RINX: Information Extraction, Search and Insights from Resumes

Rajiv Srivastava
Tata Research Development and Design Centre,
54 B Hadapsar Industrial Estate, Pune 411013
+91-20-66086319
Rajiv.Srivastava@tcs.com

Girish Keshav Palshikar
Tata Research Development and Design Centre,
54 B Hadapsar Industrial Estate, Pune 411013
+91-20-66086333
Gk.palshikar@tcs.com

Swapnil Hingmire
Tata Research Development and Design Centre,
54 B Hadapsar Industrial Estate, Pune 411013
+91-20-66086333
Swapnil.hingmire@tcs.com

ABSTRACT

A *resume* is a vital source of information about the education, skills, experience and expertise of a person. A repository of resumes (both candidates and employees) is used for several critical tasks: selecting candidates best matched for a position, forming best teams, understanding technology and experience profile of the organization and gathering market intelligence. In this paper, we describe an information extraction system called RINX which extracts various types of information elements from resumes using deep syntactic, linguistic and domain knowledge. We also describe a real-life application of RINX in a large IT organization.

1. INTRODUCTION

A *resume* (also called *CV* or *bio-data*) is a vital source of information about the education, skills, experience and expertise of a person. A resume contains a summary (or profile) of the entire work experience of a person. Typical information elements in a resume are: personal details (e.g. name, current address, gender, date of birth, marital status, phone number, e-mail etc.), career profile (e.g. period, organization, designation), educational qualifications (degree, branch, year of passing, college, score etc.), project worked on (e.g. duration, role, services delivered, technologies used etc.). A repository of resumes of employees in an organization or candidates who have applied to an organization form important reservoirs of information for an organization and can be used for various purposes:

- Search and short-list the right candidates having the best match with the skills and experience requirements for specific positions;
- Use information extracted from employee resumes to populate the databases in the internal Human Resource Management Systems; e.g., updating employee skills along-with proficiency levels;
- Search and select the right person for a specific role for a new or existing project or task;
- Identify trends and profiles of the "job market"; e.g., "people having more than 10 years experience in database administration is increasing";
- Identify trends and profiles for technologies and tools; e.g., "availability of persons having SAP and Oracle skills is decreasing sharply";
- Discover market intelligence (trends and profiles) about competitor organizations; e.g., "competitor X has increasing number of projects in Europe related to finance or banking";
- Discover market intelligence (trends and profiles and other insights) about clients; e.g., "manufacturing companies in Japan and Far East have increasing number of projects in Java and Oracle platforms".

It is clear from these use-case scenarios that making an effective use of resumes in a repository is critical for an organization. But several factors tend to inhibit this. First, the number of employee resumes in large enterprises may range from few thousand to few hundred thousand. Number of resumes from candidates who have applied to a large organization may be over a million. In addition, the resumes are typically free-form or at best semi-structured English documents, thus making it difficult to manually locate and extract information and to compare resumes. HR executives, managers and project leaders need to manually read resumes and extract different types of information from them – a time-consuming, error-prone and subjective task, particularly when dealing with thousands of resumes.

One significant way to improve the operational efficiency of people who need to use resumes is to automatically extract all the relevant information elements from each resume in the given set. In this paper we describe the design and implementation of a tool called RINX, built specifically for extracting different types of information elements (which we call as *entities*) from the given set of resumes. RINX stores the information extracted from resumes in a structured repository such as relational database tables or XML files. This functionality in RINX is based on the classical information extraction (IE) technology, which consists of hand-crafted patterns and gazettes (or lists). The speed and accuracy of extraction, along with easy ways to upgrade and deploy this extraction knowledge, were the critical design criteria for this functionality in RINX. RINX also has a search facility to match the given position profile with the resumes in a given repository and to automatically create a (ranked) short-list of most suitable candidates for the position, along with proper justifications. RINX includes specially designed business analytics functionality to automatically create various types of intelligence from the resumes in a repository. Finally, we describe a real-life case-study where RINX was used. For simplicity, we assume that the resumes deal with persons in the IT industry. The key contribution of this paper is the explicit description of linguistic (lexical, syntactic and semantic) knowledge as well as domain-knowledge required for extracting entities from resumes.

