

Warana

Recruitment Helper for Enterprises



Department of Computer Science and Engineering

University of Moratuwa

Group Members

100087V W. A. M. de Silva

100162X U. L. D. N. Gunasinghe

100408J W. D. T. P. Premasiri

100487X W. A. D. Sashika

Project Supervisors

Mr. N. H. N. D. de Silva

Dr. A. S. Perera

Date of Submission: 12th February, 2015

Abstract

At present, Natural Language Processing techniques are widely used in almost all the computer applications, in which language related processing procedures are being followed. This area of Language processing related work is still in the process of development. Although researchers and application developers try to embed Natural Language Processing modules increasingly with systems.

Warana Recruitment Helper is one such application which uses Natural Language Processing related technological aspects in order to avoid the difficulties, which are faced by the Recruiters at Enterprises during the employee recruitment process. According to the recent researches and surveys carried out by many parties, they have found that the process of recruitment is trying to minimize the time as possible. Researches show that the recruiters look at the resumes of the candidates a very small amount of time. Statistically this value takes about 6-10 seconds.

We all know nowadays people maintain their online profiles such as LinkedIn, Blogposts, etc. At the recruitment process, recruiters observe these profiles in order to get a wider picture about the candidate. Although this is the trend, this process slows down the recruitment process. When the time required increases, attention paid to capture valuable information further decrease. If this kind of process can be automated and provide analytical details about the candidate in an impressive manner, then the process of recruitment not only becomes easier, but also allow the recruiter to have wider set of details about the candidates.

Warana Recruitment Helper is an application, which pays attention towards afore mentioned facts as well as the best solution for the problem. Warana Recruitment Helper's focus is to map the candidates with the company requirements and generate a comparable score to evaluate the candidates. As a proof of concept of this fact, in this initial solution we are focusing on the domain of IT Industry and evaluate the candidates based on their technical proficiencies compared to the technological requirements of the company.

Acknowledgement

Warana Recruitment Helper was not a success of only the four members of the Warana Research and Development Group. Upon the success of this project, there is a bunch of people who should be given the credit of the success of this project.

First, we would like to thank Mr. Nisansa Dilushan de Silva who was the initial project idea holder for this research work. From the beginning and until the end of the project he guided us throughout the journey. It was not an easy task to complete and to gain the expected outcome without proper guidance. His immense support and guidance was the reason for the success of this project. Even though he was not in the country during the project last four months, he guided us and encouraged us by having weekly meetings and monitoring the project progress and providing useful guidance during the journey.

In addition, our heartfelt gratitude should be given to Dr. Amal Shehan Perera, who was the internal supervisor of this project. Since Mr. Nisansa was not available in the country at the time of the project, on behalf of the team, Dr. Amal hold the responsibility of guiding us by providing required resources and monitoring the progress of the carried out work. We should also pay our respect to him on behalf of the successful completion of the project.

As well as afore mentioned pillars of success we would like to thank Project Coordinator Dr. Malaka Walpola who coordinated the whole project on behalf of the Department of Computer Science and Engineering.

In addition, we would like to thank all of the others who provided us help throughout the time. Specially Mr. Kasun Indrasiri (Computer Architect, WSO2 Lanka Pvt. Ltd), Mr. Nirmal Fernando (WSO2 Lanka Pvt. Ltd) on behalf of providing useful resources for the project as well as all the people who gave us help to identify the pitfalls of the project work and help to overcome them.

Table of Contents

Abstract	ii
Acknowledgement	iii
Table of Contents	1
Table of Figures	4
Table of Equations	6
Table of Tables	8
1. Introduction	9
1.1 Overview	9
1.2 Warana Recruitment Helper for Enterprises	10
2. Literature Review	11
2.1 Introduction	11
2.2 Information Extraction from Resumes	12
2.2.1 Information extraction from data and its importance	12
2.2.2 Resumes and Structure	12
2.2.3 Choosing an information extraction Model	12
2.2.4 Process and Tools	13
2.2.5 Evaluation Methods	14
2.3 Sentence Similarity between Sentences	15
2.3.1 Similarity between Words	15
2.3.2 Semantic Similarity of Sentences	18
2.3.3 Syntactic Similarity of Sentences	20
2.3.4 Hybrid Methods	21
2.4 Skill Recongnition	21
2.4.1 Skills over technologies	21
2.4.2 LinkedIn Endorsements	21
2.4.3 Confirmation of Endorsements	22
2.4.4 Automatic Term Recognition (ATR)	22
2.4.5 TermEx term recognition algorithm - Sclano et. al 2007	23
2.4.6 IBM gloss ex	23
2.4.7 CValue term recognition algorithm. See Frantzi et. al 2000	24
2.5 Measuring graph similarity	25

2.5.1	Graph similarity scoring and matching.....	25
2.5.2	Measuring Similarity of Graph Nodes by Neighbor Matching	27
2.6	Existing Solutions	29
2.7	Summary	30
3.	Design.....	31
3.1	Rationale.....	31
3.2	Use cases of the system.....	31
3.3	Overall System Architecture	34
3.3.1	Information Extractor.....	34
3.3.2	Profile Generator.....	36
3.3.3	Analytical Engine.....	36
3.4	Application Architecture	37
4.	Implementation.....	38
4.1	Information Extraction from Resumes based on Information Grouping Property....	38
4.1.1	Introduction.....	38
4.1.2	Resumes and Identifiable Characteristics	39
4.1.3	Resume Information Extraction Module Architecture	42
4.1.4	Tools used	43
4.1.5	Resume Information Extraction Module Implementation	46
4.1.6	Results and Analysis	51
4.2	Aggregated Profile Generator	57
4.2.1	Introduction.....	57
4.2.2	Information Sources.....	58
4.2.3	Skill Recognition	59
4.2.4	Synergistic Approach to Key Term Extraction.....	62
4.2.5	Analysis of Algorithms	64
4.2.6	Weighted Aggregation of Algorithms	66
4.2.7	Analyzing & improving aggregated algorithm.....	68
4.2.8	Discussion on new Algorithm.....	70
4.3	Extracting technologies from company knowledge base.....	71
4.3.1	SigmaC Document analysis based Automatic Concept Map Generation for Enterprises	71
4.3.2	SigmaC Application Architecture.....	73
4.3.3	Knowledge base about technologies used by company	75
4.4	Measuring the Suitability of the Candidate for the Company.....	78

4.4.1	Graph Similarity Measuring Procedure	79
4.4.2	Implementation of the Graph Similarity Measuring Module	83
4.4.3	Results and Analysis	87
4.5	Statistics Representation	89
4.5.1	Statistics Representation Tools	90
4.5.2	Candidate Statistics	90
5.	Discussion.....	93
5.1	Assessment on overall solution	93
5.1.1	Adaptability of the solution	94
5.1.2	Business aspect of the system	94
5.1.3	Comparison with existing solutions.....	95
5.2	Project Management.....	95
5.2.1	Development process	95
5.2.2	Project planning	95
5.3	Project Outcomes	96
5.3.1	Warana- Recruitment helper for enterprises	96
5.3.2	Research papers	97
6.	Conclusion.....	99
7.	Future Work.....	100
7.1	Integrating new online profiles for aggregate profile generation.....	100
7.2	Integrating a Skill Inventory module to the current system	100
7.3	Generic Model for All the Enterprises	100
8.	References	101

Table of Figures

Figure 2. 1 Fragment of is-a relationship.....	16
Figure 2. 2 Sample Dependency Tree.....	19
Figure 2. 3 The prototype hub–authority graph G_{HA}	26
Figure 2. 4 Two directed graphs	26
Figure 3.1 System Use Cases.....	33
Figure 3.2 Recruitment Helper Module Architecture	35
Figure 3.3 System MVC Architecture	37
Figure 4.1 Information Groupings	40
Figure 4.2 Types of Information in Groupings.....	41
Figure 4.3 Layered Architecture (Info Extract from Resume)	42
Figure 4. 4 Info Extractor Module Pipe Line.....	50
Figure 4. 5Overall Comparison Heading Identification	54
Figure 4. 6 Overall Comparison Information Extraction.....	57
Figure 4. 7 Information Sources	58
Figure 4. 8 Score Generation	61
Figure 4. 9 Web Article Extraction.....	61
Figure 4. 10 Algorithmic Performance	64
Figure 4. 11 Key Term Performance I.....	65
Figure 4. 12 Key Term Performance II.....	65
Figure 4. 13 Average Precision.....	66
Figure 4. 14 Weighted Aggregation Precision.....	67
Figure 4. 15 Recall variation with depth.....	68
Figure 4. 16 Number of Top n Results(II)	68
Figure 4. 17 Precision before and After Abbreviation Integration	69
Figure 4. 18 SigmaC App	71

Figure 4. 19 SigmaC Application Architecture	73
Figure 4. 20 Technology Graph.....	78
Figure 4. 21 Graph Similarity Measuring Activity Diagram.....	79
Figure 4. 22 Technical Proficiencies	90
Figure 4.23 Advanced Search	91
Figure 4.24 Tabular Data Representation	91
Figure 4.25 Candidate Comparison	92
Figure 4.26 Spider Web Graph	93

Table of Equations

Equation 2. 1 calculate precision	15
Equation 2. 2 calculate recall	15
Equation 2. 3 F measure.....	15
Equation 2.4 Wu & Palmer Measure function.....	16
Equation 2.5 Leacock & Chodorow's Measure function	17
Equation 2. 6 Resnik's Measure function	17
Equation 2. 7 Lin's Measure function.....	17
Equation 2.8 C – value.....	24
Equation 2. 9 Similarity matrix update method	26
Equation 2. 10 similarity matrix update method of neighbor matching method	27
Equation 2. 11 calculating in-degree similarity	28
Equation 2. 12 calculating out-degree similarity	28
Equation 2. 13 initializing in-node similarity matrix.....	28
Equation 2. 14 initializing out-node similarity matrix.....	28
Equation 2. 15 calculate the similarity score of the two graphs	29
Equation 4.1 Learning Weights	67
Equation 4.2 Update Similarity Matrix Entry	80
Equation 4.3 Calculating In node Similarity matrix	80
Equation 4.4 Max in degree	80
Equation 4.5 Minimum in degree	81
Equation 4.6 Calculating out node similarity matrix	81
Equation 4.7 max out degree	81
Equation 4.8 min out degree	81
Equation 4.9 Initializing in node matrix	81
Equation 4.10 Initializing out node matrix	81

Equation 4.1 Final Similarity Score.....	83
--	----

Table of Tables

Table 2.1 Contingency Table.....	14
Table 2. 2 Similarity matrix of two graphs of Figure 2.4	27
Table 2.3 Similarity matrix of two graphs in Figure 1 using neighbor matching method.....	29
Table 4. 1 Heading Identification Statistics.....	51
Table 4. 2 Educational Heading Contingency Table	52
Table 4. 3 Educational Heading Contingency Table	52
Table 4. 4 Interests Heading Contingency Table.....	52
Table 4. 5 Achievements Heading Contingency Table	53
Table 4. 6 Personal Heading Contingency Table.....	53
Table 4. 7 Referee Heading Contingency Table.....	53
Table 4. 8 Work Information Heading Contingency Table	54
Table 4. 9 Educational Information Extraction Contingency Table	55
Table 4. 10 Project Information Extraction Contingency Table.....	55
Table 4. 11 Achievements Information Contingency Table.....	56
Table 4. 12 Personal Information Contingency Table	56
Table 4. 13 Referee Information Contingency Table	56
Table 4. 14 Work Information Heading Contingency Table	56
Table 4. 15 Graph Similarity Result Table	88

1. Introduction

1.1 Overview

Today almost every Enterprise receives hundreds of resumes from the candidates for various employees. In the recruitment process, enterprises have to analyze the received resumes and to select best candidates for the interviewing. Performing this analysis is almost all the time happens manually, which is a tedious and a time consuming task.

When we consider the recruitment process of an organization, what they really need to have the *big picture* about the candidate they are. According to some discussions with interviewers in the IT industry, they emphasized that they not only look at the resume of a certain candidate. Without only looking at the resume, they need to get a clear idea about the other skills of the candidate. For this reason, they go through the online profiles of the candidate for seeking more information. At the evaluation process, they not only consider how the candidate face the interview and the content of the resume, they give an equal value for the information they found from such online profiles. One reason that the interviewers expressed is that the candidates sometimes do not include some other information in the resumes, they tends to include information regarding the position they are applying. If we have a system, which can provide such a *big picture* about the candidate, it can help and improve the recruitment procedure.

At present people, maintain their online profiles such as LinkedIn, Stackoverflow, GitHub, Blogs, etc. These sources are rich of information about the candidate's qualifications and experiences except the resume of the candidate. What really happens nowadays is the recruiters has to go through this information one by one and then get an idea about the candidate's experiences. Finally, the recruiter has to have a summarized view at the candidate and this is not always an easy task to perform. By automating this analysis considering the enterprises' profile reduce the required time and provide expressive analytics for a given candidate. Such a system can be rather beneficial than the manual analytics.

1.2 Warana Recruitment Helper for Enterprises

Warana Recruitment Helper for Enterprises provides the best solution for the tedious task of analyzing candidate information and help the decision making process for the recruitment administrators. In Warana Recruitment Helper, it extract the information about a particular candidate not only from the resume but also from the candidate's online profiles. All these information are aggregated together and an aggregate profile for a particular candidate is generated. This profile is being compared with the knowledgebase of the company and then the system provides analytical score for the candidate depicting to what degree the candidate suits to the company. In addition, another advantage of this system is its capability of integrating with a skill inventory of the organization. Without having a separate skill inventory for the organization this system can provide the required details in a perspective, which enables to evaluate the skills of the candidates in an analytical manner. As a future expansion of this system, we are hoping to integrate a skill inventory module for the main system. At the moment as a first step we are only paying the attention on the recruitment helper module.

2. Literature Review

2.1 Introduction

This chapter provides detailed description of the literature review done for Warana project. Furthermore, it contains the knowledge and techniques to achieve the four main objectives of our project.

First, we divided our project into four main tasks. Those are,

- Information Extraction from Resumes
- Sentence Similarity between Sentences
- Skill Recognition
- Measuring graph similarity

As our project need to process resumes of candidates, we have to read those resumes and extract information that are relevant to us. Therefore, we had to do a research on this subject. To extract information from resumes, we have to measure similarity of the sentences. Therefore, we carried out a research on measuring sentence similarity between sentences.

One major part of our project is to create aggregated profile for each candidate. For that, we need to extract information from the online profiles and other online resources of the candidates. Therefore, we had to do a research on skill recognition of the candidate from the online resources. Warana's outcome is a sorted list of candidates according to their matching score to the enterprise. To calculate that score, we have to measure similarity of the candidates' profiles with the company concept map. As we are generating graphs for those, we had to do a research on measuring graph similarity of two graphs.

All of the knowledge we acquire from those researches will described in detail in this chapter.

2.2 Information Extraction from Resumes

2.2.1 Information extraction from data and its importance

When given a document it is not enough to go through the lines of the text. In order to give a value, we need to extract the information from the text data. This task is still a branch of developing in the area of Natural Language Processing (NLP). Why this task one of the most challenging task is, there does not exist a precise, well-defined way or methodology for extracting the information from plain text. Therefore, depending on the situation/ context, these methodologies have to be changed.

It is not always possible to go through the whole set of data to identify what that data stands for, due to the unstructured or semi structured organization of the data. Therefore, in order to make it easier for analyzing a structured representation of data provided during the information extraction procedure [7].

2.2.2 Resumes and Structure

The structure of a resume is the first and foremost challenge in this task. This is because there is not a particular standard or fixed template for writing resumes. The structure and the organization of the data in a resume heavily depend on the person, career, etc... Therefore, the resumes always stands as semi structured or unstructured data set.

There is one common feature in any resume apart of its heterogeneity. Each resume is organized in a hierarchical pattern [8]. At the top layer of this hierarchy contains the general information such as *Personal information, Education Information, etc...* Under these levels (Usually these are the headings of the resume.) resides the detailed information belonging to each category. As an example, a person usually includes his/her Name, Date of Birth, and Marital Status under the Personal Information title. This feature is common to any of the resume. This feature identification is the beginning of any Resume Information extraction system.

2.2.3 Choosing an information extraction Model

This is the most important stage of the process. Depending on the context, the model has to be decided. Before this model is determined, we need to identify the type of information to be

extracted [9]. After determining the type of information to be extracted, then the model is designed to be compatible with the goal. Most popular information extraction models are classification models, Support Vector Machine (SVM) models and Hidden Markov Models (HMM). Rather than these two models, there are variations too. Such variations use hybrid models [8]. These models are being tuned on a set of sample data set in order to acquire the expected outcome and the evaluation methods can be used to determine the quality of the models.

2.2.4 Process and Tools

2.2.4.1 Pattern/Template Matching

During the process of information extraction from resumes, Pattern Matching plays a major role. In most of the time people follows certain common writing patterns. As an example people, include their *Name, marital status, email, etc...* Under the Personal Information while including *Attended Schools and Institutions with their duration of attendance* under Educational Information. Identifying such patterns allows simplifying the model and following more human like analytical approach.

Another such important technique is template matching. However, there is a limitation on such template matching techniques. That is these heavily depend on the context/ domain. Template matching hardly used in the domain independent text parsing. One such situation where we can use template matching is, when we can identify a common template for a given set of resumes. These type of templates can be determined easily when the data is organized in a structured manner. In such situations, the parsing becomes easier and provides a higher accuracy. However, in situations where the heterogeneity of the data set is significant this technique can be discarded.

2.2.4.2 Name Entity Recognition (NER)

Name Entity Recognition plays a major role in information extraction processes. All the times we have the need of identifying the names of the persons, organizations, dates, etc. NER is the solution to the above problem. There are NER libraries such as StanfrdCoreNLP[5] which provide this facility. In the resume information extraction process we need identify the

Company names (eg: Companies a person worked before), Dates, Name and titles of persons (eg: Name of the non-related referees).

In StanfordCoreNLP text parser, there are three types of NER classifiers. Those are as follows.

3 class: Location, Person, Organization

4 class: Location, Person, Organization, Misc

7 class: Time, Location, Organization, Person, Money, Percent, Date

2.2.4.3 POS tagging and Lexical Features Extraction

POS tagging (Part of Speech Tagging) and Lexical Feature Extraction provides further fine grain parsing for the information extraction procedure. By using this methodology, we can identify the noun groups, verb groups and the relation between the phrases in the document. This method provides a greater flexibility on handling the unstructured data. When handling the unstructured data it is essential to identify the relation between sentences or phrases which lies together (Proximity analysis).

2.2.5 Evaluation Methods

Since Natural Language Processing does not have a precise methodology to have an exact solution to a given problem and the solution is based on, different heuristics there should be proper evaluation criteria to verify that the results are acceptable. One of the most widely used methods are *precision and recall*. Since Precision and Recall alone does not provide much satisfactory and comparable analysis, both are used in combination. Both the precision and recall can be represented in terms of the entries in a table as follow Table 3.1.

Table 2.1 Contingency Table

	Relevant	Non Relevant
Retrieved	TP (True Positive)	FP (False Positive)
Not Retrieved	FN (False Negative)	TN (True Negative)

$$Precision = \frac{TP}{TP + FP}$$

Equation 2. 1 calculate precision

$$Recall = \frac{TP}{TP + FN}$$

Equation 2. 2 calculate recall

Since analyzing the results based on the above two metrics sometimes arises conflicts. In order to provide more conflict free analyzing Combined F measure can be used either. Since the Combined F measure is calculated based on both precision and recall, it provides the same statistical results in terms of one value. The Combined F measure is calculated as follows.

$$F = \frac{2PR}{P + R}$$

Equation 2. 3 F measure

Finally choosing among the available evaluation metrics also depends on the context as well as the application domain. Therefore having certain domain specific knowledge can decide proper evaluation criteria.

2.3 Sentence Similarity between Sentences

2.3.1 Similarity between Words

Each sentence can be seen as a set of words, which are ordered in a manner based on a specific grammatical structure of a language. Therefore, the foundation of the semantic similarity between two sentences is the similarity of two words. There are several methods for measuring the semantic similarity between words. These methods can be categorized in to four as *Distance based measures*, *Information Content based Measures*, *Feature based Measures* and *Hybrid Measures* [2], [3]. All these measures are based on either a semantic network or a lexical database. One such popular lexical database is the WordNet Electronic Lexical Database [1]. In WordNet, words are considered as concepts and they are organized in to a tree structure based on the relatedness of the concepts. Due to this concept organization, WordNet is used as a base for similarity measures.

2.3.1.1 WordNet Structure

In WordNet the concepts are ordered in a hierarchical manner based on several relationships between the concepts. Such relationships, which can be found in the semantic network of WordNet, are *is-a*, *part-of*, *member-of*, *substance-of*. Depending on the network, the distance metrics are being calculated in order to use in the Distance based similarity measures.

Bellow Figure 2.1 shows a fragment of such is-a relation between concepts [2].

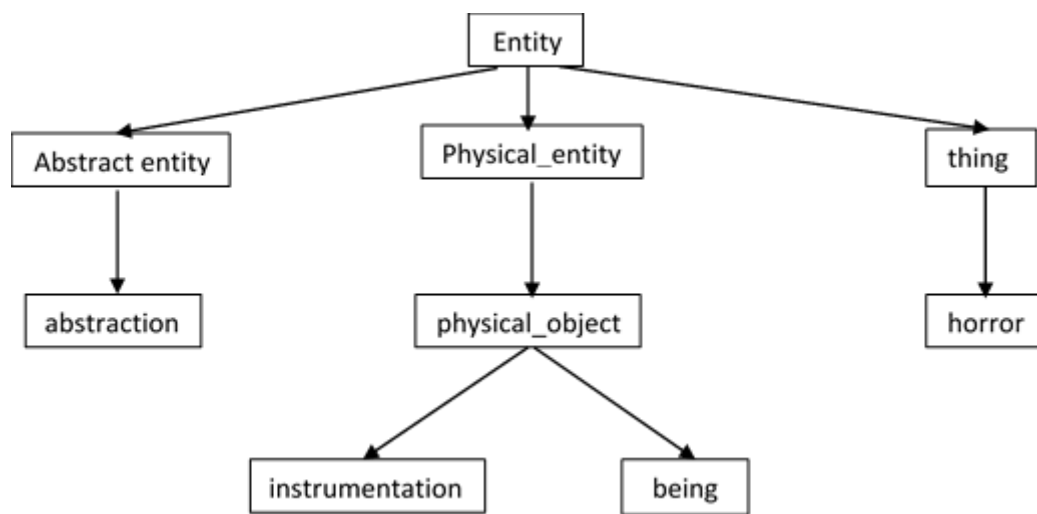


Figure 2. 1 Fragment of is-a relationship

2.3.1.2 Distance Based Measures

The main idea of the distance-based measures is to define the similarity as function of the distance between two concepts/words according to the position in the Semantic network. [2], [4].

2.3.1.2.1 Wu & Palmer Measure

In the Wu & Palmer Measure, the similarity is a function of distances between the two concepts.

$$\text{Sim} = \frac{2 * \text{depth}(\text{lso}(C_1, C_2))}{\text{len}(C_1, C_2) + 2 * \text{depth}(\text{lso}(C_1, C_2))}$$

Equation 2.4 Wu & Palmer Measure function

In the above equation,

$lso(C_1, C_2)$ - Depth of the least common subsume of the two concepts from the root.

$len(C_1, C_2)$ - Distance between the two concepts.

2.3.1.2.2 Leakcock & Chodorow's Measure

Leakcock's measure take in to account the distance between the two concepts as well as the maximum depth of the hierarchy.

$$Sim = -\log\left(\frac{len(C_1, C_2)}{2 * deep_max}\right)$$

Equation 2.5 Leakcock & Chodorow's Measure function

$deep_max$ - Maximum depth of the hierarchy.

$len(C_1, C_2)$ - Distance between the two concepts.

2.3.1.3 Information Content Based Measures

In information content based measures main idea is that each concept in the semantic network hold itself a certain amount of information. In the information, content based measures this amount of information is take into account. [2], [4].

2.3.1.3.1 Resnik's Measure

Resnik's Measure take into consideration the least common subsume and the information content holds with it in the semantic network/ lexical database [2], [4].

$$Sim = IC(lso(C_1, C_2))$$

Equation 2. 6 Resnik's Measure function

2.3.1.3.2 Lin's Measure

Lin's Measure takes the same approach as the Resnik's Measure with one change. In the Lin's Measure it consider not only the Information content of the least common subsume but also the Information content of both the concepts taken into account.

$$Sim = \frac{2 * IC(lso(C_1, C_2))}{IC(C_1) + IC(C_2)}$$

Equation 2. 7 Lin's Measure function

2.3.1.4 Feature Based Measures

Feature based methods takes a different approach on the similarity measuring procedure. It does not consider the properties of the semantic taxonomy, rather than it uses the ontology properties of the concepts [2]. These properties are such as the words used to describe a particular concept. Based on these properties, we can determine how much similar characteristics shared by a pair of concepts and based on this measure we can determine the similarity between the two given concepts. One such Feature based measure is *Tversky's Measure*.

