it contains either equal or higer value in every criteria than the other two. After applying skyline we get that candidate 1 best suits for the job and others are discarded from the list.

Table 3. Score Table after Filtering Using Skyline Query

CV no.	Skills	Experience	Major	Total
1	50	15	20	85

Output Generation. The system output will show the result of the potential candidates after the filtering process. Then the employer can choose the criteria they want to selcect candidates having higher score. Suppose, an employer want to have candidates with higher score in skills criteria. He/she can rank the candidates according to the scores in the skills criteria. If multiple criterias are to be considered, it is also possible to rank the candidates. Then the system can output the ranked output. The output will be sorted according to the score obtained and personal details like name, email, phone number of each candidate will be displayed.

4 Implementations and Experiments

In this section, we have described the implementation and experimental setup of our

system with necessary illustrations.

4.1 Experimental Setup

Potential candidate selection system has been developed on a machine having Windows 10, 2.50GHz Core i5-3210 processor with 12GB RAM. The system has been developed in Python 3.7.3, Asp.Net Core and Angular5 in the front end and MS SQL Server is used in the back end for storing related data to complete this project.

4.2 Implementation

At the beginning of our system workflow, resume documents are fed into the system. All the resumes are stored in a folder according to the specific job id. These resumes are then converted into text format using UTF-8 encoding and stored in a file named lookup.py. The text files are then called for tokenization and named entity recognition. Next, calling the trained model of NER, we extract the information from the tokenized data of the resume documents. We have extracted information of 12 entities- University, degree, major, experience, publication, skill, certification and personal information (name, date of birth, email, phone etc.). The values of these entities are extracted in the the information extraction table and stored in document database.

On the other hand, employers set the necessary information for setting the requirements and scores of each criterion. Job_info_details table holds the columns like Job_info ID, Keyword, Value, Weight, Job Criteria Name i.e. the information set by the recruiters on the score setting step. For the specific job position, extracted information table can be uploaded next for score generation. After scoring according to the rules set, the system generates the score table. This table can be downloaded by the recruiter. Next the recruiter is given the option to choose the mandatory requirement criteria. If any of the criteria is chosen then the candidates holding zero value in that specific criterion are removed before applying skyline query. Applying skyline query on the score table now returns the dominant applicants for the specified job by finding the max value and mapping them according to the job criteria. Then the unique candidates holding maximum values in any of the criteria are returned.

4.3 Performance Evaluation

Potential candidate selection system performance is evaluated in two different phases- Information extraction performance, and filtering using skyline queries. We tested the performance of our system using 150 resumes of engineering background.

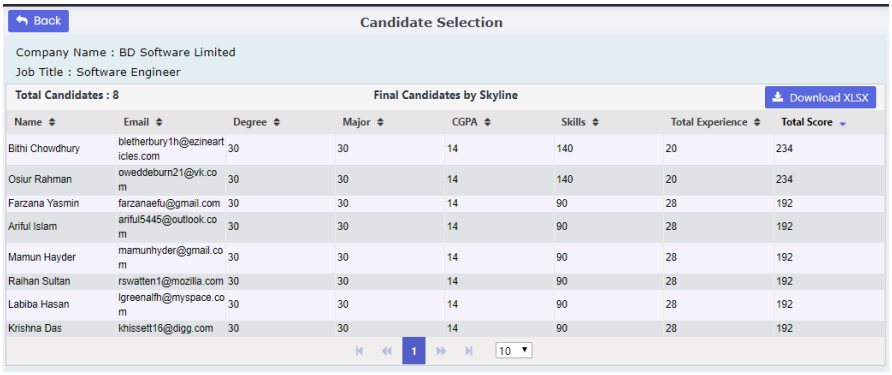
For the training of our NER model, we used a dataset of 300 manually annotated resumes and validated the model using resumes from the dataset. We found some

incorrect values for extracted information and also some missing values. The prcision, recall and F-measure of each entity of the NER model is given below in Table 4:

Table 4. Accuracy, Precision, Recall and F-measure of the Entities Recognized

Entity	Accuracy	Precision	Recall	F- measure
Name	99.77	0.99	0.99	0.99
Email	100.0	1.0	1.0	1.0
Phone	100.0	1.0	1.0	1.0
Date of Birth	99.87	1.0	0.99	0.99
University	99.87	1.0	0.99	0.99
Degree	99.25	0.99	0.99	0.98
Major	98.36	0.99	0.98	0.98
Publication	98.71	0.98	0.98	0.98
Skills	94.84	0.99	0.94	0.96
CGPA	100.0	1.0	1.0	1.0

We have tested the candidate filtering using skyline query with 3 different job criteria- Software Engineer with 2-4 years experience, Research Assistant with CGPA above 3.5 and 2 publications and Assistant Programmer with skills Java, JavaScript, HTML and CSS. We have scored the 150 resumes for these 3 different job positions. The returned output showed that the top scored candidates were different for the 3 job positions as the requirements were placed different. Comparing with the manual ranking, we found that the filtering were accurate with a faster response. So based on the observations we can come to the conclusion that the accuracy of the skyline query depends on the accuracy of the scores generated. If the score generation is accurate, the skyline query returns those candidates that are best for the vacant position quite accurately and within few seconds. A snapshot of the system's output is shown in Fig. 4.