The tasks handled by RINX were quite difficult due to the various challenges related to large variety in resume formats, spelling mistakes, grammatically incorrect sentences, missing or incorrect punctuation, missing or incorrect sentence boundaries, fragmented sentences, incorrect capitalization, variations in style of writing and contents.

- Across multiple resumes, there is absence of any format or arrangement or sequence among various sections like project details, personal details, career profile etc.
- The resume text is often fragmentary and is not suitable for parsing to use tools like named entity taggers.
- There is no fixed list or dictionary of all known roles, organizations and services.
- There are multiple occurrences of role names in a project description, making it difficult to select the appropriate one.
- Date or duration formats vary across resumes.
- The technology names are not standardized; ways to refer to the same technology vary across multiple resumes.

This paper is organized as follows. Section 2 describes the related work and tools. Section 3 describes the architecture, design and internal algorithms in RINX. Section 4 discusses the case-study in detail. Section 5 discusses conclusions and further work.

2. RELATED WORK

RINX functionality described in this paper is related to the task of Named Entity Recognition (NER). NER techniques can be roughly divided into 3 types: (i) dictionary and linguistics based techniques, (ii) unsupervised statistical techniques and (iii) supervised machine learning (ML) techniques. RINX NER is based on dictionary and linguistics based techniques.

Supervised ML techniques attract the most attention of NER researchers because of their advantages such as high performance and domain neutrality (ease of adapting to different domains). Major supervised ML techniques that have been used for NER are: support vector machines [9], [11], conditional random fields [17], [15] (see [12] for Hindi NER), hidden Markov models [3], [4], [10], maximum entropy [13], [20]. Much research in domain-specific NER has focused on the biology and medicine domains; see [7] for an evaluation of such a system.

[2] describes a method for extracting parts of objects from wholes (e.g., wings from airplane) a large corpus using hand-crafted patterns. The FASTUS system [1] uses a series of hand-crafted finite-state transducers to perform the NER task; see also [21]. See [19] and [8] for other rule and dictionary based approaches. [Negri and Magnini 2003] use WordNet predicates multilingual NER; we use WordNet differently. A WordNet predicate returns TRUE if a given word could be used for a given concept and FALSE otherwise. For example, a WordNet predicate *loc* can be defined for the concept of location in terms of a set of synsets (e.g., road#1, ground#1 etc). Then given a word *capital*, the predicate would return TRUE because there is at least one sense of *capital* which matches with the concept of location.

The unsupervised learning approaches reduce the need for a large correctly tagged corpus. [5] propose DL-CoTrain algorithm, which starts with a few simple seed rules (decision lists), identifies the entities in the corpus using these rules, exploits redundancy in the text (two features hint at the same entity) to identify new rules and then continues the cycle. It also proposes CoBoost, a boosting based unsupervised algorithm for NER. [6] presents a system called KNOWITALL, which implements an unsupervised domain-independent, bootstrapping approach to generate large facts of a specified entity (such as City or Film) from the Web. It starts with a few seed patterns – e.g., *NP such as NPList* – and uses them to deduce from the text `cities such as London and Beijing` that `London` and `Beijing` are cities. It contains (i) an algorithm to identify new extraction rules and (ii) a classifier based on point-wise mutual information for validating the extracted entities. [16] propose an approach where they first use an unsupervised approach similar to [5] and [6] to generate a gazette of instances of the entities (e.g., cities) and then use some effective heuristics to identify new instances of the entity.

We have compared IE functionality of RINX with other tools such as Daxtra and Resume Mirror using the marketing collateral available from their web-sites. The strength of RINX is in its deeper analysis and extraction of information from project descriptions.