2.3.1.5 Hybrid Measures

In hybrid measures, as its name implies is a combination of methods described above. When choosing a hybrid measure it is worthwhile to consider the tradeoffs between several methods depending on the context of use.

2.3.2 Semantic Similarity of Sentences

2.3.2.1 Dependency Grammar and POS Tagging

In dependency grammar, which was proposed by Lucien Tesnière [10] mainly focuses about how the words are related to each other with identified grammatical relationships based on the verbs as the structural centre. In order to identify the correct set of Dependencies between the syntactical units of a sentence, it is necessary to identify the correct role of each syntactic unit in a given sentence. This is because, for a given set of verbs, nouns, adverbs, adjectives, etc. (Syntactical units) the role they play in a sentence can be changed according to the way those units are ordered. Therefore, it is necessary to identify the dependencies between the syntactical units, before following Natural Language Processing tasks.

Identifying the dependency structure and the POS tags of sentences, StanfordCoreNLP text parser [5] provides a better solution. In Stanford Dependency Structure, there are approximately 50 grammatical relationships. In the implementation process, we have used Penn Tree Bank part-of-speech tags and phrasal labels. Bellow sentence describe how Stanford Parser represent Dependency relationships.

“Bell, based in Los Angeles, makes and distributes electronic, computer and building products”.

- nsubj(makes-8, Bell-1)
- nsubj(distributes-10, Bell-1)
- vmod(Bell-1, based-3)
- nn(Angeles-6, Los-5)
- prep in(based-3, Angeles-6)
- root(ROOT-0, makes-8)
- conj and(makes-8, distributes-10)
- amod(products-16, electronic-11)
- conj and(electronic-11, computer-13)
- amod(products-16, computer-13)
- conj and(electronic-11, building-15)
- amod(products-16, building-15)
- dobj(makes-8, products-16)
- dobj(distributes-10, products-16)

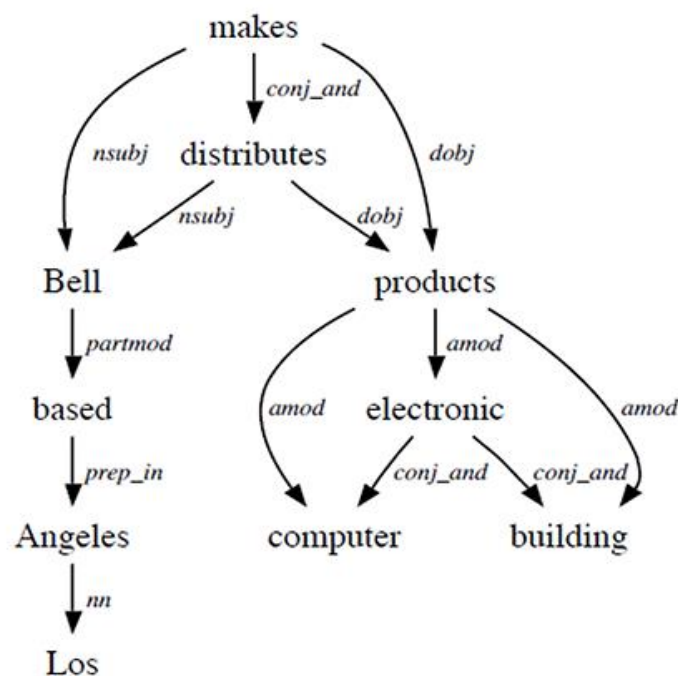


Figure 2. 2 Sample Dependency Tree

2.3.2.2 Semantic Similarity and Dependency Grammar

Semantic similarity measuring between two words/concepts is a process of quantifying the relatedness between two concepts based on the semantic properties of them. This same idea has been extended to measure the semantic similarity between given two sentences. Measuring the semantic similarity of the sentences is based on identifying the semantic properties of the sentences and comparing those semantic properties of the two sentences. One such major technique of identifying the semantic properties is Part of Speech Tagging (POS Tagging).

In semantic similarity, measuring the major focus is on the meaning of the sentences. The reason for such approach is there can be two sentences, which gives the exact same meaning but having different set of words in each sentence. As an example,

Sentence one: Nimal can run fast

Sentence two: Nimal runs swiftly

In both above sentences gives the same meaning, but having different words. In the semantic similarity measuring these kind of situations are being considered and gives them a value. In order to achieve the expected outcome of the semantic similarity based methods, those methods have to rely on the word similarity measures. As mentioned earlier in this section, word similarity measures are the base of the sentence similarity measures. Therefore, each semantic similarity-measuring algorithm highly depends on one of the above word similarity measuring techniques depending on the context.

2.3.3 Syntactic Similarity of Sentences

Syntactic similarity measures are another type of measures, which used most of the time. In this technique only considers the positioning of the words in a sentence. This positioning is defined by considering the POS tagging and determining the sentence has the required POS tags, which can be used to declare it as a sentence. One important fact is missing in this technique. In this technique, the sentences are considered as a bag of words, which are being structured, in a particular grammatical order. When we take into account the same two example sentences in the above sub section, similarity between those two sentences with respect to the syntactic similarity measures is lesser than the semantic similarity measures. Because the syntactic similarity measuring techniques do not consider the meaning of words, just the syntactic characteristics based on the bag of words model. In particular, contexts certain

tradeoffs have to take into account when deciding the best similarity measure, because in some situations syntactic similarity techniques can behave optimally rather than the semantic similarity measures.

2.3.4 Hybrid Methods

Hybrid methods allow to overcome the weaknesses of each individual method and to improve the quality of the method by integrating the strengths of each individual method. Hybrid methods take into account, both semantic and the syntactic similarity measures combined together [5]. One such hybrid method uses a vector space model to calculate the similarity between sentences as described in [5]. According to the results, this method provides a greater flexibility against the sentence pairs which having lower in size (lesser number of words). This is one another important factor, which should be considered during the usage of a particular similarity measuring procedure.

2.4 Skill Recognition

2.4.1 Skills over technologies

In aggregated profile generation, we need to recognize and list the skills possessed by the person who represented in the profile. In profile generation, we can easily find the technologies the person familiar with just by using his GitHub repositories. Technology is one thing and skill is another thing. Therefore, we could easily list the technologies instead of skills in profile. The reason to select skills over technology is, identifying skills are more important in the interviewing process. For example, consider a scenario where a company, which built computer vision, based software-interviewing candidates. A candidate may have java listed as a technology he is familiar with in his profile. However, the problem is, even if the entire computer vision library is written in java, that person may know nothing about computer vision. Therefore, it is important to identify skills in aggregated profile generation.

2.4.2 LinkedIn Endorsements

In LinkedIn, there is a feature called endorsements, which allows users to endorse upon the suggested skills of their connections. This could have been used as a direct source to find the skills for the aggregated profile. However, the problem is, this is not trustworthy. LinkedIn

does not do any background check about these skills and they are only based on the endorsements made by connections of a particular profile. Therefore, in order to make use of these endorsements in our project, we need to confirm whether these skills are actually possessed by the respective person.

2.4.3 Confirmation of Endorsements

In order to confirm the skills, first we built a list of terms and phrases related to the skill by processing the documents corresponding to the skill. Then, we built a list of terms and phrases (userlist) by processing the by processing the documents such as CV data, blog posts, project descriptions, publication descriptions of the interested person. Then a measure of presence of the key phrases for a particular skill in the userlist could be used as a way to confirm whether the person actually possess the skill.

2.4.4 Automatic Term Recognition (ATR)

ATR is a very important aspect, which deals with the extraction of technical terms in domain specific language corpora. The importance of this to our project is, we can use these techniques to identify technical terms related to Skills as described above.

Currently, there are several methods to extract terms given a set of documents with their own pros and cons. Therefore, most of practical implementations have used a combination of these methods [11]. In addition, most of these works are based on the simplifying assumption that “the majority of terms consist of multi-word units”.

(Caraballo & Charniak 1999; Deane 2005; Fahmi, Bouma, & van der Plas 2007; Wermter & Hahn 2005). However, there are evidences which supports and against the validity of this assumption.

85% of domain-specific terms are multi-word units (Nakagawa & Mori 1998)

Only a small percentage of gene names are multi-word units (Krauthammer & Nenadic 2004)

2.4.4.1 Zipf's law

This is an important law in keyword and phrase extraction of natural languages. The law states that,

“Given a corpus of Natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table” [12].

This means, the 1st most frequent word will be found in the corpus as twice as the 2nd frequent term and three times as often as the 3rd frequent word.

2.4.5 TermEx term recognition algorithm - Sclano et. al 2007

This method uses best-reported terminology extraction techniques. Then filter out non-terminological strings from the list of syntactically plausible candidates. The following list summarizes the filters used in this technique.

- **Domain Pertinence:** This measures the domain relevance of a term with respect to a particular domain
- **Domain Consensus:** This is an entropy-based measure. The consensus is high if a term has an even probability distribution across the documents chosen to represent the domain.
- **Lexical Cohesion:** The degree of cohesion among the words that compose a terminological string.
- **Structural Relevance:** This measures the importance of a term. For example, if the term is found in a title, that term has a high importance even if it has a low frequency in the document. Therefore, in order to compensate this, the measure of its frequency is increased by a factor of k defined by the user.
- **Miscellaneous:** This is used for things like detect misspellings, acronym...etc. GlossEx term recognition algorithm. See Kozakov, et. al 2004

2.4.6 IBM gloss ex

This is a method developed by IBM in order to extract glossary items for a particular technical domain from a large collection of unstructured data. This process has 4 major stages as below.

1. Glossary-item identification
2. Glossary-item acquisition
3. Term filtering
4. Glossary-item aggregation

First stage is done using morphological analysis, POS tagging, and noun phrase recognition. In the second stage, abbreviations and out-of-vocabulary terms are identified. This part is important especially for technical documents because often, those types of words are common in technical domain. Term filtering is used to remove unimportant glossary items generated in previous steps such as person names, arbitrary strings including special characters... etc. After completing these stages, we get a set of candidates as glossary items. Among them, there may be same terms in different forms such as singular-plural, in different tenses, misspellings, abbreviations and full forms of same thing. In aggregation, these different forms will be identified and remove redundant items.

2.4.7 CValue term recognition algorithm. See Frantzi et. al 2000

This is a domain independent multi-word ATR methodology, which combines linguistic and statistical information of the corpus.

The linguistic part contains the following three stages.

1. Part-of-speech(POS) tagging
2. The linguistic filter
3. Stop list

POS tagging tags the terms in the corpus as noun, verb, adjective... etc. Then the linguistic filter removes not-allowed patterns (eg: [Adj|Noun] + Noun) and gives a list of candidate phrases. The stop list contains a list of words that are not expected to present in terms. If there is any candidate terms with a stop word, that term will be removed from the candidate list.

Next stage is statistical part.

$$C - value(a) = \log_2 |a|. f(a) \quad : \quad a \text{ is not nested}$$

$$= \log_2 |a|. f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b) : \quad otherwise$$

Equation 2.8 C – value

Where,

- a : candidate string.
- $f(x)$: x' frequency of occurrence in the corpus.
- T_a : the set of extracted candidate terms that contain a.

$P(T_a)$: is the number of these candidate terms.

Then, the candidate strings are ordered by the determined C-value. [13]

Other than above explained methods, there are several other techniques used for term extraction with their own pros and cons.

- The RIDF algorithm, (Church, K. and Gale, W. 1995a)
- Word weirdness algorithm applied to term recognition algorithm. (Ahmad et al 1999)
- Average corpus tf
- TF-IDF algorithm

2.5 Measuring graph similarity

Many of data analysis systems in today requires a richer problem representation and corresponding data analysis techniques while keeping the structural information and original data. Therefore, many practical systems use graphs to represent the structural information in the data. Many applications of today need to have a quantitative measure of the “similarity” of two graphs. Many researches going on this problem and some researches such as graph isomorphism problem, edit distance and maximum common sub graph help to build a solution for that.

We say that, an element in graph G_A and an element in graph G_B are considered similar if their respective neighborhoods within G_A and G_B are similar [14]. Almost all researches based on this theory and they are trying to measure the similarity based on that.

2.5.1 Graph similarity scoring and matching

One influential iterative approach was the HITS search algorithm introduced by Kleinberg in [15]. He suggested that useful Web search results could be divided into two different categories: ‘hubs’, which point to many good sources of information on the query; and ‘authorities’, which are pointed to by high-quality hubs. Kleinberg proposed an iterative algorithm for computing a hub and authority score for each node in the Web graph G_W ; the authority score of a node n at iteration k is simply the sum of the hub scores of the nodes that point to node n at iteration $k - 1$, and the hub score of node n at iteration k is the sum of the

authority scores of the nodes to which node n points. The even and odd iterates of this procedure will each converge, but possibly to different limits which may depend on the initial values chosen.



Figure 2.3 The prototype hub-authority graph G_{HA} .

In [16], Blondel et al. view Kleinberg's algorithm as a comparison between each node in the graph G_W and in the light of this observation, Blondel et al. generalize Kleinberg's update equation to construct a similarity measure between any two nodes in any two graphs. Blondel et al. denote the similarity of node i in G_B and node j in G_A at stage k by $X_{ij}(k)$ and define it via the following iteration [16].

$$\tilde{x}_{ij}(k) = \sum_{r:(r,i) \in E_B, s:(s,j) \in E_A} x_{rs}(k-1) + \sum_{r:(i,r) \in E_B, s:(j,s) \in E_A} x_{rs}(k-1).$$

Equation 2.9 Similarity matrix update method

Using this equation, we update a similarity matrix of two graphs iteratively. Then we normalize the similarity matrix X by $X \leftarrow X / \|X\|_2$. When we calculating the similarity score, we can see that the values are, tend to converge in each iteration. Therefore, we can stop this update function after the scores are converge to a specific value that we are given.

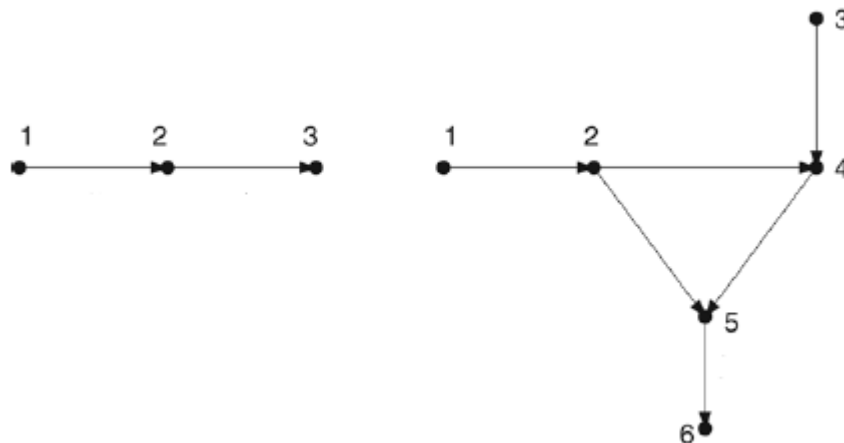


Figure 2.4 Two directed graphs

Table 2. 2 Similarity matrix of two graphs of Figure 2.4

Nodes	1	2	3
1	0.124	0	0
2	0.348	0.445	0
3	0.157	0.054	0
4	0.094	0.563	0.193
5	0	0.338	0.340
6	0	0	0.094

Table 2.2 shows the similarity matrix of the two graphs shown in Figure 2.4. It represent the similarity of each node of graph G_A to the each node of graph G_B .

2.5.2 Measuring Similarity of Graph Nodes by Neighbor Matching

This research is based on the research that mentioned in sub section 5.1 above. In here [17], they improve the update method of the similarity matrix to get better similarity score of the nodes. Actually, in the Equation 1.10, they use only the in-nodes of two nodes that are measuring the similarity. However, in here they consider both in-nodes and out-nodes of two nodes that are measuring the similarity to update the similarity matrix.

$$x_{ij}^{k+1} \leftarrow \frac{s_{in}^{k+1}(i, j) + s_{out}^{k+1}(i, j)}{2}.$$

Equation 2. 10 similarity matrix update method of neighbor matching method

Equation 2.10 will calculate the similarity of i^{th} node of graph G_A and j^{th} node of graph G_B in $(k + 1)^{th}$ iteration. As we see, we need to calculate $S(i, j)_{in}$ and $S(i, j)_{out}$ in $(k + 1)^{th}$ iteration first. These $S(i, j)_{in}$ is the in-degree similarity of node i in G_A and j in G_B . $S(i, j)_{out}$ is the out degree similarity of node i in G_A and j in G_B . They can be calculate using Equations 2.11 and 2.12.

$$s_{in}^{k+1}(i, j) \leftarrow \frac{1}{m_{in}} \sum_{l=1}^{n_{in}} x_{f_{ij}^{in}(l)}^k g_{ij}^{in}(l)$$

$$m_{in} = \max(id(i), id(j))$$

$$n_{in} = \min(id(i), id(j))$$

Equation 2. 11 calculating in-degree similarity

$$s_{out}^{k+1}(i, j) \leftarrow \frac{1}{m_{out}} \sum_{l=1}^{n_{out}} x_{f_{ij}^{out}(l)}^k g_{ij}^{out}(l)$$

$$m_{out} = \max(od(i), od(j))$$

$$n_{out} = \min(od(i), od(j))$$

Equation 2. 12 calculating out-degree similarity

In here $id()$ means in-degree and $od()$ means out-degree of the node. We calculate the similarity of in-degree and out-degree of nodes by taking the summation of the neighbors' similarity in previous iteration. We choose those similarity values according to the enumeration functions. Enumeration functions is a functions that gives the maximum similarity value for the each node in the given node list. We initialize the in-node similarity matrix and out-node similarity matrix as Equation 2.13 and 2.14.

$$s_{in}^1 = \frac{\min(id(i), id(j))}{\max(id(i), id(j))}$$

Equation 2. 13 initializing in-node similarity matrix

$$s_{out}^1 = \frac{\min(od(i), od(j))}{\max(od(i), od(j))}$$

Equation 2. 14 initializing out-node similarity matrix

The termination condition of the iterations is $\max_{ij} |x_{ij}^k - x_{ij}^{k-1}| < \varepsilon$ for some chosen precision ε . Finally, we can calculate the similarity score of two graphs using the final similarity matrix that we calculated. It can be calculate using the Equation 2.15.

$$s(G_A, G_B) = \frac{1}{n} \sum_{l=1}^n x_{f(l)g(l)}.$$

Equation 2. 15 calculate the similarity score of the two graphs

In the Table 5.2, shows the similarity matrix of two graphs that shown in Figure 5.2 above. We can see that, the accuracy of the similarity score of each node increased than the method described in section 5.1.

Table 2.3 Similarity matrix of two graphs in Figure 1 using neighbor matching method

Nodes	1	2	3
1	0.682	0	0
2	0.100	0.364	0
3	0.579	0.045	0
4	0.2	0.195	0.091
5	0	0.4	0.091
6	0	0	0.700

2.6 Existing Solutions

We did a research on existing solutions, which address this problem. However, existing solutions have generalized the problem to simply parsing CVs and searching in CVs. Those solutions provide facilities for HR manager to sort the CVs according to a given skill set. Therefore, the existing solutions have are not complete solutions. They have given only partial solutions for the problem

- DaXtra Parser & DaXtra Search:

This system provides facilities to parse CVs and then the users can search in CVs, sort them according to a given skillset. This is just a partial solution for the problem [18].

- RezSore:

This software is developed for candidates, not for the Enterprises. This system assign a score to a resume based on the target job defined by the user. This system uses a keyword-based approach when analyzing the Resume/CV [19].

The difference between existing solutions and the proposed system, "Warana" is it has a component, which can learn from the CVs of currently working people in the organization. Therefore, it knows what kind of people needed for a particular vacancy. Therefore, it can categorize CVs for most relevant vacancies and sort them according to the candidates' suitability.

2.7 Summary

During this research, our major focus was on three major aspects. Natural Language processing, Graph Similarity matching and the aggregated profile generation are those three major focused areas. Not only the above aspects but also we focused on the existing systems today. Today similar existing systems are not much available and the available/existing systems are only focused on the information extraction through parsing the resume.

In the Natural Language Processing, most frequently raised issue is the inconsistency and the heterogeneity of the writing formats. Therefore most of the time the natural language processing techniques have to be integrated with the domain knowledge as well, in order to get the expected outcome.

In the graph similarity matching there are several methods to get the similarity scores based on the different scoring mechanisms. These scoring mechanisms' performance and the accuracy drastically changes based on the domain. Therefore, in order to have a proper similarity measuring mechanism, it is better to compare several scoring methods and to determine between one of them based on the requirement.

Aggregated profile generation always have the ambiguity of determining the redundancy of the information as well as the truthfulness of information. It is always not possible to retrieve exactly accurate information from the available resources. In order to improve the accuracy and to remove the redundancy information retrieval mechanisms such as the tf-idf measures have to be integrated together.

3. Design

In this section, we are going to provide a detailed understanding about the design and the architecture of the system.

3.1 Rationale

As mentioned previously, this system is being integrated with the existing concept map generator called SIGMAC. In the Warana Recruitment helper, this concept map generator is being used to generate the concept map for the company knowledge base. Bellow main design decisions are the mainly identified design decisions of the system.

1. J2EE and Java Programming Language is used for the backend implementations
2. System is being developed as a web application
3. Maven Build tool is being used throughout the project in order to enable the easy system building and deploy
4. Main web system architecture is MVC architecture
5. Java Spring MVC framework, JPA/Hibernate and JBoss Application server 7 is used
6. Stanford CoreNLP and WordNet is being used as the supportive natural language Processing tools

3.2 Use cases of the system

It is always an important task to identify clearly, the relevant use cases of any system. In this section, we are going to have a look at the use cases of the system. Let us refer bellow diagram.

In Warana Recruitment Helper, there is only one identifiable user, which is the HR Manager of an organization or the recruiter. Before the user can use the system, he/she needs to create an account in the system. After creating a valid account, the user can proceed for the provided tasks. In our system, the user has been provided with several major tasks as described following.

1. Upload Resumes
User can upload one or more resumes at a time

2. Process Resume

After uploading the resumes, use can choose which resumes to be processed and then the user can command the system to process selected resumes

3. Analytic Generation

After processing the resumes, all the required and essential information is extracted from the resume as well as from the online profiles of the candidate. If the user need to generate the analytical scores for specific candidates, then user can select the candidates whose resumes have been processed and generated the candidate profiles. Then the user can command the system to generate the analytical cores for candidates. At this stage the similarity matching with the organizations concept map is done and a score value is generated for the candidate

4. View Analytics

All the analytical data has been Stored in the system at this stage. Then the user can check each and every candidate whose analytical details generated and examine the results in graph formats.

5. Search Candidates

There are some situations in, which the user may want to search for candidates who are having specific technical proficiencies and then, compare candidates with each other. This functionality has been provided at the system and the detailed comparison is being shown in graphically

6. View candidate's aggregated profile

If the user need to check the candidate details other than the analytical scores, user can check for the candidate profile and examine relevant details

According to the Figure 3.1 and the descriptions, we designed the system in which all the required requirements were fulfilled. After doing some research in the IT industrial domain we could find the requirements of the system and then all the use cases were designed accordingly. Our overall system architecture was built taking these requirements and the use cases in to consideration. In the next section, we are going to take a look at the overall system architecture in depth.