Name	Email	Degree	Major	CGPA	Skills	Total Experience	Total Score
Bithi Chowdhury	blethertbury1h@ezinearticles.com	30	30	14	140	20	234
Osiur Rahman	oweddeburn21@vk.com	30	30	14	140	20	234
Farzana Yasmin	farzanaefu@gmail.com	30	30	14	90	28	192
Ariful Islam	ariful5445@outlook.com	30	30	14	90	28	192
Mamun Hayder	mamunhyder@gmail.com	30	30	14	90	28	192
Raihan Sultan	rswatten1@mozilla.com	30	30	14	90	28	192
Labiba Hasan	lgreenalfh@myspace.com	30	30	14	90	28	192
Krishna Das	khissett16@digg.com	30	30	14	90	28	192

Fig. 4. Snapshot of the High Scored (Total) Candidates for Software Engineer position of our system

We have also tested the skyline filtering with 50,000 synthesized score data. The execution time for different number of resume data is given in Table 5. The table shows that the skyline query can perform filtering in a very responsive way.

Table 5. Response Time of Skyline Filtering

No. of Resume Data	Response Time (mili sec)
1000	67.23
3000	75.30
6000	83.27
25000	91.18
50000	112.80

5 Conclusion

In this paper, we have narrated the idea of a candidate selection system which finds the best potential candidates by extracting information and filtering using skyline query. Automating the total task may help the HR agencies by reducing time, cost and effort of searching and screening the pioneer applicants from vast applications. There are many automated candidate ranking system available online. But we have developed a novel idea of using skyline query in filtering and returning the dominant candidates for the job specified. Skyline queries are mostly applied in multidimensional decision application. In candidate filtering, the implementation of skyline is new and we have applied this novel approach in an efficient manner. In the system performance evaluation, we have used 150 resumes of technical background in testing of the system and found that, the system works in an efficient way of returning best candidates by matching the given requirements with qualifications of the candidates. Altogether the system performs better in filtering the documents as

well as the candidates based on the information extracted from the resume documents. Our system works for only English documents currently. In future, we hope to extend it for Bangla resumes as it is fifth most spoken native language in the world by incorporating Bangla Language Processing and test the system performance accordingly.

References

1. Information Extraction, https://en.wikipedia.org/wiki/Information_extraction
2. Celik, D.: Towards a Semantic-Based Information Extraction System for Matching Resumes to Job Openings. Turkish Journal of Electrical Engineering & Computer Sciences. vol. 24, pp. 141-159 (2016)
3. Farkas, R., Dobó, A., Kurai, Z., Miklós, I., Nagy, Á., Vincze, V., Zsibrita, J.: Information Extraction from Hungarian, English and German CVs for a Career Portal. In: Prasath R., O'Reilly P., Kathirvalavakumar T. (eds) Mining Intelligence and Knowledge Exploration. Lecture Notes in Computer Science, vol. 8891, Springer, Cham (2014)
4. K. Yu, G. Guan, M. Zhou, "Resume Information Extraction with Cascaded Hybrid Model", In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, pp. 499–506, Ann Arbor, June 2005
5. Information Extraction from CV, https://medium.com/@divalicious.priya/information-extraction-from-cv-acec216c3f48
6. Writing Your Own Resume Parser, https://www.omkarpathak.in/2018/12/18/writing-your-own-resume-parser/
7. Resume Parser, https://github.com/bjherger/ResumeParser
8. Shah, S., Thakkar, A., Rami, S.: A Survey Paper on Skyline Query using Recommendation System. In: International Journal of Data Mining And Emerging Technologies, vol. 6, issue. 1, pp. 1-6, ISSN. 2249-3212 (2016)
9. Kalyvas, C., Tzouramanis, T.: A Survey of Skyline Query Processing. 2017
10. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An Optimal and Progressive Algorithm for Skyline Queries. In: ACM SIGMOD International Conference on Management of Data, pp. 467-478 (2003)
11. Patil, S., Palshikar, G.K., Srivastava, R., Das, I.: Learning to Rank Resumes. In: FIRE, ISI Kolkata, India (2012)
12. Yi, X., Allan, J., Croft, W.B.: Matching Resumes and Jobs Based on Relevance Models. In: SIGIR, Amsterdam, The Netherlands, pp. 809–810 (2007)
13. Rode, H., Colen, R., Zavrel, J.: Semantic CV Search Using Vacancies as Queries. In: 12th Dutch-Belgian Information Retrieval Workshop, Ghent, Belgium, pp. 87–88 (2012)
14. Bollinger, J., Hardtke, D., Martin, B.: Using Social Data for Resume Job Matching. In: DUBMMSM, Maui, Hawaii, pp. 27–30 (2012)

15. Dandwani, V., Wadhwani, V., Chawla, R., Sachdev, N., Arthi, C.I.: Candidate Ranking and Evaluation System Based on Digital Footprints. In: IOSR Journal of Computer Engineering (IOSR-JCE), e-ISSN. 2278-0661, p-ISSN. 2278-8727, vol. 19, issue. 1, ver. 4, pp. 35-38 (2017)
16. Kumari, S., Giri, P., Choudhury, S., Patil, S.R.: Automated Resume Extraction and Candidate Selection System. In: International Journal of Research in Engineering and Technology, e-ISSN. 2319-1163, p-ISSN. 2321-7308, vol. 03, issue. 01 (2014)
17. Faliagka, E., Ramantas, K., Tsakalidis, A., Viennas, M.: An Integrated E-Recruıtment System for CV Rankıng Based on AHP. In: 7th International Conference on Web Information Systems and Technologies, Noordwijkerhout, The Netherlands, (2011)
18. spaCy, https://spacy.io/
19. UTF-8 encoding, https://www.fileformat.info/info/unicode/utf8.htm
20. Named Entity Recognition, https://en.wikipedia.org/wiki/Named-entity_recognition
21. spaCy NER training model, https://course.spacy.io/chapter4
22. Tiakas, E., Papadopoulos, A. N., Manolopoulos, Y.: Skyline queries: An introduction. In: 6th International Conference on Information, Intelligence, Systems and Applications (IISA), DOI: 10.1109/IISA.2015.7388053, E-ISBN: 978-1-4673-9311-9**,** July (2015)