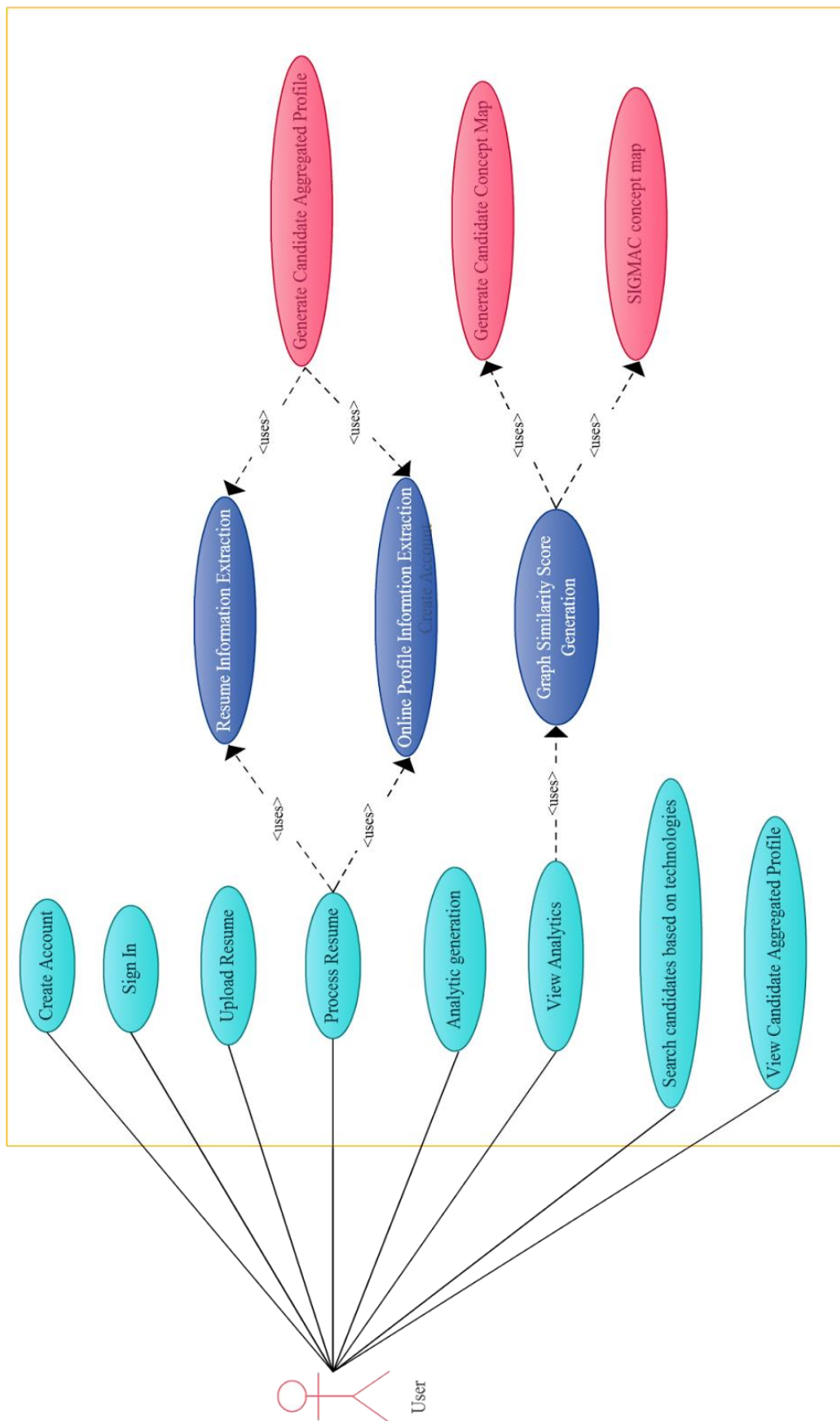


Figure 3.1 System Use Cases

3.3 Overall System Architecture

As mentioned in the above section this system architecture was designed in order to have the capabilities to fulfill the user requirements and the corresponding use cases. Bellow diagram shows the modules in Warana Recruitment helper and the interaction of the modules. This is a simplified version of the system's overall Architecture and later in this section describes the actual detailed architecture of the system.

In the bellow system architecture, there are three major sub components as shown. Those major components are *Information Extractor*, *Profile Generator* and *Analytical Engine*. Let's take a look at each component briefly.

3.3.1 Information Extractor

Information Extractor is one of the major component of this system. This component is responsible for extracting information from the online profiles as well as the candidates' resumes. In order to accomplish the required outcome of this sub system this component is being implemented as a separate module of the system. Even though this component is implemented as a separate module at the implementation process of the module it is being developed in a way, which can be easily, integrated with the main system. This information extraction component is being further divided in to two sub components. One component/module is for extracting information from candidates' resumes and the other component/module is for extracting information from the online resources. These information extraction modules and their implementation perspectives are described in depth in a later section of this document.

In the resume information extraction module on behalf of extracting the information, Natural Language Processing Techniques were followed. This module's implementation is based on identification of the information grouping structure of the resumes and depending on the type of information grouping, information is extracted accordingly.

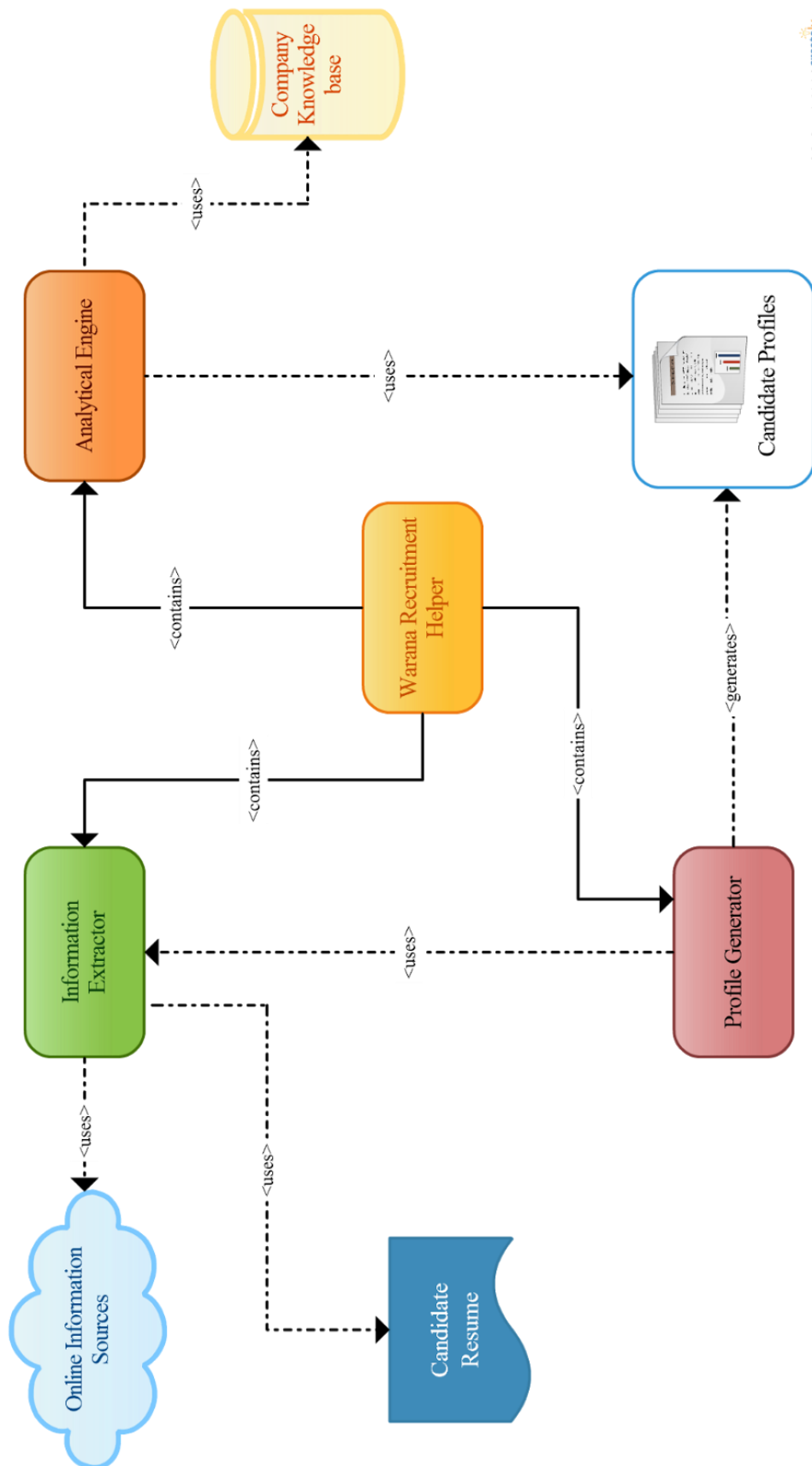


Figure 3.2 Recruitment Helper Module Architecture

In the online information extraction sub component, mainly the online profiles such as LinkedIn, blog posts of the candidate, Github project Information are taken in to account in order to extract the information required. From these profiles, mainly the information such as technological proficiencies of the candidates, recommendations, project information, publications of the candidates, etc.

3.3.2 Profile Generator

Profile generator is the component, which has the responsibility of generating aggregated profiles for the candidates. Profile generator combines information extracted from the resume as well as from the candidate's online profiles together and generates a single aggregated profile for each of the candidates. All these profile information is stored in the relational database of the system and at the analytics step these information is being used to generate concept graph for the candidate.

3.3.3 Analytical Engine

Analytical engine is the most important component in the Warana System. After creating the aggregated profiles for the candidate via the Profile generator component, analytical engine uses this information to create a concept map for the candidates. When you look at the use cases of the system use case diagram, there are two use cases for candidate's score generation as well as candidate statistics generation. For both of these two use cases analytical engine get the responsibility to generate the analytical scores. This analytical engine's implementation is being achieved having a module for score generation for the candidate. This score generation is based on a special technique called graph similarity matching. This module is being described in depth regarding the algorithmic implementation and the related research work. This analytical engine uses the company knowledge base to generate the company concept map. Then the graph similarity-matching module uses these two concept maps in order to generate the similarity score.

3.4 Application Architecture

During the system implementation, our main concern was its web based behavior. Therefore, our application architecture was mainly adhered to the Spring MVC architecture.

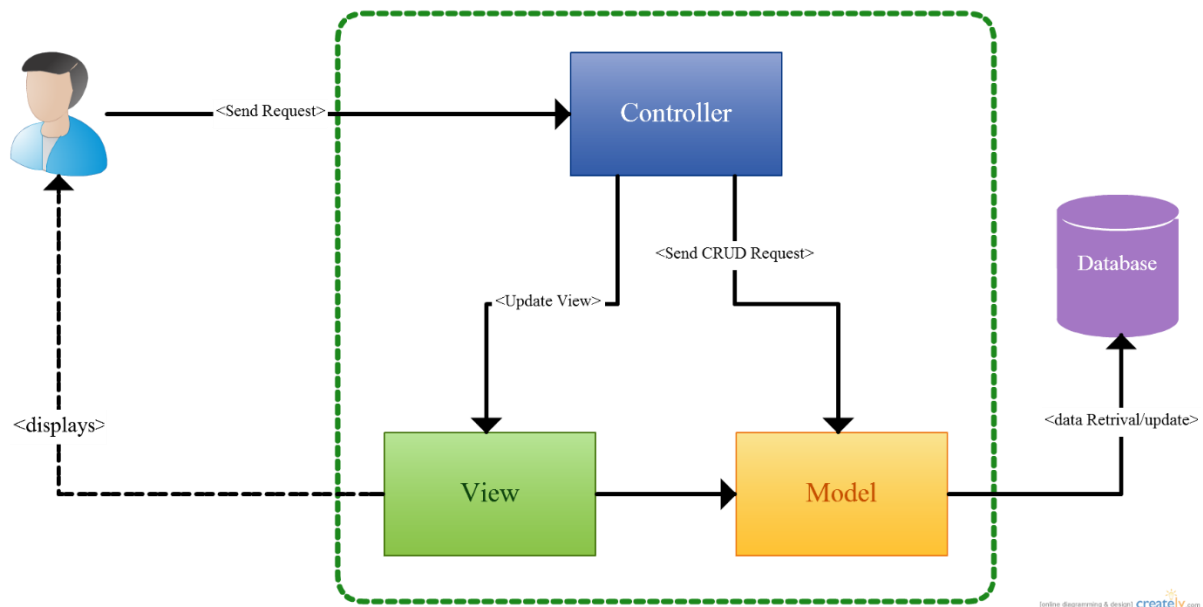


Figure 3.3 System MVC Architecture

In the application architecture as shown in Figure 3.3 Application logic and the user requests are handled at the Controller level. According to the users' request, controller perform relevant functionalities. If the user request has to be followed with a CRUD operation, at the controller level corresponding request is sent to the model. Model is the component, which handles all the database related operations, and after the completion of the function, it sends the results to the Controller component. If the controller's action has to be followed by a data representation to the user, then a request is being sent to the view components in which the relevant views are updated.

4. Implementation

4.1 Information Extraction from Resumes based on Information Grouping Property

4.1.1 Introduction

During this research component, our main goals were to extract valuable information from resumes of the candidates. In the recruitment process, when the recruiter/interviewer do not goes through the whole resume from top to the bottom and word by word. Based on the research [21] the recruiters mostly try to catch the information types such as

- Candidate Name
- Current Title and Company
- Previous Position, start and end date
- Educational Details

Except for the above details, the recruiters then focus on the technological background of the candidate, Extra-Curricular activities, etc. Therefore the recruiters always tries to have a little bit of time spent on the resume and to get hold of a considerable amount of information. Our Information extraction model is completely based on the above-mentioned methodology, in which we catch the most crucial types of information.

For the moment, let's consider two types of industrial domains. As an example, let's take in to account the domain of IT Industry and the Textile Industry. Depending on the type of the business domain, type of information the recruiters focus on widely changes. As an example, the IT Industrial recruiters focus on the technological skills of the candidates. These skills are the programming languages, Information Technological Concepts, etc. When we consider the Textile Industry, types of technologies are considered as machine operating skills, tools handling skills, etc. Due to this reason depending on the industrial domain, information extraction schema has to be changed. Depending on the domain, we can inject the domain knowledge to the information extraction model and this improves the accuracy as well as the performance of the system. As a proof of concept of this information extraction methodology,

our sample domain was chosen as the IT Industry domain and a test set of 125 resumes were used in the process.

When we consider resumes of the several candidates in two different industrial domains, we can clearly identify the difference between them. Based on the sample resume set we could identify when we consider resumes in the same industrial domain, we can identify the organization of information highly varies from one person to the other. Due to this reason when designing the model it was important to handle this heterogeneity of resume writing in order to minimize the effect and to improve the accuracy of the method. By identifying the underline common characteristics as well as the domain specific characteristics this model had been improved to gain a higher accuracy level. More about the accuracy and the performance of the system will be provided later.

4.1.2 Resumes and Identifiable Characteristics

As mentioned earlier in this section, from one person to another, writing style and the format of the resumes differs. In the domain of Natural Language Processing, we try to adhere to domain dependent processing methodologies. When we look at any resume, there is a common underlying feature. In every resume, we can find that the information is included under meaningful information groupings.

According to the Figure 4.1, we can clearly see how the candidates groups information depending on the type of information. This structure is common to any resume and our model for information extraction from the resumes is based on this structure identification.

Let's assume for the moment that we have a model which can clearly distinguish these information groupings. In our system, the goal of this module is to extract the necessary information from the text rather than section identification. Even though we could identify the information groupings, we cannot exactly say each information grouping has information been included in the same manner/format. This was the next most crucial and challenging task of the model. After analyzing a wide range of resumes (Not only the test set, because it may lead to over fitting of the model) we came in to a conclusion that we cannot exactly identify a relationship between the information in different sections. Due to this analysis of the resumes we identified information formatting in similar groupings have similar characteristics.

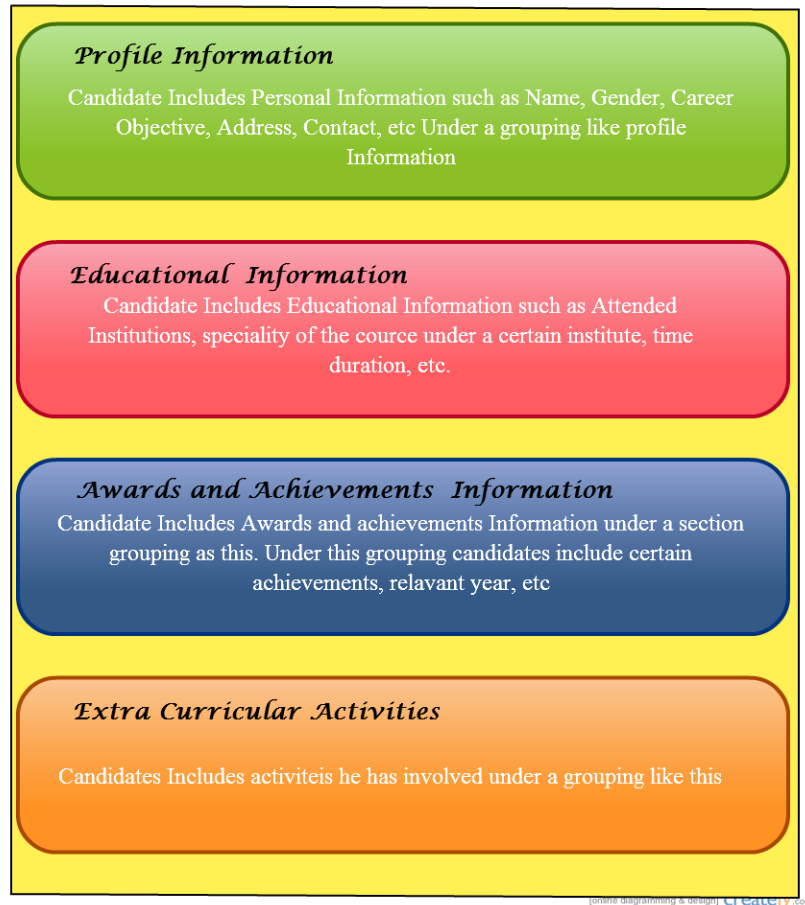


Figure 4.1 Information Groupings

As an example depending on the type of information grouping we can parse the plain text and extract information. Best way to elaborate this is when we consider the personal information and the educational information. What we focus when extracting personal information is to identify the Name, Gender, Contact Information, etc. However, when we consider the educational information we focus on extracting Institution Names, Time Durations, Course Specialties, Subject areas, etc. Both these two types of information are completely different and therefore same parsing procedure cannot be used for both situations.

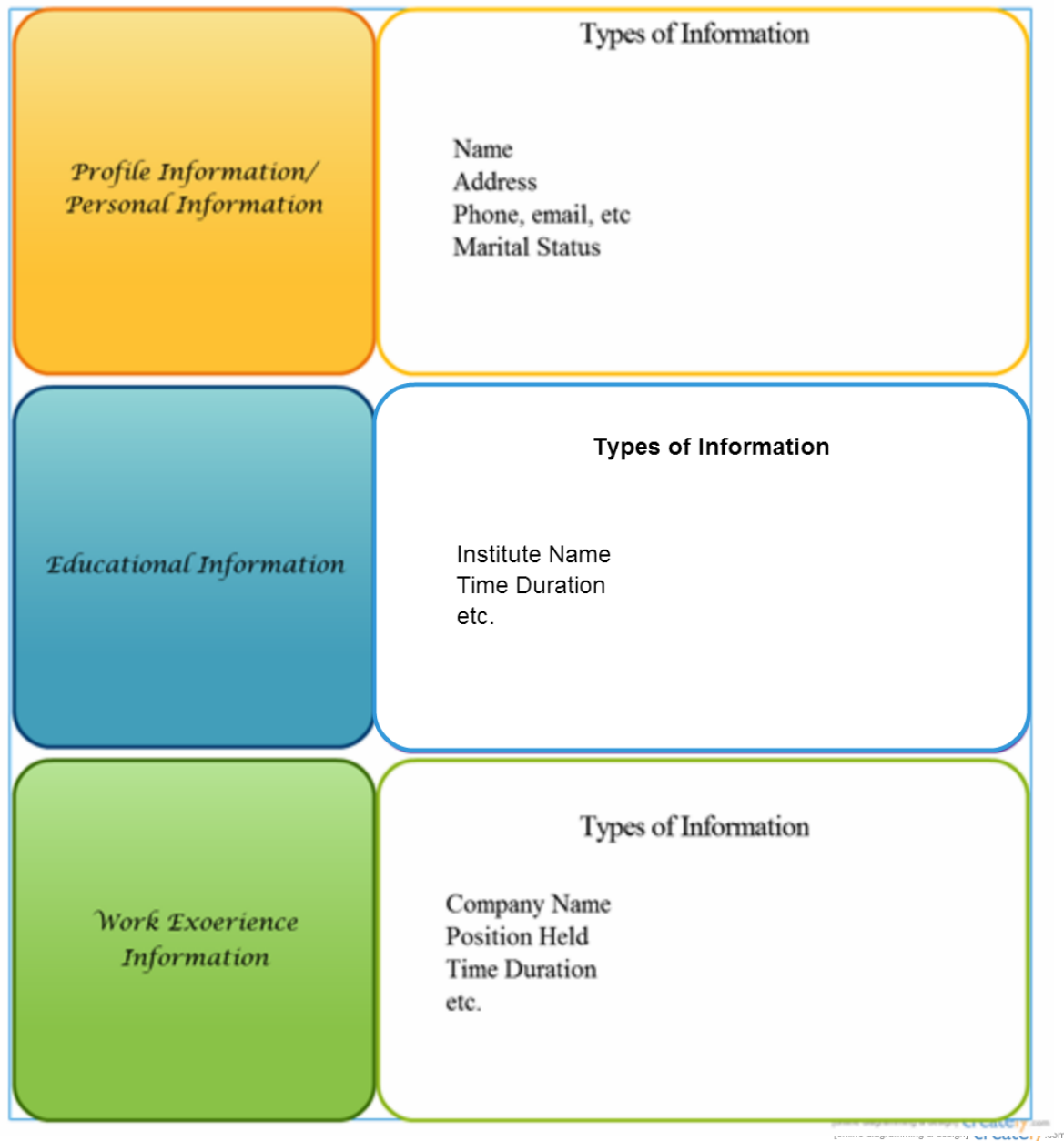


Figure 4.2 Types of Information in Groupings

In our model for information extraction, we identified this heterogeneity and the information extraction is done by parsing the plain text lines depending on the type of information going to be extracted. More on this fact will be provided in later sections.

4.1.3 Resume Information Extraction Module Architecture

As mentioned earlier in this document our system has been implemented as several separate modules and these modules are being integrated together to get the final system working. In this section, we are going to provide a detailed understanding about the Resume Information Extraction module's architecture before we are going through the implementation perspectives of the aforementioned module. First let us have a look at the below architecture Diagram of the system.

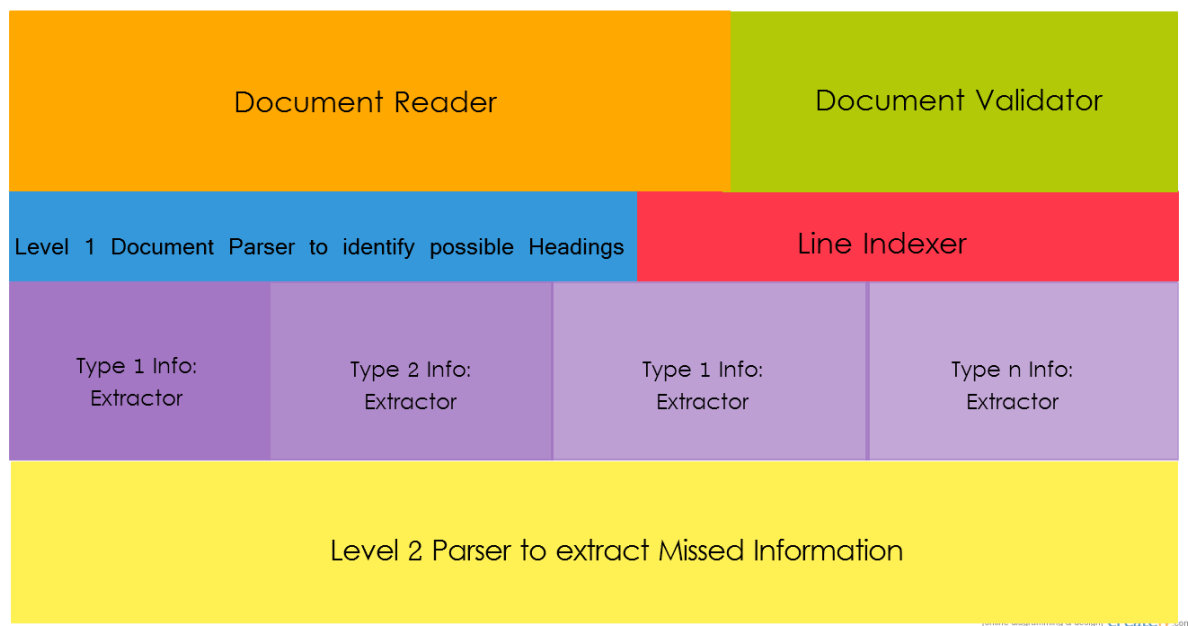


Figure 4.3 Layered Architecture (Info Extract from Resume)

This Information Extraction module has been structured according to the above layered architecture. At the first layer there are two component *Document Reader* and *Document Validator*. At this layer, the document is validated and the document is read line by line, before parsing the lines for information extraction. In addition, the document validator is the responsible component for validating the document. This component identifies if the document is in the valid format and then the *Document Reader module* is reading the document.

In the next layer, we have the *Level 1 Document Parser to identify Possible Headings*. At this level, the document has been already parsed and then these lines are traversed and processed in order to identify if a certain line can be identified as a heading or it is considered as a regular

line hopefully having some valuable information. Next Component at this level is the *Document Indexer* component. When we parse the lines one by one, we need to index these lines in order to process these lines in the next level of processing. This Indexing module indexes the lines with appropriate index values saying a certain line is a heading or a regular line having information.

In the next layer there are several modules implemented. At this level, the document has been read and parsed by the level one parser and the possible heading lines have been identified. At this level, these lines are being processed in order to extract information. These modules are being implemented in a manner, which is specific to the type of information being extracted. As an example in order to extract Educational information and Project information, we have implemented two separate modules. One of the most important fact we considered during the implementation of this layer is the extensibility of this layer. This is because, as we mentioned earlier the type of information we are going to be extracted can be different from one domain to the other. Therefore, this layer should be able to extend in future. Depending on the domain number of module in this layer can be vary.

There is another layer in the architecture diagram above. This layer is another important layer, which we included in order to avoid missing the required information during the information extraction process at the above layer. When we extract information at the above layer, we index the lines as processed lines if we could identify or extract required information from the lines. Otherwise, the lines are indexed as they are not being processed or no information is extracted from them. Then we come to the current layer and then we parse the remaining lines which were indexed as no information was extracted, again with the information extraction models to check whether these lines includes any information which we missed at the previous layer. Why we use such additional layer to extract information is because there may occur erroneous heading identification at the *Level 1 Document Parser*. Due to this reason, there is a possibility of missing valuable information during information extraction. With this additional layer, we guarantees we are extracting as much as possible amount of information from the document.

4.1.4 Tools used

This module's main focus is to extract information form the resumes. Since the resumes are read as plain text, we need to use Natural Language Processing techniques to extract

information from plain text. In addition, the document has to be read line by line since the module uses the relative positioning of the lines as a feature to extract the information. During this section, we are going to provide a detailed understanding about the tools we used in order to extract the information from the resumes.

4.1.4.1 Apache PDFBox

Apache PDFBox is an open source library, which is specially designed to deal with PDF documents. Apache PDFBox provides set of supportive tools to create PDF Documents, Manipulate Existing PDF Documents and to extract content from a PDF Document. Some of the available features of this tool is as follows [13]

- Extract Text – Extract Unicode Text from PDF
- Split and Merge – Split a single PDF Document in to many files or merge multiple files in to one
- Fill Forms – Extract data from PDF Forms or fill a PDF Form
- Preflight – Validate PDF files against the PDF/A-1 standard
- Print – Print a PDF file using the standard Java printing API
- Save as Image – Save PDFs as image files, such as PNG or JPEG
- Create PDFs – Create a PDF from scratch, with embedded fonts and images
- Signing – Digitally sign PDF files

According to the requirement of our application and the module, we wanted to extract the text and the text properties from the PDF Documents. With the capabilities provided with the PDFBox PDF Document parser, we could achieve this requirement.

Sample Code Snippet used in the project module for reading the PDF Document

```
public void readPdfDocument(File file) {  
    PDFParser parser = null;  
    String text = "";  
    PDTextStripper stripper = null;  
    PDDocument pdoc = null;  
    COSDocument cdoc = null;  
    try {
```

```

        parser = new PDFParser(new FileInputStream(file));
    } catch (IOException e) {
        e.printStackTrace();
    }
    try {
        parser.parse();
        cdoc = parser.getDocument();
        stripper = new PDFTextStripper();
        pdoc = new PDDocument(cdoc);
        stripper.setStartPage(1);
        stripper.setEndPage(pdoc.getNumberOfPages());
        text = stripper.getText(pdoc);
        docLines = text.split("\\r\\n");
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        try {
            cdoc.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

4.1.4.2 Stanford CoreNLP

Stanford CoreNLP is another open source tool set which contains a number of Natural Language Processing tools such as Part of Speech (POS) Tagger, Name Entity Recognizer (NER) classifier, parser, the coreference resolution system, sentiment analysis and the bootstrapped pattern learning tools. All these tools are designed to be compatible with the English language [5].

During the implementation of the information extraction module of the system as well as in the online profile information extraction module, we encountered situations to identify Natural Language properties of the plain texts. Due to this reason, rather than re implementing the tools

we used the Stanford CoreNLP Natural language-processing toolkit. During the resume information extraction process, we encountered a situation in which we wanted to identify the Organization names, Person names, Dates, Locations, etc. We had to identify this information when we try to extract referee information from the resumes, Previous Work History information of the candidates, previous work duration of the companies, etc.

In the Stanford CoreNLP tool set, they have provided the facility to identify the above-mentioned information. All the above-mentioned information are related to one of the major branch of natural language processing, which is Name Entity Recognition (NER). In Stanford CoreNLP toolkit, they have three types of NER classifiers embedded with it. Depending on the requirement of the application, we can decide which type of the classifier to be used. Those classifiers as follows,

3 class Classifier: Location, Person, and Organization

4 class Classifier: Location, Person, Organization, Misc

7 class Classifier: Time, Location, Organization, Person, Money, Percent, and Date

During our implementation of the module, we decided to use the 7-class classifier since we need to identify the Dates also.

4.1.5 Resume Information Extraction Module Implementation

In the module architecture section, we described the information extraction procedure as an overview. In this section, we hope to provide a detailed description about the implementation perspectives of the module.

Let us take a quick review on the information extraction process's steps.

- Step 1 : Validate and Read the document uploaded
- Step 2 : Parse the plain text Lines to identify the lines which are possibly headings and index them
- Step 3 : Extract Information from the Identified sections.
- Step 4 : Re parse the remaining lines in order to extract possibly missed information

Now let's have a deeper look at these each step one by one.

At step 1 the document is validated in order to determine the document is in a valid format before parsing and if the document is validated by the module, it is being read with the Apache PDFBox PDF parser tool. In our approach, we only extract the contents of the document as plain text. As I mentioned earlier PDFBox provides the capabilities to extract the text properties also. As an example, Underlined Texts, Bolded texts, Bullets, Font sizes, etc.

Before implementing the currently using version of the information extraction module, we implemented two other modules to extract the information from the resumes. One of these modules were designed based on identification of resume specific text property characteristics. As an example the bullets being used for listing technological expertise, etc. After going through some feasibility analysis and performance evaluations, we decided to get rid of that module due to its poor performance and the over fitting nature of the module. Therefore, the text property extraction capability of the PDFBox parser is not used in the current resume information extraction module.

Let's take another look at the text parsing code snippet above. It clearly shows that the parsed text is being returned as a plain text string, by the parser. Also this string contains the Unicode characters for the special symbols as well. Such as the bullets, etc. Since we only need the plain text, we first need to split the single string into a set of lines by carriage returns and the new line characters. Then we need to remove the special character, which we do not frequently use during the writing. When removing the special characters we had to be careful to avoid removing the special characters such as @, #, %, \$, !, *, etc. because these characters are used frequently when we are writing valuable information. As an example, the email addresses, programming Languages such as C#, etc. At the end of the first step, we have already extracted the text lines from the document as plain text strings. These lines are being kept in an ArrayList as well as a copy is also kept since we need to remove processed lines before we extract missed information at the last step.

Step 2 is the most important step in this information extraction process as described earlier. This is because our resume information extraction module is based on identification of headings of the resume and categorize these headings into a set of identified heading types and then information is extracted from the resume under these headings. Now let's have a detailed understanding about the procedure of identification of the headings of the resume.

As described in an earlier section resumes have these headings and under these headings, candidates include information. When we consider the headings of the resumes, we can find several important as well as common features for each resume. Since we use resumes to include only a selected set of information types these information can be divided in to more generic heading types. In this project as a proof of concept, we develop the application for IT industries. We used a sample of 150 resumes and based on these resumes we identified the included headings are similar and only the representation is different from one resume to the other. In other words, a person can include his/ her educational information under a heading such as “*Educational Qualifications*”. Some other person can include the same type of information under a heading such as “*Academic Qualifications*”. If we examine a large number of resumes, we can clearly identify this similarity. In our procedure, we used a special technique to identify these headings. For this purpose, we keep a collection of “*Gazetteer Lists*”. For each of the heading type we have a Gazetteer list containing a list of identified headings, which can be used to map the lines and decide whether it belongs to a certain heading. At this step all the lines of the document after initial parsing is mapped with the Gazetteer Lists and if a match is found then these are indexed with a unique identifier for a heading type. Only mapping with the Gazetteer lists is not enough to identify the headings. This is because the same heading phrase may be included in a non-heading line. In order to avoid arising such false positives we used identification of common writing patterns, which we normally use. We usually do not have headings in the resumes, which have a large number of words. Do you commonly see headings having about 15-20 words in a resume? The simple answer is a NO. Do you start a heading with a simple letter when you write resumes? Again, the answer is a NO. We used these two properties to avoid the false positives we collected at the earlier implementations of this module.

After identifying and properly indexing the lines, we come to the next most important step of this module, which is the information extraction modules. Let’s again have a look at the “*Resume Information Groupings*” Figure. When you look at it, you can identify that, between two headings information regarding to the previous heading is included. Also we described earlier that we cannot have a generic module to extract all types of information from each of the information categories. Due to this reason as shown in the layered architecture diagram of the module, we used sub information extraction modules to extract information from each type of information category. Current sub information extraction modules are as follows.

- Achievements Information Extractor
- Educational Information Extractor
- Interests Information Extractor
- Personal Information Extractor
- Project Information Extractor
- Referee Information Extractor
- Work Information Extractor

Each type of the above information extractor extract relevant information depending on the type of information to be extracted.

As an example Educational Information extractor, extract educational information such as School Names, College Names, Time Durations, Degree type, etc. Referee information Extractor extracts information such as Referee name, Referee contact details, etc. Work Information extractor extracts information such as Worked Organization, Time Duration, Post, etc.

After this step we have another important step, in which we try to examine and extract valuable information from the text lines we could not extract information. This is because as we mentioned earlier, to avoid the impact of the erroneous heading identifications.

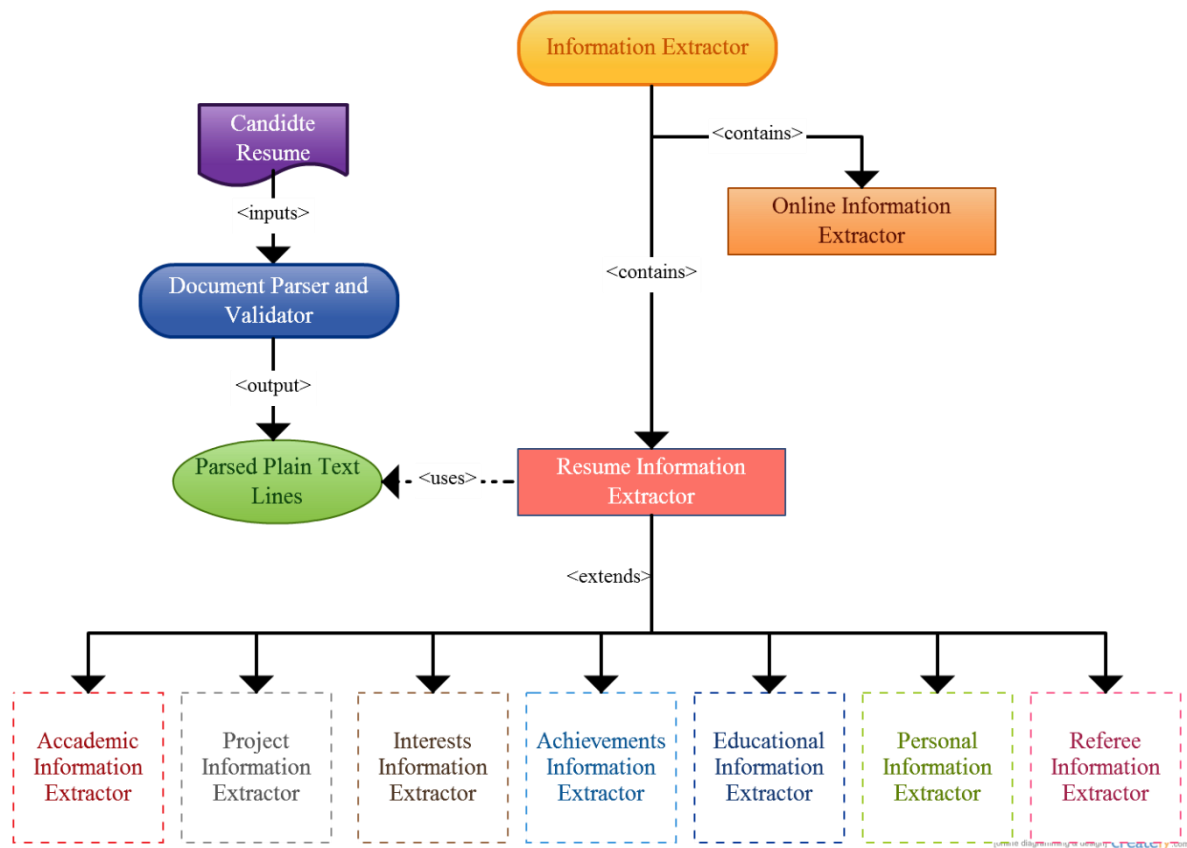


Figure 4. 4 Info Extractor Module Pipe Line

Above diagram implicitly shows the structure of the information extraction module of the system. As shown in the diagram Document parser and the validator takes the candidate resume as the input and then it outputs the parsed plain text lines for the processing at the information extractors. All the information extractors are extended from the information extractor module and the implementation is done depending on the type of information being extracted at the extractor.

During all these steps in this module, this extracted information is bind to a Candidate object, which is used for the processing of information throughout the system. After completing the processing stages of this module, our next step is handing the candidate object with extracted information to the next module, which is the online information extraction module.

4.1.6 Results and Analysis

4.1.6.1 Information Grouping Identification

In this section, we are going to analyze the results of the resume information extraction module. For this module's testing purposes we used 100 resumes and 65 of them were used as the training set of the module and 35 were used as the test set of the module. Depending on the results of the test set, we are analyzing these results as follows.

First of all it is important to take a look at the header identification test results of the module and all these test results are compared against the human ratings.

Table 4. 1 Heading Identification Statistics

Resume	Educational		Project		Interests		Achievements		Personal		Referee		Work	
	H/R	A/O	H/R	A/O	H/R	A/O	H/R	A/O	H/R	A/O	H/R	A/O	H/R	A/O
1	1	1	1	1	1	1	2	2	2	1	1	1	1	1
2	2	1	1	1	0	1	1	1	2	1	1	1	1	1
3	1	1	1	2	0	0	1	1	2	2	1	1	1	1
4	1	1	1	1	1	1	1	1	2	1	1	1	1	1
5	1	1	1	1	1	1	1	1	2	1	1	1	1	1
6	1	1	2	2	1	1	1	1	2	1	1	1	1	1
7	1	1	2	2	1	1	1	1	2	1	1	1	1	1
8	1	1	2	2	1	1	1	1	2	1	1	1	1	1
9	1	1	2	2	1	1	2	1	2	1	1	1	2	2
10	1	2	2	2	1	1	1	1	1	1	1	1	1	1
11	1	1	2	1	1	1	1	1	1	1	1	1	1	1
12	2	1	2	2	1	1	2	2	2	1	1	1	2	1
13	1	1	1	1	1	1	2	1	1	1	1	1	1	1
14	2	1	1	1	1	1	2	2	1	1	1	1	2	1
15	2	1	1	1	1	1	2	2	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1	2	1	1	1	1	1
17	2	2	1	1	2	2	1	1	1	1	1	1	2	1
18	1	1	2	1	1	1	1	1	1	1	1	1	2	1
19	1	1	2	2	1	1	1	1	1	1	1	1	2	1
20	1	1	2	2	2	1	1	1	2	2	1	1	2	1
21	1	1	1	1	1	1	1	2	2	1	1	0	3	1
22	1	1	1	1	1	1	1	1	2	1	1	1	3	1
23	1	1	1	1	1	1	1	1	2	1	1	1	1	1
24	1	1	1	1	2	1	1	1	2	1	1	1	1	1
25	1	1	1	1	1	1	1	1	1	0	1	1	1	1
26	1	1	1	0	2	1	1	1	1	1	1	1	1	1
27	1	1	1	0	2	1	2	1	1	1	1	1	2	1
28	1	3	1	1	2	1	2	1	1	0	1	1	1	1
29	1	0	1	1	2	2	2	1	1	0	1	1	1	1
30	2	2	1	1	2	2	1	1	1	1	1	1	1	1
31	1	1	1	1	1	1	1	1	1	0	1	1	2	1
32	2	1	1	1	1	2	1	1	1	1	1	0	3	2
33	2	1	1	1	1	1	1	2	1	1	1	1	1	1
34	1	1	1	1	1	1	1	1	1	1	1	1	1	1
35	1	1	2	1	1	1	1	1	1	1	1	1	1	1
Total	43	39	46	42	41	38	44	41	51	33	35	33	50	37

In the above table, there are statistics included for 35 test resumes of our test set. According to the test results, below contingency tables are created for each type of header type.

Educational Headings

Table 4. 2 Educational Heading Contingency Table

	Relevant	Non Relevant
Retrieved	36	3
Not Retrieved	6	267

Precision = 0.92

Recall = 0.85

Combined F Measure = 0.88

Project Headings

Table 4. 3 Educational Heading Contingency Table

	Relevant	Non Relevant
Retrieved	41	1
Not Retrieved	3	264

Precision = 0.97

Recall = 0.93

Combined F Measure = 0.94

Interests Information Heading

Table 4. 4 Interests Heading Contingency Table

	Relevant	Non Relevant
Retrieved	36	2
Not Retrieved	5	269

Precision = 0.94

Recall = 0.87

Combined F Measure = 0.90

Achievements Information Heading

Table 4. 5 Achievements Heading Contingency Table

	Relevant	Non Relevant
Retrieved	39	2
Not Retrieved	5	266

Precision = 0.95

Recall = 0.88

Combined F Measure = 0.91

Personal Information Heading

Table 4. 6 Personal Heading Contingency Table

	Relevant	Non Relevant
Retrieved	33	0
Not Retrieved	18	259

Precision = 1.0

Recall = 0.64

Combined F Measure = 0.78

Referee Information Heading

Table 4. 7 Referee Heading Contingency Table

	Relevant	Non Relevant
Retrieved	33	0
Not Retrieved	2	275

Precision = 1.0

Recall = 0.94

Combined F Measure = 0.96

Work Information Headings

Table 4. 8 Work Information Heading Contingency Table

	Relevant	Non Relevant
Retrieved	37	0
Not Retrieved	13	260

Precision = 1.0

Recall = 0.74

Combined F Measure = 0.85

Overall Comparison

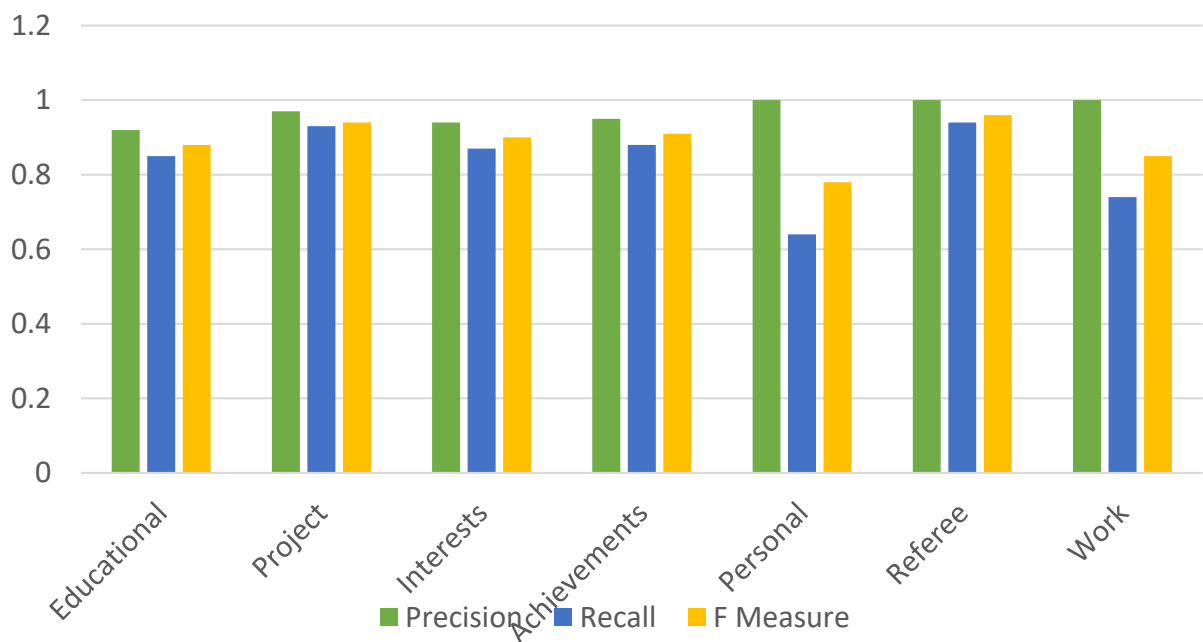


Figure 4. 5Overall Comparison Heading Identification

According to the above test results depending on the type of information, F measure differs. In most of the times according to the test results, precision takes a considerable higher value, while the recall sometimes takes lower values. One reason for this type of difference is the errors of the parser. This is because the PDF Parser always do not exactly extract the lines in the relative order. Sometimes the PDF Parser encounters issues regarding the text boxes and

the text styles misidentifying as images. In such situations, parser fails to convert those in to the text lines and missing information leads to such kind of lower recalls.

4.1.6.2 Information Extraction from The Groups

In the above analytics we have been shown the precision, recall and the Combined F-Measure values for identification of the information groupings from our Resume Information Extraction algorithm. In this section we are going to show the analytics of extracting the information from each of the information groupings.

Educational Information Extraction

Table 4. 9 Educational Information Extraction Contingency Table

	Relevant	Non Relevant
Retrieved	58	13
Not Retrieved	6	463

Precision = 0.0.82

Recall = 0.91

Combined F Measure = 0.86

Project Information Extraction

Table 4. 10 Project Information Extraction Contingency Table

	Relevant	Non Relevant
Retrieved	189	29
Not Retrieved	36	305

Precision = 0.86

Recall = 0.84

Combined F Measure = 0.85

Achievements Information Extraction

Table 4. 11 Achievements Information Contingency Table

	Relevant	Non Relevant
Retrieved	90	19
Not Retrieved	67	370

Precision = 0.82 Recall = 0.57 Combined F Measure = 0.67

Personal Information Extraction

Table 4. 12 Personal Information Contingency Table

	Relevant	Non Relevant
Retrieved	165	34
Not Retrieved	20	352

Precision = 0.82 Recall = 0.89 Combined F Measure = 0.85

Referee Information Extraction

Table 4. 13 Referee Information Contingency Table

	Relevant	Non Relevant
Retrieved	47	3
Not Retrieved	3	477

Precision = 0.94 Recall = 0.94 Combined F Measure = 0.94

Work Information Headings

Table 4. 14 Work Information Heading Contingency Table

	Relevant	Non Relevant
Retrieved	29	26
Not Retrieved	5	493

Precision = 0.53

Recall = 0.85

Combined F Measure = 0.65

Overall Comparison

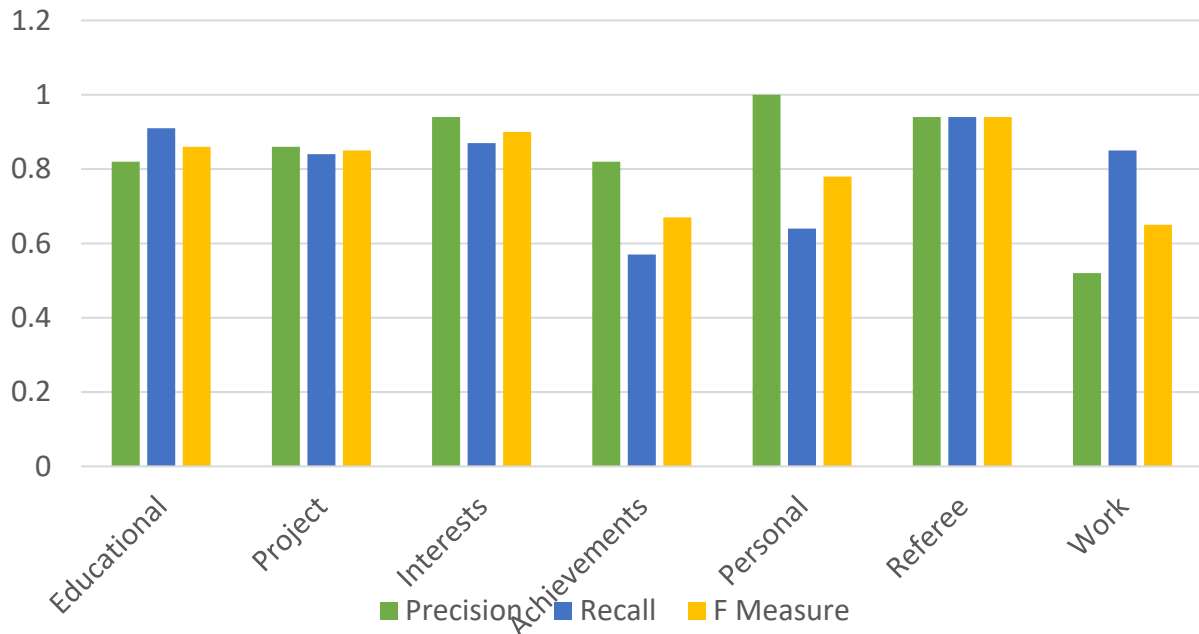


Figure 4. 6 Overall Comparison Information Extraction

4.2 Aggregated Profile Generator

4.2.1 Introduction

Aggregated profile generation means making one profile for one person or entity using the information retrieved from various sources. The objective of Warana is to find the most suitable candidates for a given organization. In order to find the best matches, it is not enough to consider only the information extracted from CV. Because, the candidates cannot include everything about them in a two or three page CV. In addition, the more we know about the candidate, the more we can accurately predict their suitability for an organization.

4.2.2 Information Sources

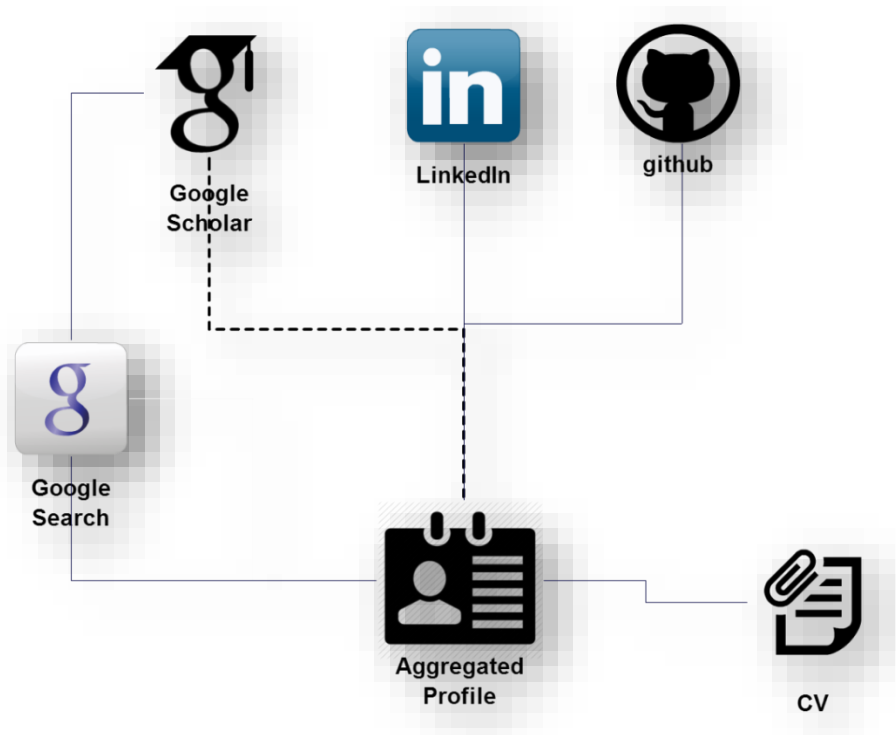


Figure 4. 7 Information Sources

Above figure represents the information sources used in generating aggregated profile. The reason to use a dotted line to connect Google scholar is, it does not have an API. Therefore, the corresponding Google scholar profile for a candidate has to be found via Google search. Initially, by processing CV, all the information is extracted. Then using the information extracted from the CV, other profiles are found using web services and provided by respective websites and Google search.

4.2.2.1 Verifying profile

There are many people with the same name. Therefore, when looking for online profiles of some candidate, the search engines can give false results. Therefore, we implemented a mechanism to make sure that the online profile represents the actual person. Before integrating extracted information from online profile with the main profile, we make a temporary profile using information extracted from the online profile. Then by comparing that profile with the existing profile, we verify whether both profiles represent same person.

4.2.2.2 LinkedIn

LinkedIn is the largest professional network in the world. Therefore, almost all the people have created LinkedIn profiles. Therefore, it is a good source to extract information about candidates. Our system extracts educational details, work experience, projects details, publication details, endorsements from candidates LinkedIn profile upon the availability of those information.

4.2.2.3 Google Scholar

Google scholar is a place where authors of research papers can publish details about their work. If the candidate have published any research papers, most probably we can find information about it in Google scholar. In addition, we can extract information about the co-authors of the research papers. However, Google scholar does not provide an API. Therefore, only available option for extracting information from Google Scholar was web scraping. Nevertheless, the risk of web scraping is, if the structure of the website is changed, scraping would not work as expected. For web scraping, we used Jsoup library.

4.2.2.4 GitHub

GitHub is the largest website where we can find the repositories of many open source projects. GitHub provides an API to retrieve information about the public repositories and their contributors. By analyzing the repositories a candidate has contributed, we can identify what are the languages he is familiar with, what kind of technologies he is good at,...etc.

4.2.3 Skill Recognition

In the aggregated profile we intended to create for person, we needed to identify the skills possessed by the person. Skills are not just some skills; they are more like the key words, which represent their career, abilities, industry exposure and passion.

4.2.3.1 LinkedIn Endorsements

From the sources available in the internet, LinkedIn endorsement are the most suitable source to extract skills for profiles. However, the problem is, endorsements are not trustworthy. Because, anybody in LinkedIn can endorse his/her connections without really knowing the

other ones skills. Therefore, we implemented a methodology to confirm whether the interested person actually possesses those skills found on endorsements.

4.2.3.2 Skill confirmation methodology

In order to confirm whether the person has the skills listed in LinkedIn, first we retrieved all the related documents to the person (candidate corpus). That includes blog posts, publication details, personal details, projects details and experience details. Then we extracted a list of key terms from the candidate corpus (candidate list list). Then, for each skill that the person claims to have, we built a corpus by retrieving up to 10 documents related to the skill. Then, we extracted a list of key terms from each individual corpus for each skill (skill lists). Then by comparing the candidate list with skill lists, we measured the similarity of two lists for each skill. By defining a suitable threshold, we found the skills the person actually possess.

In key term extraction process, we used an aggregated algorithm developed by weighted aggregation of common automatic term recognition algorithms. Next section contains details about the new algorithm we developed.

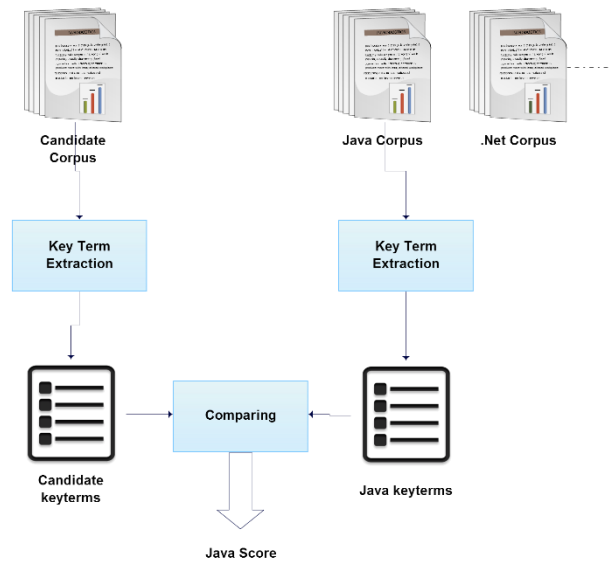


Figure 4. 8 Score Generation

4.2.3.3 Extracting blog/web articles

Nowadays most of the websites are not just simple HTML pages. Most of the parts are loaded after calling JavaScript functions. However, when a webpage is loaded through a java program by simply using a HTTP request, these functions are not called. Therefore, the program would not retrieve the complete web page. In order to avoid this problem, we used selenium web driver. Using the selenium web driver, a browser window is opened and the desired webpage is loaded to the browser. After that, the source of the webpage is retrieved to the program from the loaded web page. In this way, the complete source code of the web page could be retrieved.

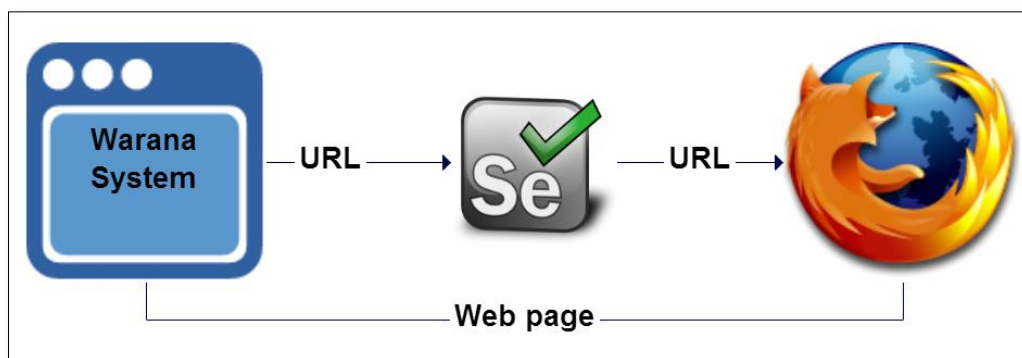


Figure 4. 9 Web Article Extraction

4.2.3.3.1 *Web Crawler*

In the generated aggregated profile, only the homepage URL of the candidate's blog or website could be found. Then how do we retrieve all the articles in written by the candidate. For that, we should have the URLs of all the articles. For that, we implemented a web crawler. When a URL is provided to the web crawler, it finds all the different URLs under the given URL. Those URLs represents the different articles or pages. Those articles are used in building the user corpus.

4.2.4 Synergistic Approach to Key Term Extraction

4.2.4.1 Introduction

The objective of terminology extraction is extracting relevant terms from a given corpus. In this research, we analyzed the performance of eight key term extraction algorithms and how does it vary from corpus to corpus depending on the nature and the quality of the corpus. We measured the performance of algorithm by comparing the list of key terms obtained using each algorithm with the golden standard. Based on the results we obtained from the analysis we performed, we implemented a new improved algorithms for key term extraction by aggregating those eight existing algorithms we analyzed. In addition to aggregation, we introduced several mechanisms to improve the performance of aggregated algorithm such as common word removal and abbreviation integration. Those mechanisms are explained in detail later in the paper. The eight existing algorithms we used are,

- Average corpus term frequency algorithm
- C value algorithm[13]
- Glossex algorithm[22]
- RIDF algorithm[23]
- Simple term frequency
- Termex algorithm[24]
- TF-IDF
- Weirdness[25]

4.2.4.2 Setting up the Experiments

4.2.4.2.1 Building the Corpus

For the experiment, we implemented a program, which retrieve articles from the web for the corpus on given topic. For example, if we need to build a corpus on computer science, the program retrieves and stores up to 10 articles available in the web related to the topic “Computer Science”. Here we assumed that top 10 results given by Google search on the given topic are the most relevant articles. We observed that most of these results contained articles from Wikipedia. Likewise, we built 20 corpora on different topics. For each corpus we built, we built a “Golden Standard” which contains the most relevant list of key terms of the corpus.

4.2.4.2.2 Common Word Removal

As mentioned above, in corpora we built, most of the articles are retrieved from Wikipedia. When the majority of articles are from same source, there is a potential to identify the terms, which represents metadata about the source as key terms of the corpus. For example, in many Wikipedia articles there is a sentence, which says “From Wikipedia, the free encyclopedia”, right after the topic of the article. Therefore, we developed a mechanism to identify and remove such words by inspecting the entire corpora.

This mechanism is based on the assumption that, “if one key term has a high significance on majority of individual corpus across the entire corpora, then that term is not so significant for the respective topic of the corpus.” Therefore, it is reasonable to assume that such key terms includes the metadata, which represents the source of article. The reason we cannot simply put those terms into a stop list is those terms cannot be predicted earlier. They depends on the topics we choose in building corpora and the sources we use. Therefore, we implemented a program to search and find key terms from entire corpora and then remove the key terms with a higher score than a predefined threshold from each individual key terms list identified from searching in each individual corpus.

4.2.5 Analysis of Algorithms

Before implementing the aggregated algorithm, we first analyzed the performance of each individual algorithm by comparing the results obtained in each algorithm on same corpus against the golden standard. In this case, we used 20 different corpora to compare the performance of each algorithm.

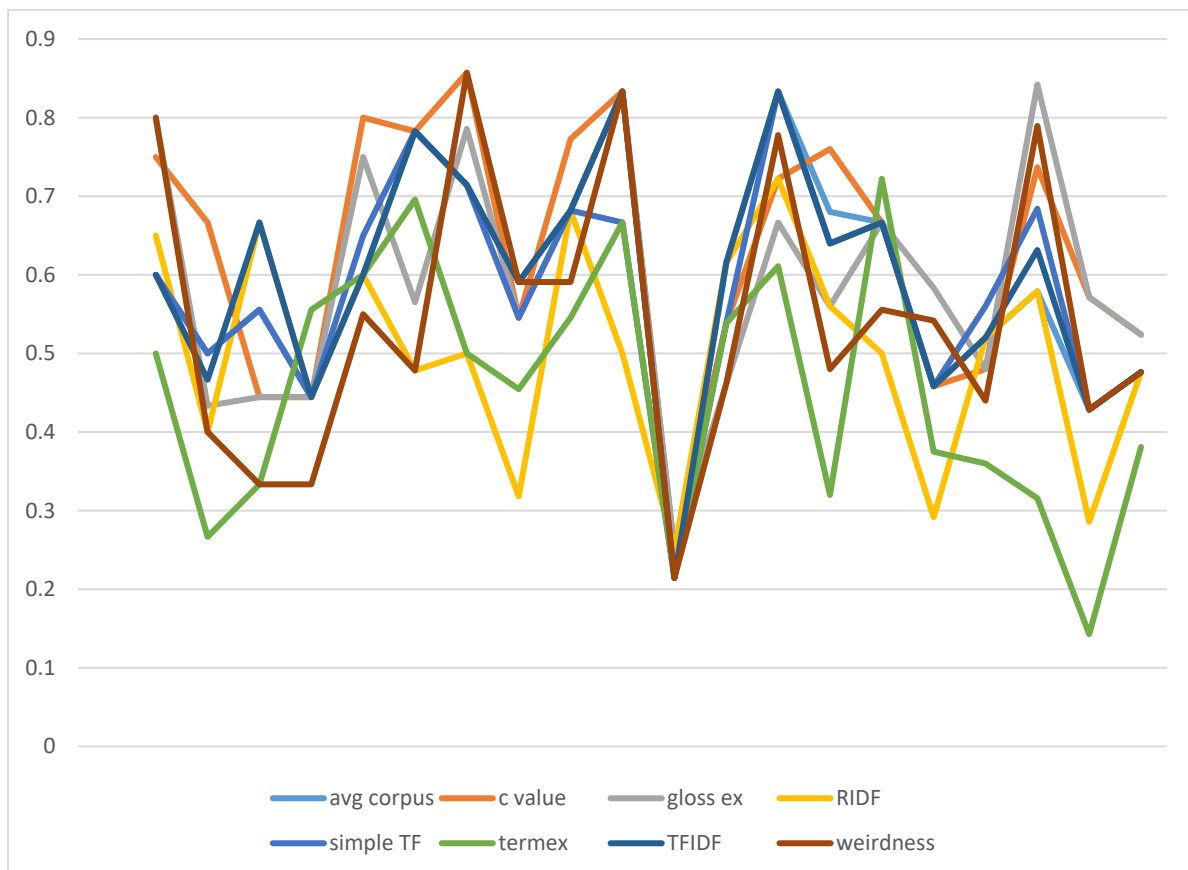


Figure 4. 10 Algorithmic Performance

According to the observations, the precision of the algorithms were varied among different corpora. Each algorithm makes different assumptions on the number of words of a key term. Therefore, it would be important to study the performance of algorithms against allowed max number of words per key term.

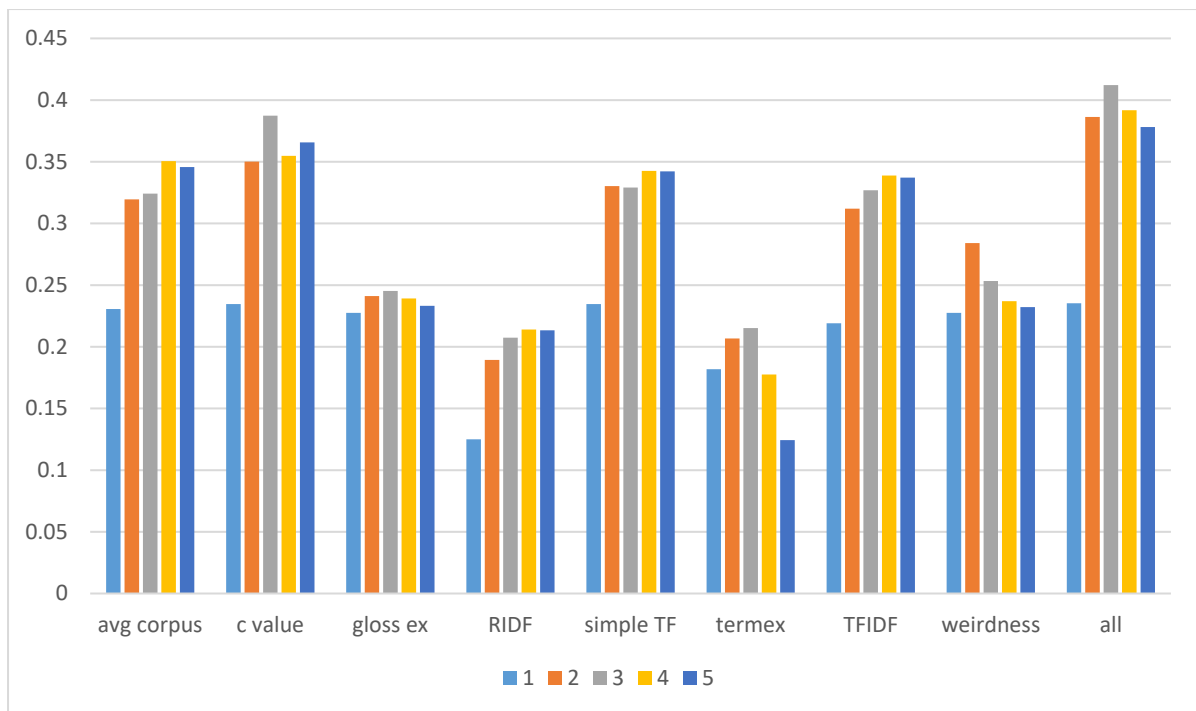


Figure 4.11 Key Term Performance I

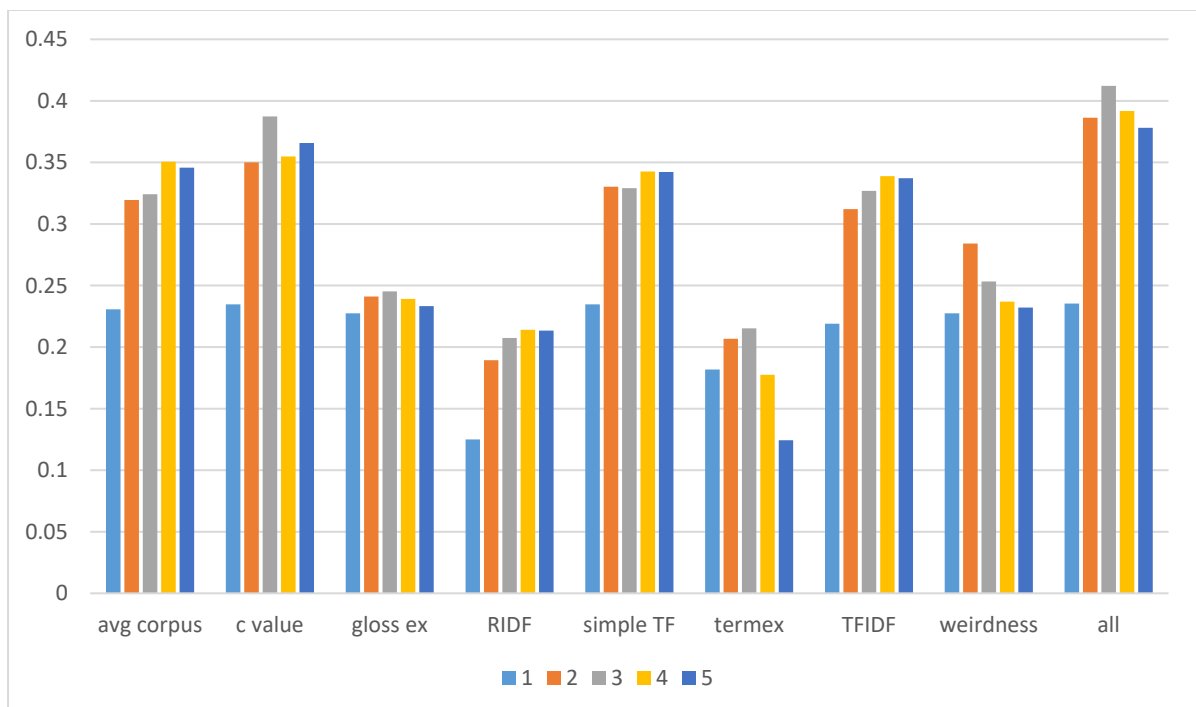


Figure 4.12 Key Term Performance II

By analyzing this graphs, we can conclude that, majority of the algorithms show best performance with 3 for the allowed number of words for key terms. Therefore, from now on, we have used 3 as the maximum number of words per key term in our research.

4.2.6 Weighted Aggregation of Algorithms

According to the corpus, precision of algorithms changes. Following bar graph shows the average precision of each algorithm and the combined algorithm without weighting. The precision is measured by comparing the obtained key terms list from each algorithm against the golden standards. By observing the graph shown in figure 4.11, we can conclude that, C-value algorithm shows the best precision at key term extracting.

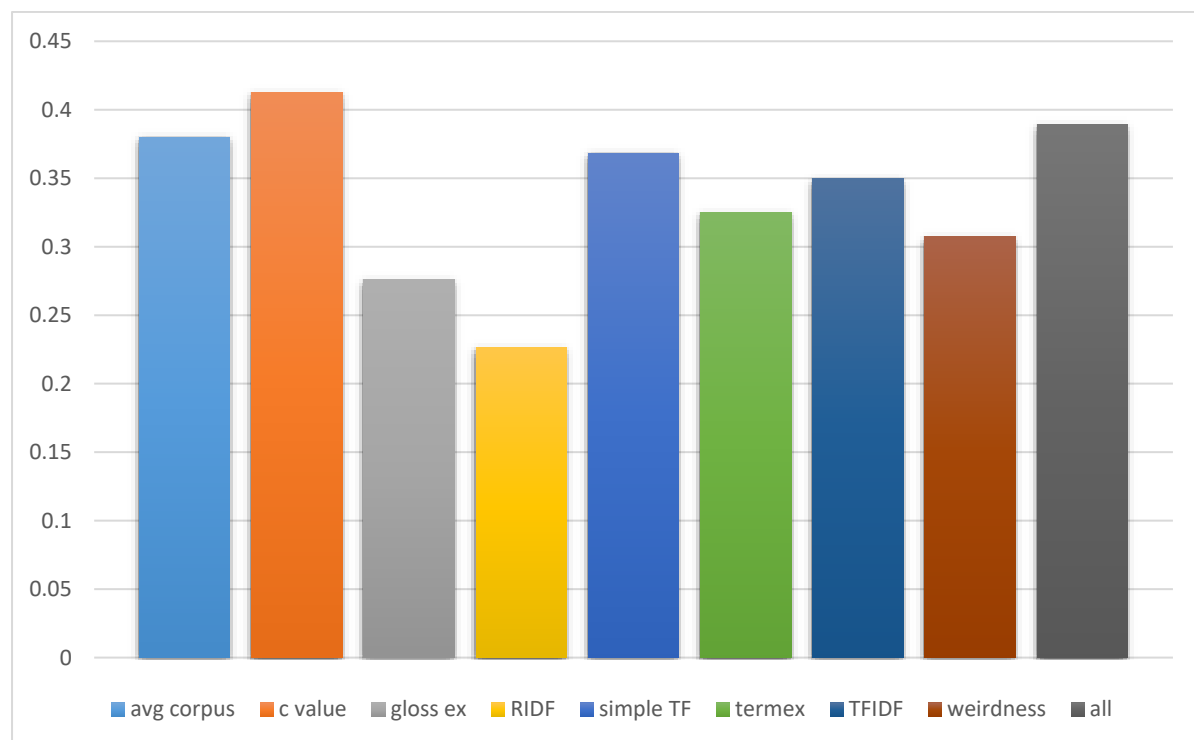


Figure 4. 13 Average Precision

4.2.6.1 Learning Weights

We developed an iterative mechanism to adjust weights for next iteration based on the weights of previous iteration and the average precision of all individual algorithms.

$$W_{next} = \text{Max}(W_{previous} + \eta(P - P_{average}), 0)$$

Equation 4.1 Learning Weights

P represents the precision of algorithm being considered. Following graph shows the variation of average precision over corpora with the iteration. Initial weight is 1 for each algorithm.

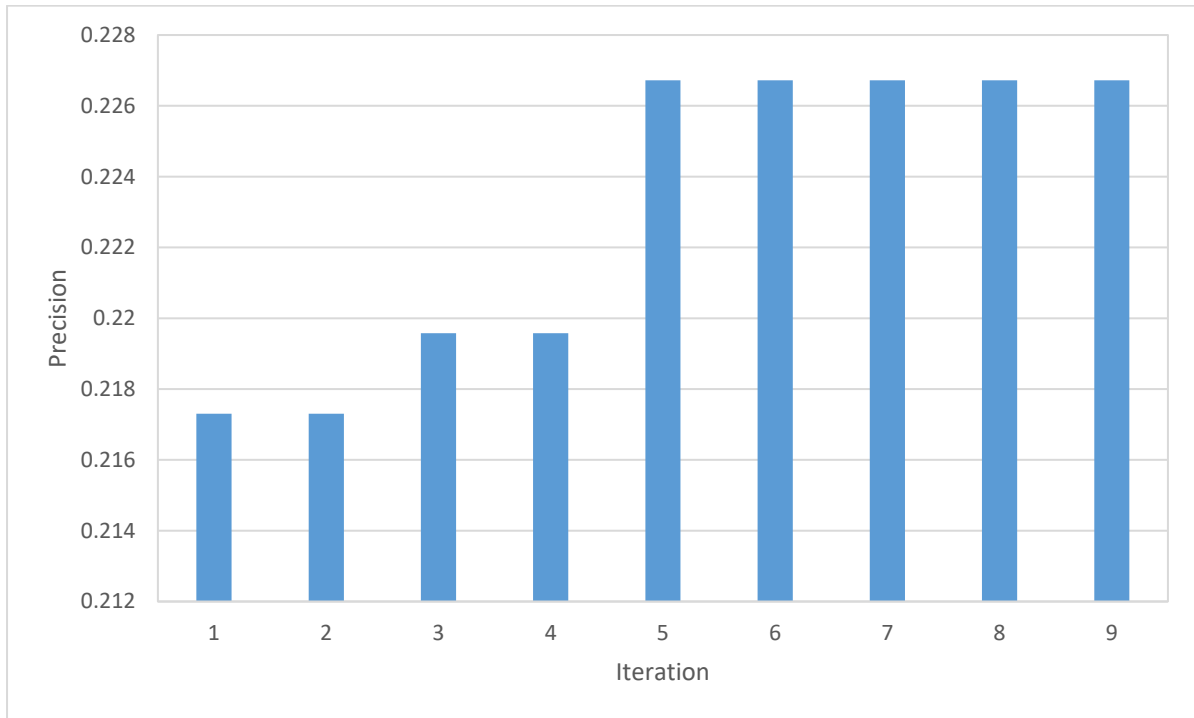


Figure 4. 14 Weighted Aggregation Precision

According to the above graph, we can see that the precision of weighted aggregation of algorithms reduces the average precision. Therefore, we can conclude, the optimum value for the weight of each algorithm is 1. In other words, aggregating algorithms without weighting. This happens because, the average precision of each algorithms have a little difference over the corpora.

4.2.7 Analyzing & improving aggregated algorithm

This section explains the analysis and methods we applied in order to study and improve the aggregated key term extraction algorithm.

4.2.7.1 Search Depth

In above graphs, we have considered only top 20 results of the outputs of algorithms. Then we analyzed the variation of recall for each algorithm with the depth of the list we considered.

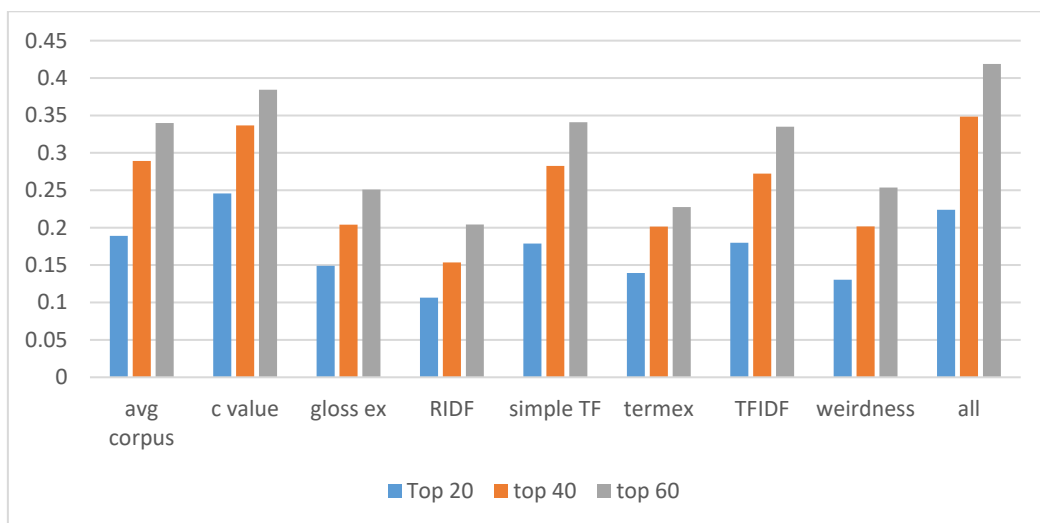


Figure 4. 15 Recall variation with depth

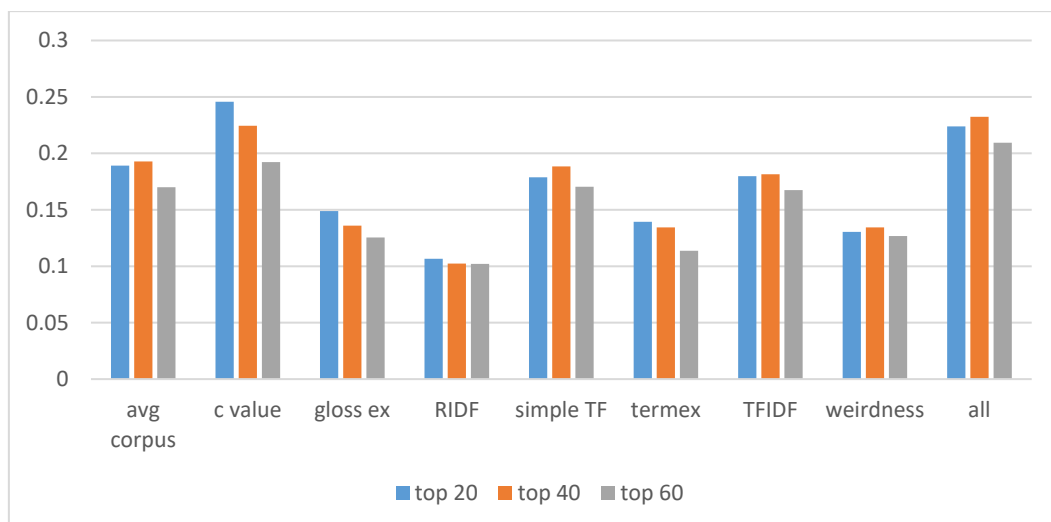


Figure 4. 16 Number of Top n Results(II)

At the depth of 40, aggregated algorithm have surpassed the recall of C-value algorithm. Therefore we can conclude from this graph, even if the C-value algorithm could accurately predict the top keywords for a given corpus, the ability of aggregated algorithm to retrieve keywords is high.

4.2.7.2 Abbreviation Integration

Even though Termex and Glossex algorithms claimed that, they take abbreviations and acronyms into consideration when identifying keywords, most of the time, those algorithms have failed to identify a particular term and the abbreviation of the same term as one term. Therefore, we developed a method to identify and group the abbreviation and its long form as one entity. Following graph compares the precision of aggregated algorithm before and after abbreviation integration.

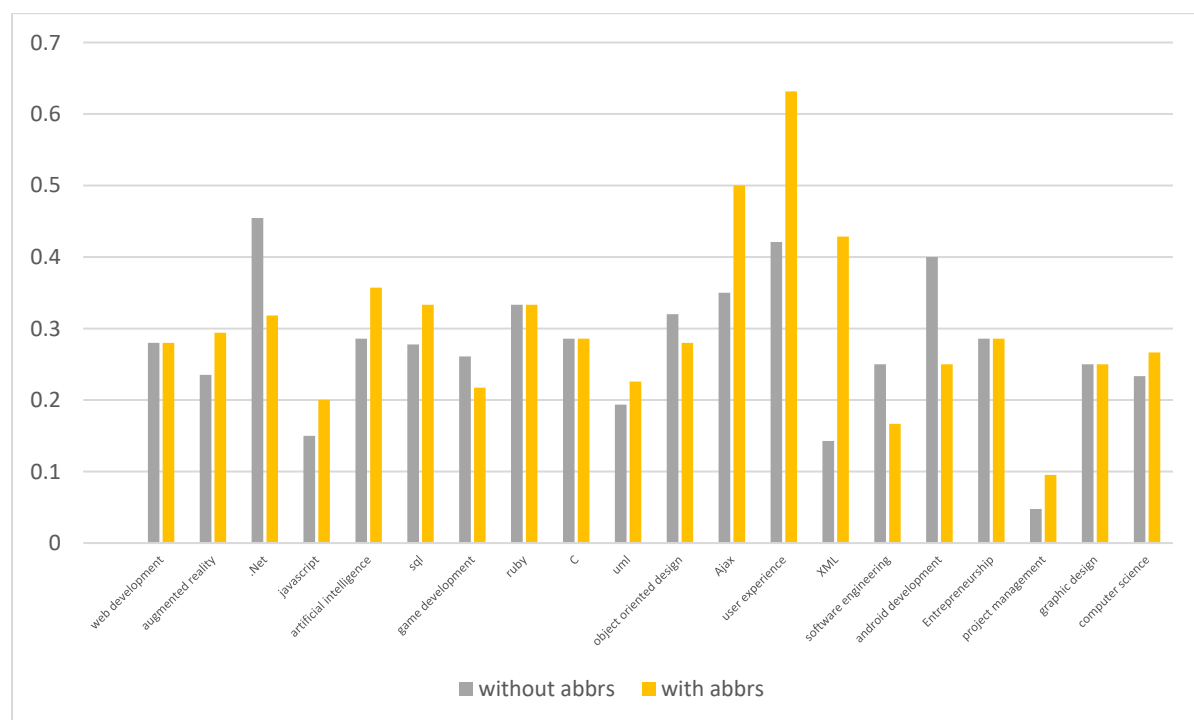


Figure 4. 17 Precision before and After Abbreviation Integration

By observing the graph, we can conclude that the precision of the algorithm has considerably improved after abbreviation integration.

4.2.8 Discussion on new Algorithm

In this study, we have used Wikipedia as the source to find articles for 20 different corpora we build in order to analyze the performance of algorithms. Characteristics of key terms differs from corpus to corpus. In addition, the quality of corpus has a direct impact on the results obtained by algorithms. Also on some topics, number of related articles found in Wikipedia was limited while other topics have a plenty of articles. This directly affects the quality of corpora. The reason not to regulate this condition is, in practical application, we cannot always expect to get equal quality corpora and therefore we must implement our methodology in a way it can perform best on practical applications. It can be a reason to vary the performance of term extraction algorithms on different corpora.

Even though weighted aggregation of algorithm did not show a massive performance improvement over other algorithms, in terms of robustness, it has outperformed other algorithms. Because, even if some algorithms have shown a slightly better performance in some corpus than aggregated algorithm, that particular algorithm can perform very poorly in another corpus. However, aggregated algorithm has always maintained average performance over each corpus. In future work, we hope to implement a condition based weighting mechanism to aggregate algorithms. For that, we have to study the correlation between different algorithms and implement an automated performance measure.

By analyzing the recall and F1 measure of algorithms against the depth of the extracted key terms list, we expected to measure how the ability to retrieve more keywords. For example, even though some algorithms showed a better performance in top 20 results, after that, the results contains more key terms, which mostly contain top results as the part of term. This reduces the ability to retrieve more different key terms. According to the results, we could confirm that the aggregated algorithm has retrieved more key terms than all other algorithms even if in some cases, it has shown a slightly lower performance in top most results.

In most key term extraction algorithms, methods like stemming and lemmatization are used in order to identify multiple forms of same words as a single word. However, it does not consider abbreviations and acronyms. GlossEx claims that it identify abbreviations and full forms and remove redundant terms from extracted lists. Termex also claims that that it takes acronyms into account. Still in the implementation we used for the research, it does not seem that those

algorithms have successfully identified abbreviations and their full forms. Even if these two algorithms did take abbreviations and acronyms into account, since the other six algorithms do not consider them, the results of aggregated algorithm will not reflect that it has identified abbreviations and their respective long forms are identified as a single entity. That is why we developed an abbreviation integration mechanism for aggregated algorithm. According to the results we obtained, abbreviation integration has improved the performance of aggregated algorithm considerably.

4.3 Extracting technologies from company knowledge base

4.3.1 SigmaC Document analysis based Automatic Concept Map Generation for Enterprises

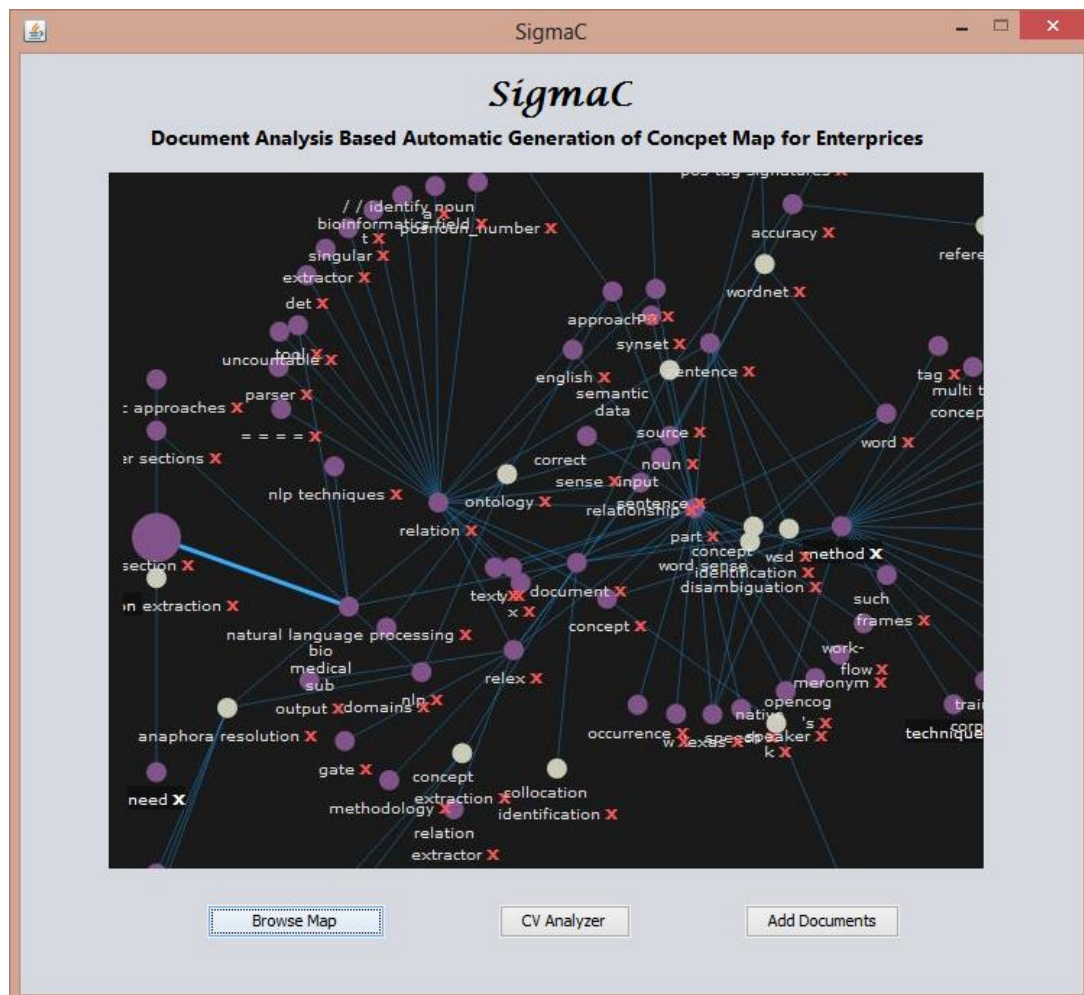


Figure 4. 18 SigmaC App

In order to compare a particular candidate with the company it needed per-define knowledge base. This was done by using SigmaC Document analysis application, which is developed by 08 batch. In this application, it provides a concept map according to the documents, which is, consist with technical and technological details in a particular company.

In this concept generating map application, high-level natural language processing techniques and document preprocessing techniques are used to generate concept map. Following are objectives that have been successfully achieved.

4.3.1.1 Aim of SigmaC

Introducing novel way to retrieve written knowledge of company by creating concept map using accurate concepts and relationship extraction methods.

4.3.1.2 Objectives of SigmaC Application

- Organizing enterprise knowledge base in convenient manner
Internal personal in an enterprise uses this knowledge base documentations frequently and the knowledge base is becoming bigger and bigger. By this research project we are looking for a convenient way for browsing and extracting relevant information using the idea of concept maps.
- Support all types of knowledge base documents
Currently many document types store enterprise knowledge base. This project is expected to support common document types that gather the knowledge base of an enterprise.
- Construct a word graph which would have relationships among words
The suggested system would crawl through the documents in the organization and construct a word graph which would have relationships among words (is a, part of, etc.)
- Some of the relationships can be weighted
Weightings to determine the strength of the relationship
- Editable and self-adjustable concept map
The map should be editable (add, remove relationships) and the map should self-adjust with the edits. (Other edges also to be recomputed based on the edits).

- Provide an intuitive UI

Show the graph of words and a mechanism of browsing through the documents of the organization by clicking on the words in the graph. It would be a novel way of browsing the knowledge base of the organization.

4.3.2 SigmaC Application Architecture

This application consist with three main layers. Those are consider to be as document per-processing, core processing and front end. Below image shows layers and it components.

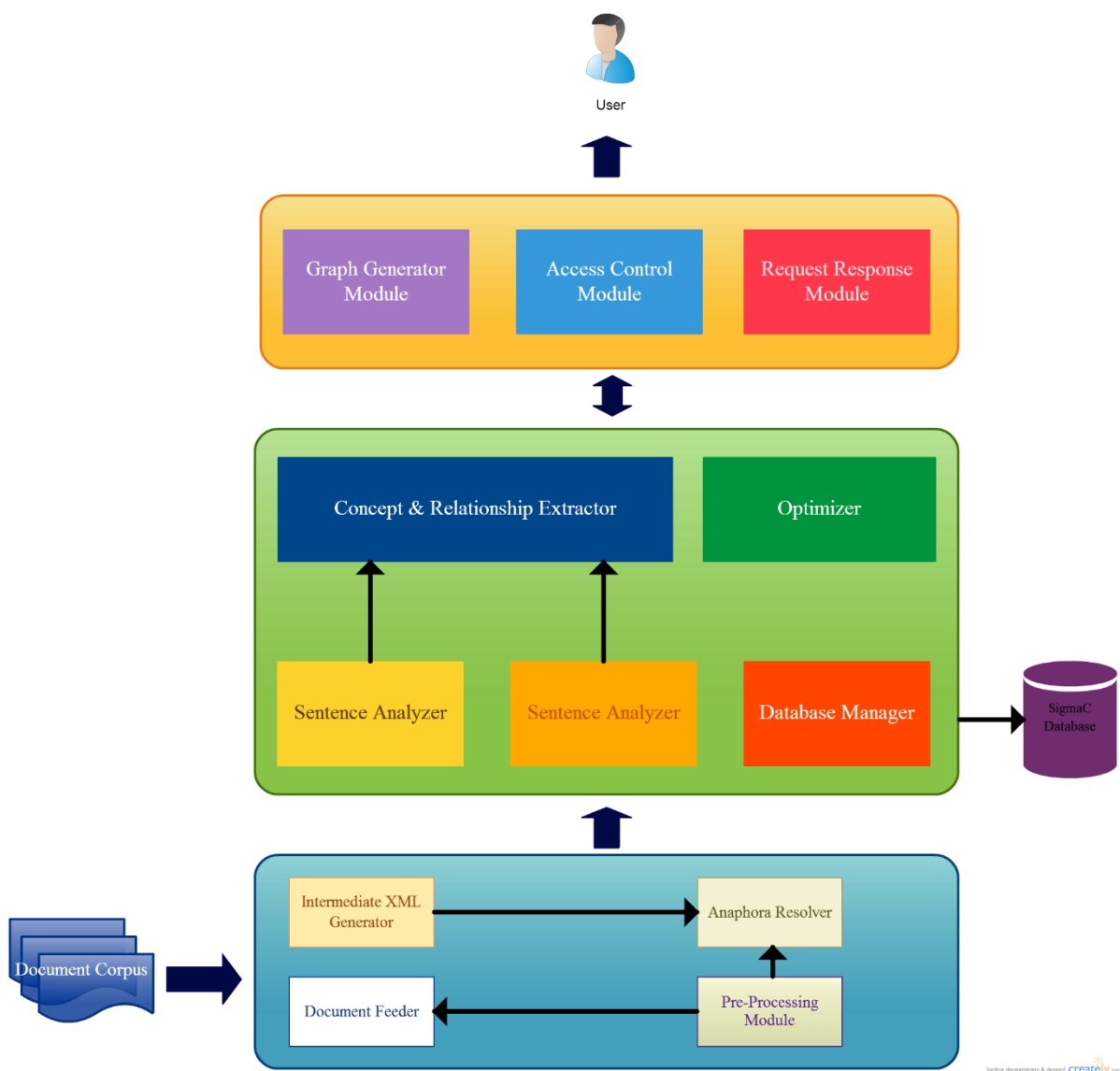


Figure 4. 19 SigmaC Application Architecture

4.3.2.1 Intermediate XML Generator

This module provides general platform for saving files in to one conventional manner to be processed in processing stage. XML files are because it simple and efficiency of using those files.

4.3.2.2 Pre-processing Module

This module is responsible several initial processing in uploading and uploaded documents. This module consist with adding documents to the application, which is implemented, with bulk document upload option with drag and drop option. Identification of the file type uploaded to the system is identified in this module and provide relevant data extraction methods be set to extract text from the document.

4.3.2.3 Anaphora Resolver

In this module, reconcile anaphora resolution engine is used. This reconcile is using two famous criteria, Message understanding conference (MUC) and Automatic Content Extraction (ACE) which are used to define set of noun phrases in co-reference relation.

4.3.2.4 Concept and Relationship Resolver

This module provides lot of core functionalities. This module use preprocessed XML file to do further processing of concept extraction, relation extraction and generating concept map. This module able to handle concepts, relationship and whole document.

Before extracting concepts, it needs to be identified concepts and relationships by matching them against phrase-structured tree of each sentence in document. These patterns are developed using Tregex and Tsurgeon. Tregex is used for identification and Tsurgeon is used for modifying the identified concepts in order to remove unwanted content from it. Then these patterns are stored in XML format for usage.

Another key component on this module is sentence analyzer. In this analyzer, it is fed with single sentence each time and it provides set of concepts and each concepts relationships. This concepts and relationships are extracted from using patterns created previously.

4.3.2.5 Optimizer

Optimizer module is responsible for optimizing concepts and relationships. In this optimizer, concepts and relationships are weighted and filter to get most relevant set of concepts and relationships, which are documented, in the documents of company.

4.3.2.6 DB Manager

This module is totally responsible all the database requests in SigmaC application.

4.3.2.7 Graph Generator Module

Application need to display concepts and it relationship in a concept map. For that, a force direct graph is used. In this force graph, concepts are represented with nodes and relationships are shown using edge in the graph.

4.3.3 Knowledge base about technologies used by company

The map is consist with all the keywords which is related to the company, its domain knowledge and expertise areas. From that map it need to generate a technology graph which is need to compare with candidate in order to check how much this particular person is suitable for this particular company.

4.3.3.1 Extracting Graph

Typically extracting particular part of graph is not trivial problem. We are able to get a part of vertices and edge from graph and able to connect. It will create a sub graph but on this scenario, it is needed to extract a sub graph according to domain knowledge according to the keys in each node.

So it is needed to extract a graph according to particular domain knowledge. As this concept map is generated according to company knowledge base so that it consist with company technologies and other relevant detail about company related things. Because of that, it is able to extract a particular domain specified graph that related to technology used by particular company with referring to concept map.

In order to extract particular related nodes, it is need a predefine dataset to extract for particular domain. Because it is needed to have domain knowledge to extract need to dataset for creating the graph.

4.3.3.2 Approach

First of all predefine gaziter list need to be create regarding to the domain we are going to extract data. In this instance, it is technologies of company and other technically related things for company. Application needs to perform in selecting any IT company. So that this data set need to be an abstractive one in order to adapt that situation.

There are many domains and many kind of technical specification in computer science field. So the data collection is divided in to major two parts. Those are as follows,

- Technology base data extraction
- Concept wise data extraction

4.3.3.2.1 Technology Based Extraction

In this data extraction, we are mostly considering the programming languages, which are existing in today computer science field. So to do that, it is need to collect the all the information about programming language which are immerging and vastly expanded.

Gathering data is done by scraping the Wikipedia. List of programming language page is consist with all of programming languages, which are widely used, and immerging language. Therefore, by scraping it is able to get list of programming language.

4.3.3.2.2 Concept wise Data Extraction

As computer science fields is vastly expanding day by day and lots of changes happing in technology and concept wise. Therefore, that it is little bit hard to track all the changes in computer science field. Therefore, that outline of knowledge, which is represent all the fields in computer science domain get to use.

So for this data extraction, Wikipedia page of Outline of Computer science is scraped and gather the relevant data to create a list of computer science filed related concepts which is abstract for whole fields in computer science subject.

4.3.3.3 Implementation

There are three main key components of implementation. Those are considered as extracting dataset and storing, comparing those dataset with the concept map node values and creating particular graph which is related to the technologies in the company.

In initial stage is the data extraction stage. Therefore, for the first time application is running, the data is scraped from Wikipedia and stored into text files for future usage. In this extraction, only relevant data is only extracted and saved to the file. All the technologies and abstractive concepts computer science domain is extracted and saved.

Then it is needed to extract technology domain graph for the particular company with respect to the concept map, which generated using the company documentations on technical details. As our application is already extracted the dataset for comparison, so it is needed to get the related node which is related to the technical details.

The node consists with a string of detail, which is extracted by using several methods above mentioned and used, in SigmaC application. In each node consists with its relationship, relationship strength, frequency and type of the relationship with the other nodes. However, we are mostly interested in the node value. With getting that node value, it is divided into words and compare each word or whole string is compared with technology and computer science abstractive concepts and extract the node, which has a relationship with that list of words in list. Then this node list values are stored in the application.

Creation of graph is the final stage. We have already extracted relevant node for company technical background, so it is needed to be connected into each other. Therefore, for that we used a simple approach.

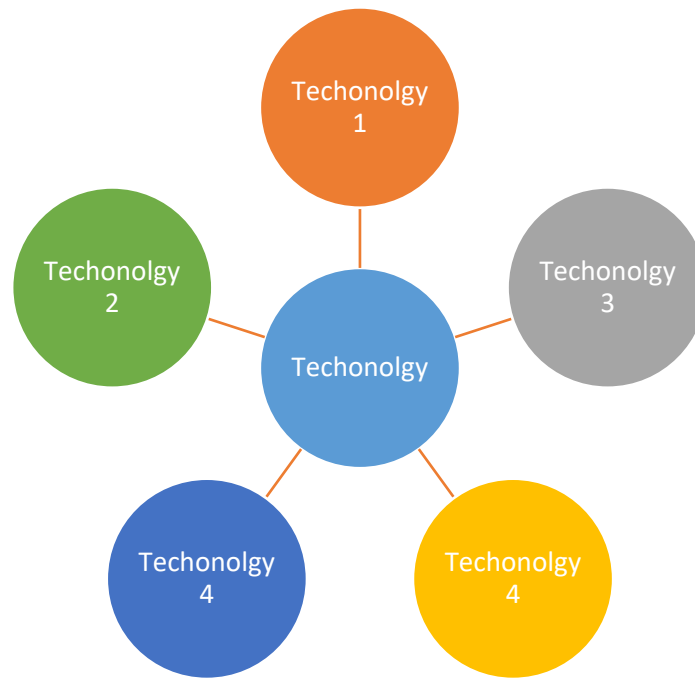


Figure 4. 20 Technology Graph

A new node call technology is created and then all other technical related nodes connect to that particular node. So by having that approach it is helped to create particular graph which is especially for particular company and able to comparison on candidate expertise in computer science filed.

Also this graph is appeared in very first begin in order to show to the users in application which kind of technical stuff used and exists in this particular company. So it is helped any user to get identified what is there company domain areas specified.

4.4 Measuring the Suitability of the Candidate for the Company

As we described in the above section, defining a proper scoring mechanism is a tedious task for this purpose. In our system, we use a technique, which is based on measuring the similarity of between graphs. This is because we generate two concept maps individually for both candidate and the company. Concept map generation for the candidate has been done in an earlier version of our product. In which a previous project called SIGMAC. Our system is an extension of this project and underlying concept map generation for the company is achieved by it. In order to extract only the relevant technology related concepts, we used some trimming mechanisms, which described in earlier section.

Now we have two concept maps, which includes same types of information, and we can then map these two graphs and generate a score for the similarity between the two graphs. When the score generated for the similarity of two graphs increases the suitability of the candidate to the company increases. As an example if candidate A has a score “a” and candidate B has a score “b” being $a > b$ we can say that the candidate A suits for the company than the candidate B. Now let’s have a look at the graph similarity measuring algorithm used in the system and the implementation perspectives of the algorithm.

4.4.1 Graph Similarity Measuring Procedure

Before we go in to detailed description about the graph similarity algorithm, let us take at the process execution of the algorithm. Bellow state chart show the procedure and let us look at it.

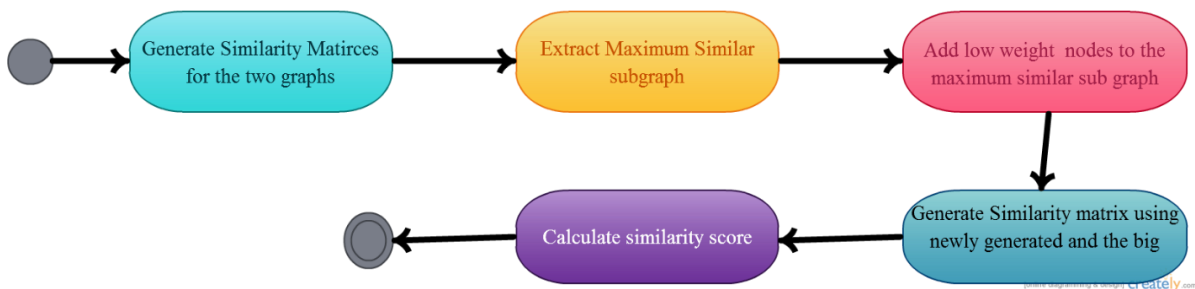


Figure 4. 21 Graph Similarity Measuring Activity Diagram

The above procedure is based on the proposed work on the paper [4]. As shown in Figure 4.19, there are five steps to calculate a similarity score.

1. Generating similarity matrix for two graphs.
2. Extracting maximum similar sub graph using similarity matrix
3. Recreate a big graph by adding low weighted edges to extracted sub graph at step 2
4. Generate a similarity matrix again for newly created graph at step 3 and larger graph
5. Calculate the similarity score using final similarity matrix.

Let us see about these steps in detail one by one.

4.4.1.1 Generating similarity matrix for two graphs

Similarity matrix of graph A and graph B can be represented as $X = [x_{ij}]$ of dimension $|V_A| \times |V_B|$ with the element x_{ij} denoting a similarity of the nodes $i \in V_A$ and $j \in V_B$. To generate this

similarity matrix of two graphs, we used “Neighbor Matching” method. This neighbor matching method is based on the simple concept that most of the existing graph similarity measure algorithm used.

An element in graph G_A and an element in graph G_B are considered similar if their respective neighborhoods within G_A and G_B are similar [5].

According to the above concept, there is a relation between the similarity of two nodes and their neighbors. Based on that, we can build our solution.

This similarity matrix generation process is an iterative process. In each iteration, it update the similarity matrix generated by previous iteration. For the updating task, we used following equation.

$$x_{ij}^{k+1} \leftarrow \frac{s_{in}^{k+1}(i,j) + s_{out}^{k+1}(i,j)}{2}$$

Equation 4.2 Update Similarity Matrix Entry

This equation will calculate the similarity of i^{th} node of graph A and j^{th} node of graph B in $(k+1)^{th}$ iteration. As we see, we need to find $s_{in}^{k+1}(i,j)$ and $s_{out}^{k+1}(i,j)$ i^{th} iteration first. $s_{in}(i,j)$ is the in degree similarity of node i in graph A and j in graph B. $s_{out}(i,j)$ is the in degree similarity of node i in graph A and j in graph B. They can be calculated by following equations.

$$s_{in}^{k+1}(i,j) \leftarrow \frac{1}{m_{in}} \sum_{l=1}^{n_{in}} x_{f_{ij}^{in}(l)}^k g_{ij}^{in}(l)$$

Equation 4.3 Calculating In node Similarity matrix

$$m_{in} = \max(id(i), id(j))$$

Equation 4.4 Max in degree

$$n_{in} = \min(id(i), id(j))$$

Equation 4.5 Minimum in degree

$$s_{out}^{k+1}(i, j) \leftarrow \frac{1}{m_{out}} \sum_{l=1}^{n_{out}} x_{f_{ij}^{out}(l)}^k g_{ij}^{out}(l)$$

Equation 4.6 Calculating out node similarity matrix

$$m_{out} = \max(od(i), od(j))$$

Equation 4.7 max out degree

$$n_{out} = \min(od(i), od(j))$$

Equation 4.8 min out degree

In here $id()$ means in-degree and $od()$ means out-degree of the node. We calculate the similarity of in-degree and out-degree of nodes by taking the summation of the neighbors' similarity in previous iteration. We choose those similarity values according to the enumeration functions. Enumeration functions is a functions that gives the maximum similarity value for the each node in the given node list.

Before start the iterations, we need to initialize the in-degree and out-degree matrices. It is very important because in each iteration, we refer the pervious iteration's similarity matrix and if there is an error with the starting matrix, that error will be propagated. We can initialize the in-degree and out-degree matrices using following equations.

$$s_{in}^1 = \frac{\min(id(i), id(j))}{\max(id(i), id(j))}$$

Equation 4.9 Initializing in node matrix

$$s_{out}^1 = \frac{\min(od(i), od(j))}{\max(od(i), od(j))}$$

Equation 4.10 Initializing out node matrix

Then we can construct initial similarity matrix using these two matrices. After constructing initial similarity matrix, we can iteratively update the similarity matrix until it met the termination condition. We can see that, in each iteration, similarity values are getting converge. Therefore using that property, we can add a termination condition. In here, if the maximum difference between previous iteration and current iteration is smaller than given epsilon value, we terminate the update function. As the epsilon value, we are using 0.0001 to increase the accuracy. Finally, we will have our similarity matrix.

4.4.1.2 Extracting maximum similar sub graph using similarity matrix

In the previous step, we created our similarity matrix. Actually, we can calculate a similarity score using that similarity matrix. However, it is not giving more accurate answer for some graphs. Since we are calculating the similarity of two nodes by looking the similarity of their neighbors, neighbors of the nodes effect heavily to the similarity value. If the nodes have more neighbors and their similarity is very low, similarity score will reduce very badly. Not only the current iteration's value, but also it will propagate the error in each iteration. That means if we are checking similarity between more connected graphs, there is a large probability to reduce the similarity score than expected similarity score. As a solution for that, we extend this algorithm to find the similarity of two graphs more accurately.

For that, we have to extract the maximum similar sub graph of two given graphs. Actually extracting maximum similar sub graph in a NP-Complete problem. However, using our similarity matrix, we could extract maximum similar sub graph very easily. Although we extracted it easily, the problem remains NP-Complete.

After creating similarity matrix as above step, we find maximum similarity value in that matrix. After finding maximum similarity value, we take the two nodes of two graph. In here, maximum similarity value means, those two nodes are the most similar nodes. When we extracting the maximum similar sub graph, this should be our starting point. Someone may argue that, is it right to choose these two nodes. Since we are using similarity matrix, we can assume that, these two nodes will be in our sub graph as they are the most similar nodes in two graphs. By starting these nodes, we are creating a maximum similar sub graph by checking the corresponding maximum similar values and adding most similar nodes to the already constructed part. Likewise, we extract the maximum similar sub graph from the two graphs.

4.4.1.3 Recreate a big graph by adding low weighted edges to extracted sub graph at step 2

After extracting maximum similar sub graph, we have to recreate the bigger graph from the given two graphs. For that, we used the extracted sub graph. We added missing nodes to the sub graph using low weight edges. Edges that were in original graph will have larger weights and other nodes and edges added by us will have lower weights.

4.4.1.4 Generate a similarity matrix again for newly created graph at step 3 and larger graph

After creating new graph, we have to generate a similarity matrix again for newly created graph and bigger graph. In here bigger graph means that the graph with larger number of nodes. Using the step 1, we can re calculate the similarity matrix for these two graphs.

4.4.1.5 Calculate the similarity score using final similarity matrix

Then using that newly created similarity matrix, we can calculate the maximum similarity score. For that, we can use the same enumeration function that used in step 1. This enumeration function will give a maximum similarity score of two graphs. However, this score is not our final score. Because this score has, the similarity values come from the edges that we added in step 3. Therefore, final score will be as follow.

$$Final\ Score = \frac{maximum\ similarity\ score}{number\ of\ edges\ added\ to\ the\ sub\ graph} \times 100\%$$

Equation 4.1 Final Similarity Score

4.4.2 Implementation of the Graph Similarity Measuring Module

4.4.2.1 Generating similarity matrix

Generating similarity matrix is the main task of this algorithm. We have to consider few sections when we generating the similarity matrix.

4.4.2.1.1 Initialize Similarity Matrix

By using following algorithm, we can initialize our similarity matrix.

```

initializeSimilarityMatrix(){
    for( i = 0 to graphA.size){
        for(j = 0 to graphB.size){
            maxDegree = max(graphAInNode(i),graphBInNode(j));
            if(maxDegree != 0){

                inNodeSimilarity[i][j]=min(graphAInNode(i),graphBInNode(j))/maxDegree;

            } else{
                inNodeSimilarity[i][j] = -1;
            }
            maxDegree = max(graphAOutNode(i),graphBOutNode(j));
            if(maxDegree != 0){

                outNodeSimilarity[i][j]=min(graphAOutNode(i),graphBOutNode(j))/maxDegree;

            } else{
                outNodeSimilarity[i][j] = -1;
            }
        }
    }

    for( i = 0 to graphA.size){
        for(j = 0 to graphB.size){
            if(inNodeSimilarity[i][j] == -1){
                similarityMatrix[i][j] = outNodeSimilarity[i][j];
            } else if(outNodeSimilarity[i][j] == -1){
                similarityMatrix[i][j] = inNodeSimilarity[i][j];
            } else{
                similarityMatrix[i][j] = (inNodeSimilarity[i][j] +
outNodeSimilarity[i][j])/2;
            }
        }
    }
}

```

4.4.2.1.2 Enumeration Function

Enumeration function is a function that give the maximum summation of similarity for a given list of nodes.

```
public double outputEnumerationFunction(List<Integer> neighborListMin,
List<Integer> neighborListMax) {
    double similaritySum = 0.0;
    Map<Integer, Double> valueMap = new HashMap<Integer, Double>();

    for (int i = 0; i < neighborListMin.size(); i++) {
        int node = neighborListMin.get(i);
        double max = 0.0;
        int maxIndex = -1;
        for (int j = 0; j < neighborListMax.size(); j++) {
            int key = neighborListMax.get(j);
            if (!valueMap.containsKey(key)) {
                if (max < similarityMatrix[key][node]) {
                    max = similarityMatrix[key][node];
                    maxIndex = key;
                }
            }
        }
        valueMap.put(maxIndex, max);
    }

    for (double value : valueMap.values()) {
        similaritySum += value;
    }
    return similaritySum;
}
```

4.4.2.1.3 Generating Similarity Matrix

As we described above, similarity matrix generation is an iterative process. For that, we need both initialization function and enumeration function. By using these two functions and the

equation mentioned above, we can generate similarity matrix. Moreover, we have to add termination logic to terminate the iterations.

4.4.2.2 Extracting Maximum Similar Sub graph

To extract maximum similar sub graph, we have to identify the maximum similar value from the similarity matrix and two nodes, which give that similarity value. For that, we can easily go through the similarity matrix and identify the maximum value.

After identify the two nodes, we have to construct the sub graph starting from this two nodes. This is a recursive function and algorithm for that function can see follow.

```
extractSubGraph(int indexA, int indexB){
    inNodeMap = in-node lists of indexA and indexB map for each node
    to get maximum value
    outNodeMap = out-node lists of indexA and indexB map for each
    node to get maximum value
    removeDuplicationFromMatrix();
    if(inNodeMap.isEmpty() and outNodeMap.isEmpty()){
        return;
    } else {
        for(int value:inNodeMap.values()){
            constructedGraph[value][indexA] = 2;
        }

        for(int value:outNodeMap.values()){
            constructedGraph[indexA][value] = 2;
        }

        for(int key:inNodeMap.keySet()){
            extractSubGraph(inNodeMap.get(key),key);
        }

        for(int key:outNodeMap.keySet()){
            extractSubGraph(outNodeMap.get(key),key);
        }
    }
}
```

4.4.2.3 Construct larger graph from extracted sub graph

We can easily construct the new graph using following function.

```
public void constructLargerGraph() {
    Integer[][] graphAMatrix = graphA.getGraph();
    for (int i = 0; i < graphSizeA; i++) {
        for (int j = 0; j < graphSizeA; j++) {
            if (graphAMatrix[i][j] == 2 && constructedGraph[i][j] ==
null) {
                constructedGraph[i][j] = 1;
            } else if (graphAMatrix[i][j] == 0 && constructedGraph[i][j]
== null) {
                constructedGraph[i][j] = 0;
            }
        }
    }
}
```

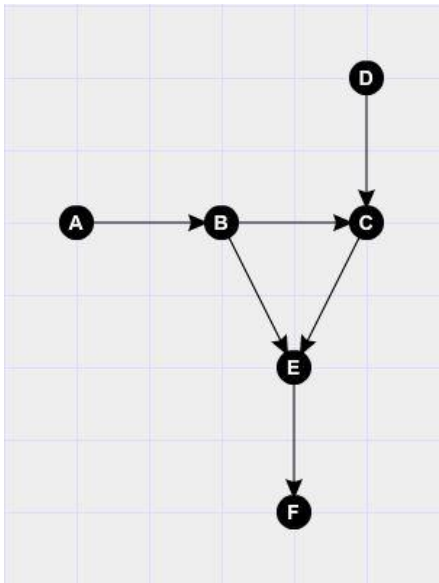
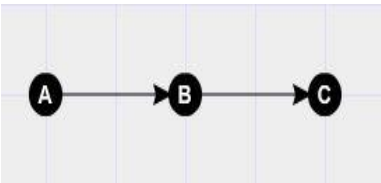
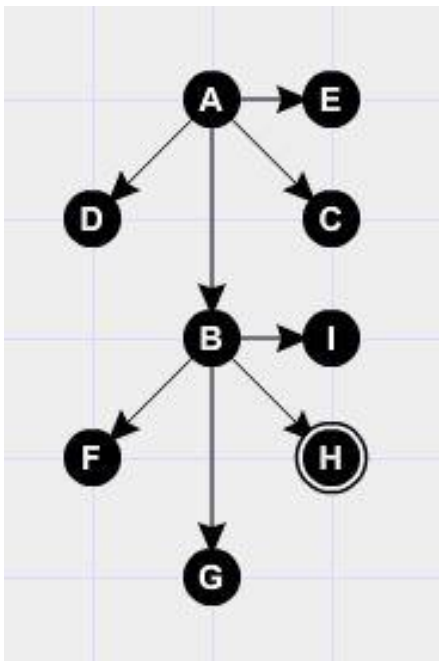
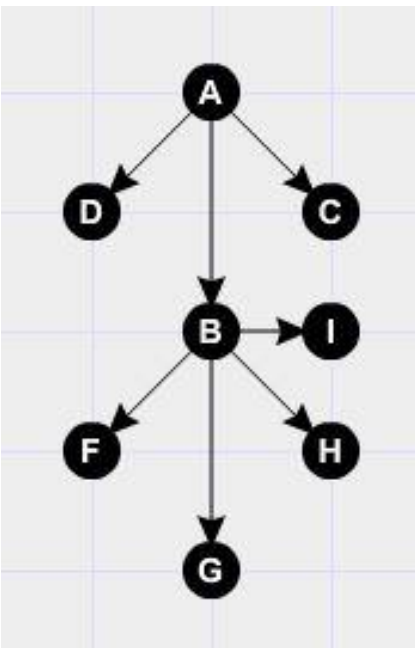
4.4.2.4 Generate new similarity matrix and calculate final similarity score

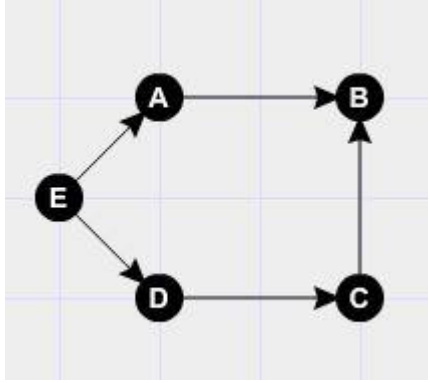
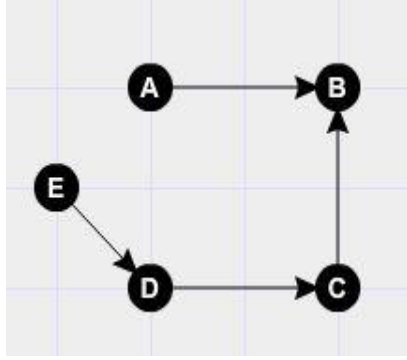
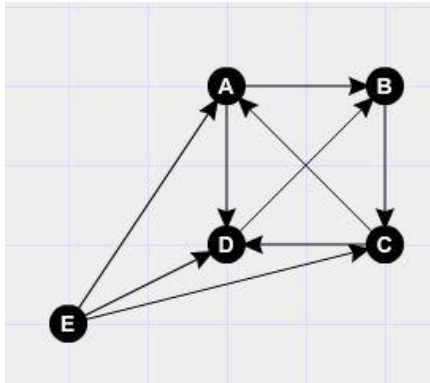
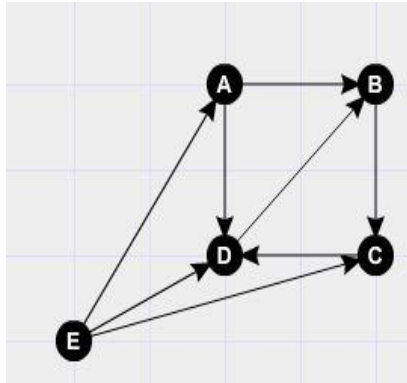
After creating new graph using extracted sub graph, we can generate a similarity matrix again by using the functions described above. After generating the function, using the same enumeration function mentioned above, we can take the maximum similarity summation. Finally, by dividing the maximum similarity summation by number of newly added edges, we can take the final similarity score.

4.4.3 Results and Analysis

In this section, we are going to analyze the results of the Graph Similarity Measuring module. For testing purpose, we used 20 graph sets and following table will display the 4 graph sets and their results. We checked those results against the human ratings. In here, human ratings means that the score given by the human for the graph sets.

Table 4. 15 Graph Similarity Result Table

Graph 1	Graph 2	Algorithm Score	Human Ratings
		18.228%	29%
		92.456%	95%

		78.5%	80%
		61.8%	71%

According to the Table 4.15, score generated by our algorithm is very similar to the human ratings. As described in above sections, when the graph being too connected, our algorithm tends to reduce the accuracy. However, when the graph is less connected, our algorithm score tend to be more accurate.

4.5 Statistics Representation

In this section, we are going to look at how statistics of the candidate is being shown to the user. In our recruitment helper, it is important to show the calculated statistics to the user (Recruiter). Because the objective of this system is to represent the analytics in a manner which let the user to get decisions easily. In our system, mostly used statistics representation is the comparison based statistics representations. For this purpose, we use graphical representations.

4.5.1 Statistics Representation Tools

For the statistics representation, we use the Javascript Library Called HighCharts. In HighCharts charting library there are many statistic representational chart types. For the use of Warana System, we are using *Bar Charts, Spider Web Charts and Scatter Plots*.

4.5.2 Candidate Statistics

4.5.2.1 Technical Proficiencies of the User

In Warana System, user can take a look at the candidate's profile in the View Statistics view. When the user click on the *View* Button for the corresponding user, a Modal appears with a tabbed pane. In that tabbed pane, there is a tab for the Technical Proficiencies. At the information extraction process, for each candidate based on his/her information these technical proficiencies are calculated as a score between 0 and 100. Figure 4.20 shows the graph for a candidate's technical proficiencies.

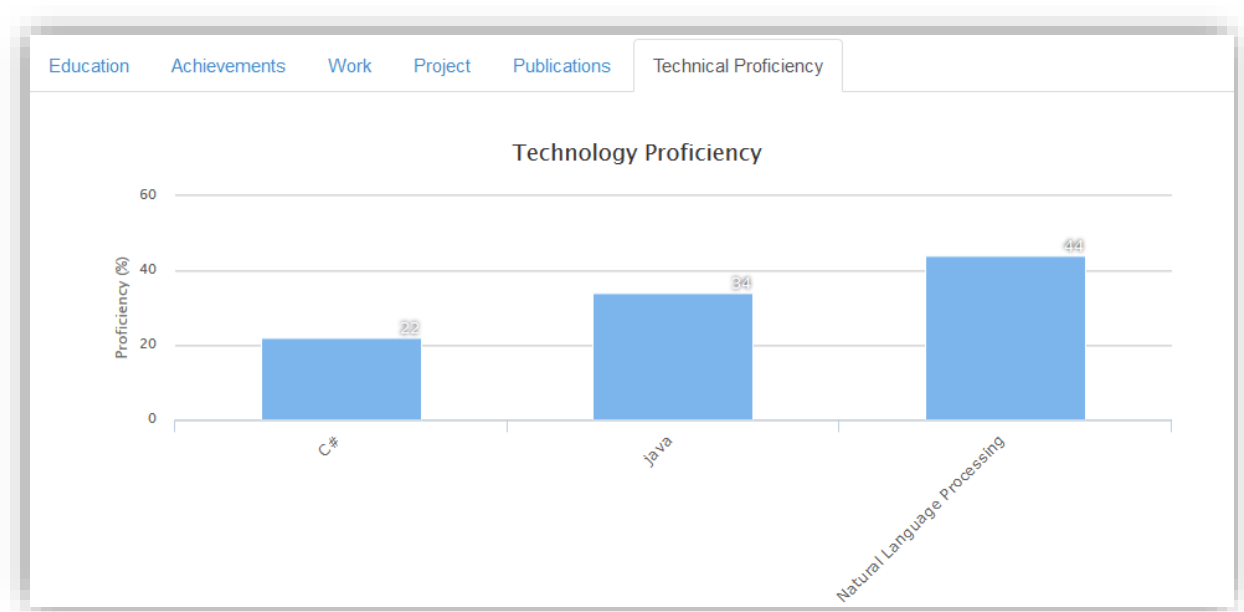
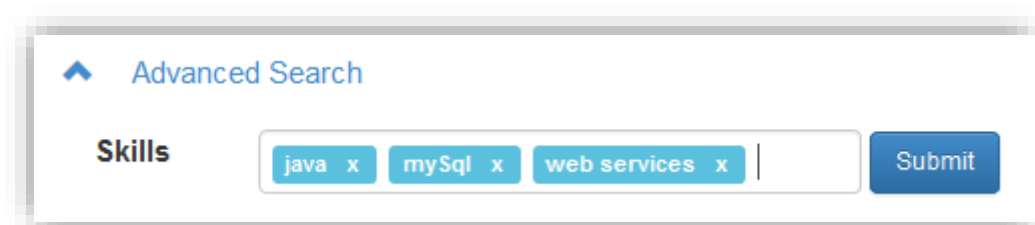


Figure 4. 22 Technical Proficiencies

Main reason for using a bar chart to display the statistical data is its ability to allow the user easy decision making looking at the height of the bars.

4.5.2.2 Comparison of Technical Proficiencies of Several Candidates

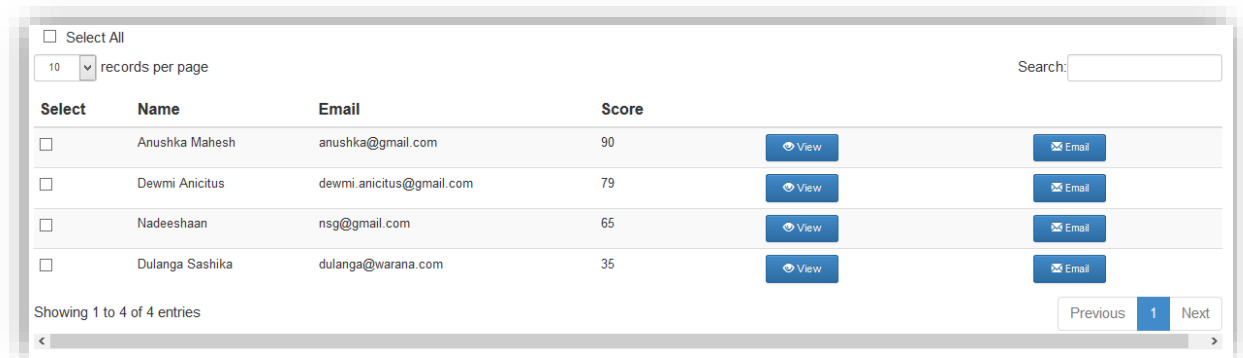
Warana allows the user to search candidates by the name of technologies. As an example if the user wants to search candidates having technical skills on *Java*, *MySQL* and *Web Services* User can use the *Advance Search* option as shown in Figure 4.21.



The image shows a web interface for an 'Advanced Search'. It features a blue upward-pointing arrow icon followed by the text 'Advanced Search'. Below this, there is a section labeled 'Skills'. To the right of the label is a search input field containing three tags: 'java x', 'mySql x', and 'web services x'. To the right of the input field is a blue button labeled 'Submit'.

Figure 4.23 Advanced Search

This search field allows auto completing based on the current technological details based on the data in database. In a formal search result, it shows a list of candidates having those technologies associated with their profiles. Figure 4.22 shows the tabular formed data represented to the user which ordered based on the overall score of the candidate.



The image shows a search results page. At the top, there is a 'Select All' checkbox and a '10 records per page' dropdown. A search bar is on the right. The main content is a table with the following data:

Select	Name	Email	Score		
<input type="checkbox"/>	Anushka Mahesh	anushka@gmail.com	90	View	Email
<input type="checkbox"/>	Dewmi Anicitus	dewmi.anicitus@gmail.com	79	View	Email
<input type="checkbox"/>	Nadeeshaan	nsg@gmail.com	65	View	Email
<input type="checkbox"/>	Dulanga Sashika	dulanga@warana.com	35	View	Email

Below the table, it says 'Showing 1 to 4 of 4 entries'. At the bottom right, there are 'Previous', '1', and 'Next' navigation links.

Figure 4.24 Tabular Data Representation

Why Warana Statistics representation becomes important? Above tabular representation is the formal search query result show to the user most often in any application. When using this kind of application user most of the time wants to compare between several candidates and take decisions. Warana addresses this problem in an interesting manner as Figure 4.23 shows.

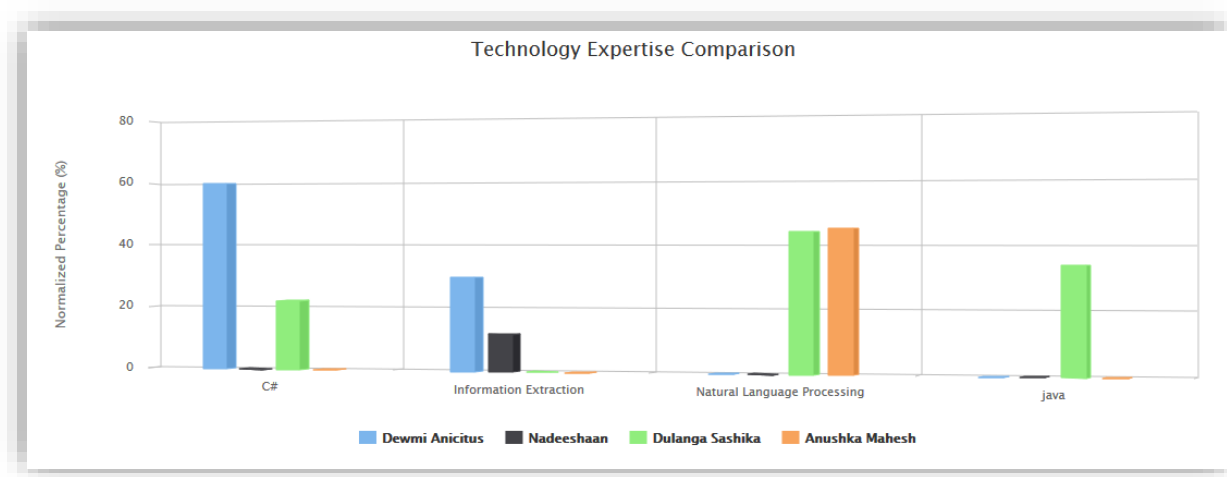


Figure 4.25 Candidate Comparison

Let's take a look at the above *Grouped bar Chart* for the Technical Expertise Comparison. According to the search, results of the advanced search candidates are grouped together in to technological groups. As Figure 4.23 shows, candidates are grouped in to four technology groups (*C#, Information Extraction, Natural Language Processing and Java*). In each of the groups each candidate is shown by a bar with a height proportional to the score calculated for the given technology. Therefore, at once the user take a look at a certain technology group he/she can take a decision which candidate has a higher expertise for the considered technology. This allows the user to choose candidates depending on the technology and evaluate them easily.

Spider web graphs are used to place the candidates' technical proficiency web on top of the company's technical proficiency web as shown in Figure 4.24. This allows the recruiters to easily get a clear understanding by mapping the candidate based on the company's technological aspects.

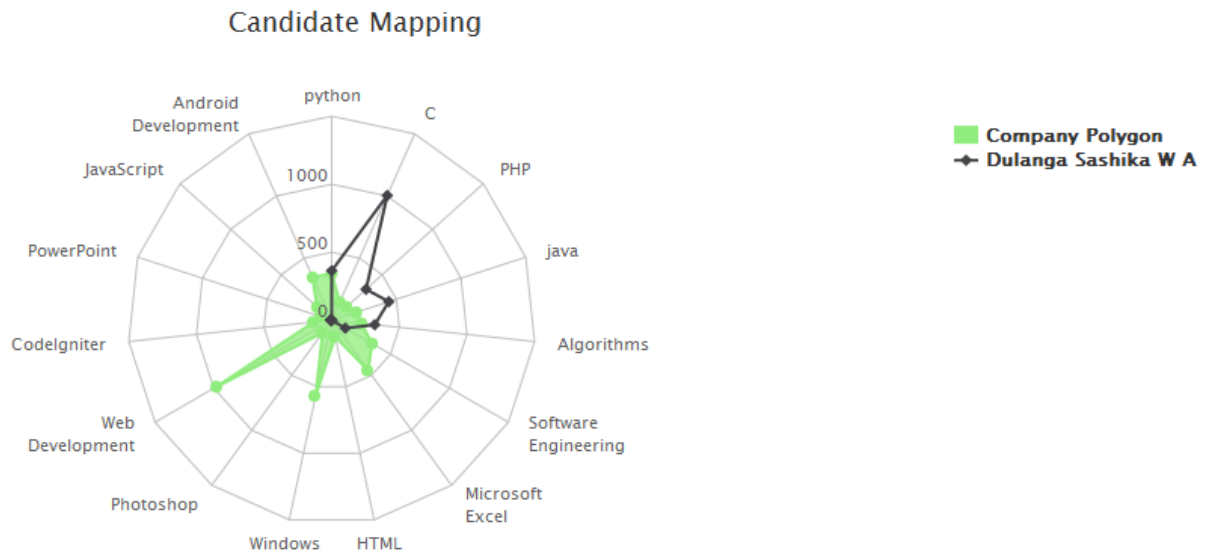


Figure 4.26 Spider Web Graph

One of the most important fact we need to emphasis on these representation mechanisms is the extendability and the understandability of the representations. These representation provide quick overview of the candidate and most importantly the information representation in such a way, can carve the idea thoroughly and provides a clear impression about the candidates. This is what we expect to achieve through the system and these analytical representations allows to enrich the outcome.

5. Discussion

5.1 Assessment on overall solution

As mentioned earlier, the objective of Warana is to provide a solution for automating the process of analyzing and sorting the CVs according to the suitability of the candidate for a given organization. We are happy to mention that, this objective has been realized in the project and we have developed a live website which can anybody test the system. Although the required analytical scores form the application was suitability score for the candidates, we could enrich the application by providing additional analytical details as expressed in the Section 4.5. These additional analytical statistics representation allows the recruiters to evaluate the candidates in multiple perspectives with less effort.

5.1.1 Adaptability of the solution

The system has a component, which generate the concept map for the organization using the documents provided by the organization. Therefore, the system can be adapted by any organization. However, for the organizations in the software industry has a special advantage over other organizations in different disciplines. Because, GitHub is used only by people in the software industry. In addition, online presence of people in other fields is less compared to people in IT field. Therefore, the effectiveness of aggregated profile can be vary according to the field of organization. However, the inline presence of people is increasing rapidly with the advancement of IT facilities. In addition, Warana system has been designed at its architectural level in a manner, which can be customized depending on the customer. In other words, this system can be enhanced by adding different information extraction modules. This aspect has been explained in detail in Section 3.3. In a shorter form, we can add other information extraction modules depending on the customer need. In addition, the data representation in the statistics view have been designed in a manner, which can be add other search fields in the future depending on the required search criteria of the customer.

5.1.2 Business aspect of the system

The current system is designed as a web application, which can be used by one company. Because only the candidate side operations are provided via the web application to the outside. Therefore, there is no way to change the concept map of organization from the front end. It can be changed only from the backend of the system. This model is suitable for a single organization. Therefore, in order to use the system by an organization, first, they have to establish the system with their own webserver and configure for their organization.

Another possible business model is, we can provide both candidate side and organization side operations via the web application. That means organizations can create accounts and upload the documents, which represent the organization and make the concept map. In addition, they can upload the CVs and process in order to get results. In this approach, the system will be hosted in one server and all the client organizations are just users of the system represented by a user account in the system. In this model, scalability is a major issue.

5.1.3 Comparison with existing solutions

There are no any solution, which is similar to what Warana does. We found 2 solutions called DaXtra Parser & DaXtra Search, RezScore. Those are just partial solutions to the problem addressed in Warana. First solution sorts the given CVs according to a predefined set of skills. ResZcore assigns a score to the resume for a given job position using a keyword-based approach. However, Warana provides a complete solution. It considers all the information found about the candidate in web, not just information included in CV. In addition, it learns about the target organizations in concept map generation process. These features are not found in any of the existing solutions. Therefore, to the date, Warana is the best solution for CV analyzing and sorting.

5.2 Project Management

5.2.1 Development process

Project management is essential for the success of project. Even though we didn't concern much about the exact development process we should have used, we used an agile approach. Because, with the academic work and other issues, it is not possible to maintain a rigid process. However, we developed the project incrementally and time-to-time, there were some changes in architecture and technologies we used. In addition, since the development team is small, management was not hard

5.2.2 Project planning

Initially, we developed a plan for the project. However, there were changes in the academic schedule and variations in academic work load time to time. In order to compensate those issues, we had to change the plan continuously. Finally, week by week, we decided what to complete by next week and we completed those parts within the decided time. That plan worked well for us.

Initially our system requirement was calculating a suitability score for a candidate, depending on the candidate's information and the company knowledge base. With the success of our system, we could almost finish these requirements week after the mid evaluation. Having two people involved testing and finalizing the score generation component, other two members

involved in the implementation of another requirement. This was the statistics representation of the candidates. At the end of the project development, cycle all these requirements could finish successfully.

5.2.2.1 Code base management

Warana is implemented using open source frameworks. Therefore, we used Github for version control and the source code is hosted as a public repository in GitHub in following repository (<https://github.com/wadsashika/Warana>). As the coding standard, we used official Java code conventions and Spring MVC naming conventions for file names.

5.2.2.2 Warana Team

Warana team is consist of four developers and two mentors. Since all the developers lived near the university mostly in the development period, we held frequent meetings in order to re-organize the work and update each other. Other than that, online means also used in communication. Since one of the mentors was abroad, meetings with him held using online means.

5.2.2.3 Work distribution

First, we identified the component to be built loosely coupled with one another. Then, each component is assigned to developers. In this phase, we worked independently. After completing that component, we moved into integration phase. In this phase, we worked together. Other than development work, respective developers for their components did writing research papers and reports.

5.3 Project Outcomes

5.3.1 Warana- Recruitment helper for enterprises

Warana recruitment helper for enterprises is the main outcome of this project. Warana website is still in the testing phase and soon it will be open for public use. This website is configured for one particular organization. However, this will be a demonstration and other companies can configure Warana for their interview process when selecting new candidates for their

vacancies. The source code of the Warana is also available in the Github and anybody interested can fork it and contribute for further development.

5.3.2 Research papers

This is the final year project of our BSc. Engineering degree program. As an academic project, other than the objective of the system developed, a main objective is acquiring knowledge and pushing the boundaries of knowledge. Therefore, using the experience and knowledge gain throughout the project, we have written four research papers so far. At the moment one of the research papers have been already published while the other three papers are still pending for publish.

5.3.2.1 Sentence similarity

In Warana system, we wanted to remove redundant data in order to maintain the system properly. Due to this reason, we wanted to develop a module to remove the redundant information. For this purpose, we implemented a sentence similarity measuring module which can be integrated with the system. Based on this similarity measuring algorithm implementation we wrote a paper and published ICTER 2014 conference.

Paper: *U. L. D. N. Gunasinghe, W. A. M. De Silva, N. H. N. D. de Silva, A. S. Perera, W. A. D. Sashika, W. D. T. P. Premasiri, University of Moratuwa, Sri Lanka. "Sentence Similarity Measuring Vector Space Model", International Conference on Advances in ICT for Emerging Regions, Colombo, 2014*

5.3.2.2 Synergistic approach to key term extraction

This research paper is based on the key term extraction algorithm we developed by combining eight existing key term extraction algorithms. The paper is completed and not published yet

5.3.2.3 Graph similarity

This research paper is based on the measuring graph similarity by neighbor matching algorithm that we developed for measure the similarity score of tow concept maps. We developed this algorithm by modifying an existing algorithm to achieve better accuracy. This paper is not completed yet.

5.3.2.4 Aggregated profile generation

This is based on the system we implemented to generate aggregated profiles for candidates. The paper is not completed yet to be published.

6. Conclusion

At present, each organization receives a large number of resumes and they spend a considerable amount of effort and time to select the best among them. Warana addresses problem and try to provide the best solution for this problem. Warana analyze the candidate resumes as well as the candidate's online profiles such as LinkedIn, Blog Posts, Github, etc. Then Warana calculate the candidates' skill proficiencies and generates a map for the candidate. Then the map generated for the candidate is compared with the company concept map to generate the suitability score of the candidate for the company. Rather than this suitability score calculation and sorting the candidates Warana have detailed statistics representation for comparing candidates.

Despite of main solution provided in this project, we have presented the knowledge gained in the development process as research papers. They would be very useful for the researchers who does researches in the similar area. In addition, the new algorithms we developed can be used in any other system and therefore, they would be very useful for software developers all around the world.

7. Future Work

7.1 Integrating new online profiles for aggregate profile generation

In the current system, we have used only LinkedIn, Google Scholar and GitHub in aggregated profile generation process. In future, we plan to add more information sources such as Academia.

7.2 Integrating a Skill Inventory module to the current system

Current system has the ability of integrating a skill inventory module with it. Since the system has candidate skills and these can be used as an inventory of candidate skills. This new module becomes an important component for the organizations in the future recruitment and performance appraisal processes.

7.3 Generic Model for All the Enterprises

At the moment this system has been developed as a proof of concept of the research work and the research domain is the IT Industry. Our next challenge is how to improve and define a generic model for all type of organizations. This generic solution is currently a vast research area and still an open branch of research in the field of computer science.

8. References

- [1] C. Fellbaum, ed., “WordNet: An electronic lexical database”, Language, Speech, and Communication. MIT Press, Cambridge, USA, (1998).
- [2] Lingling Meng¹, Runqing Huang and Junzhong Gu , “A Review of Semantic Similarity Measures in WordNet” The work in the paper is supported by Shanghai Scientific Development Foundation (Grant No. 11530700300).
- [3] Manjula Shenoy.K, Dr.K.C.Shet, Dr. U.Dinesh Acharya, “A New Similarity Measure For Taxonomy Based On Edge Counting”, in International Journal of Web & Semantic Technology (IJWesT) Vol.3, No.4, October 2012
- [4] Alexander Budanitsky, Graeme Hirst, “Evaluating WordNet-based Measures of Lexical Semantic Relatedness”, Association for Computational Linguistics, 2006
- [5] The Stanford NLP (Natural Language Processing) Group. 2015. The Stanford NLP (Natural Language Processing) Group. [ONLINE] Available at: <http://nlp.stanford.edu/software/corenlp.shtml>. [Accessed 10 November 2014].
- [6] Dingjia LIU, Zequan LIU , Qian DONG, “A Dependency Grammar and WordNet Based Sentence Similarity Measure”, Journal of Computational Information Systems, 2012
- [7] Ralph Grishman, “Information extraction techniques and challenges”, Computer Science Department, New York University, New York, USA
- [8] Kun Yu, Gang Guan, Ming Zhou, “Resume Information Extraction with Cascaded Hybrid Model”, Proceedings of the 43rd Annual Meeting of the ACL, pages 499–506, June 2005.
- [9] Tomasz Kaczmarek , Marek Kowalkiewicz , Jakub Piskorski , “Information Extraction from CV”, Witold Abramowicz (ed.), Business Information Systems, Proceedings of BIS 2005, Poznań, Poland
- [10] Hudson, Richard. “An English Word Grammar. Oxford: Basil Blackwell”, 1990
- [11] A Comparative Evaluation of Term Recognition Algorithms. Ziqi Zhang, José Iria, Christopher Brewster and Fabio Ciravegna
- [12] Powers, David M W (1998). "Applications and explanations of Zipf's law". Joint conference on new methods in language processing and computational natural language learning. Association for Computational Linguistics. pp. 151–160.
- [13] Katerina Frantzi, Sophia Ananiadou, Hideki Mima, “Automatic recognition of multi-word terms: the C-value/NC-value method,” International Journal on Digital Libraries. August 2000, Volume 3, Issue 2, pp 115-130.

- [14] L. Zager, G. Verghese, “Graph similarity scoring and matching”, *Applied Mathematics Letters* 21 (2008) 86-94.
- [15] J.M. Kleinberg, “Authoritative sources in a hyperlinked environment”, *J. ACM* 46 (1999) 614–632.
- [16] V. Blondel, A. Gajardo, M. Heymans, P. Senellart, P. Van Dooren, “A measure of similarity between graph vertices”: applications to synonym extraction and web searching, *SIAM Rev.* 46 (4) (2004) 647–666.
- [17] Mladen Nikolic, “Measuring Similarity of Graph Nodes by Neighbor Matching”, Faculty of Mathematics, University of Belgrade
- [18] Daxtra Technologies (2013), *Intelligent Recruitment Solutions*, [Online] Available: <http://daxtra.com/index.php> [Accessed 20 November 2014]
- [19] Adam Dachis (2010), *RezScore Grades Your Resumes and Offers Free Advice*, [Online] Available: <http://lifehacker.com/5719489/rezscore-grades-your-resumes-and-offers-free-advice> [Accessed 20 November 2014]
- [20] Vivian Giang. (2012). *What Recruiters Look At During The 6 Seconds They Spend On Your Resume* [Online]. Available: <http://www.businessinsider.com/heres-what-recruiters-look-at-during-the-6-seconds-they-spend-on-your-resume-2012-4>
- [21] Apache Software Foundation. (2009-2015). *Apache PDFBox - A Java PDF Library* [Online]. Available: <https://pdfbox.apache.org/index.html> [Accessed 10 November 2014]
- [22] L. Kozakov, Y. Park, T.H. Fin, Y. Drissi, Y.N Dogonata and T. Kofino, “Glossary extraction and knowledge in large organisations via semantic web technologies”, 6th International Semantic Web Conference, the 2nd Asian Semantic Web Conference. 2004
- [23] K. Church, and W. Gale, “Inverse Document Frequency (IDF): A Measure of Deviation from Poisson,” *Proceedings of the Third Workshop on Very Large Corpora*, 1995, pp. 121-130. Ps
- [24] F. Sclano, and P. Velardi, “Termextractor: a web ap-plication to learn the shared terminology of emergent web communities,” In *Proceedings of the 3rd International Conference on Interoperability for Enterprise Software andApplications*, 2007
- [25] K. Ahmad, L. Gillam, and L. Tostevin, “Weirdness indexing for logical document extrapolation and retrieval (wilder)”, in the *Eighth Text Retrieval Conference (TREC-8)*, 1999.