3. MOTIVATING APPLICATION

3.1 System Description

We now describe the design of the information extraction (IE) functionality in RINX. We focus on extraction of some important types of entities in the resume, which are essential to determine the competency and proficiency level of person: service line, role. As discussed earlier, a major requirement is that the IE should be robust in the presence of noisy, unstructured, fragmentary and sectioned text. RINX consists of a number of patterns to recognize each of these entities, which we describe later (Figure 1).

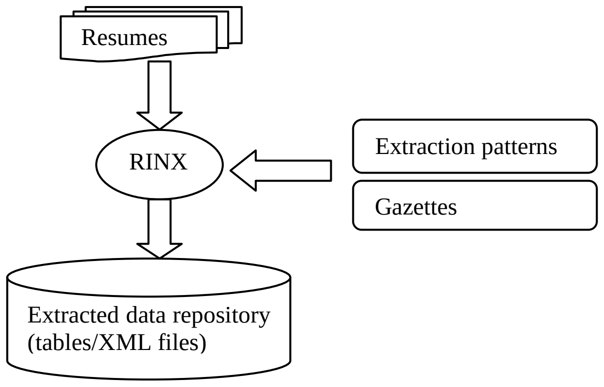


Figure 1. Information extraction functionality of RINX.

3.2 Extraction of ‘Service Line’

A ‘Service Line’ is a domain-specific (often technical) task or activity carried out by humans, with or without using any tools. A ‘Service Line’ is typically mentioned as part of the project or job description which represent the actual work done or the service provided by the resume author. A ‘Service Line’ is typically indicated by a noun phrase (NP), such as *user interface design*. Figure 2 shows a few examples of the ways in which service lines are mentioned in resumes.

```
Main responsibilities include Development,
testing, implementation and Design of the
modules offshore, developed at client
interaction, handling client escalations etc.
Development, supervision and migration of maps
as per the specification.
Execution of Unit Test Cases and Functional
Test Cases.
```

Figure 2. Examples of ‘Service Line’ values (underlined).

‘Service Line’ values are usually domain-dependent; e.g., ‘Service Line’ in IT and finance domains are different. For higher flexibility, we want to minimize the use of gazettes (i.e., lists) containing known values for ‘Service Line’. Moreover, we want to standardize different ‘Service Line’ values, so as to facilitate retrieval/comparison in later stages (e.g., when matching candidates to specific job requirements). Lastly, we aim for at least 90% overall accuracy for extracting ‘Service Line’ values. All this makes extraction of ‘Service Line’ quite challenging.

Algorithm Service_Line specifies the linguistic (lexical, syntactic and semantic) knowledge as well as domain knowledge (for IT domain) required for extracting ‘Service Line’ from a resume.

```
algorithm Service_Line
input   set of $N$ sentences $\{S_1, S_2, \ldots, S_N\}$ from given resume
input   $k$ // max no. of word senses to consider (default $k$ = 2)
input   $m$ // max no. of ‘Service Line’ values per project
            // description (default $m$ = 7)
input   $L_1$; // list of prohibited nouns: work, scope etc.
input   $L_2$; // list of known ‘Service Line’ values: walk-through,
            // documentation, review etc.
output  $L$ // set of service lines selected for given resume
for each sentence $S_i$ and its parse tree $T_i$ do
    for each lowest-level NP (say $X$) in the sentence $S_i$ do
        if $X$ occurs under a VP node containing a copula verb then
            continue;
        if $X$ occurs under SINV node then continue;
        if $X$ contains comma or ‘and’ then // a list of nouns
            treat each noun in $X$ as a separate NP // e.g., if $X$ = “design
            // and development” then treat “design” and
            // “development” as two separate NPs
        if $X$ does not contain at least one noun (NN, NNS, NNP etc.)
            then continue;
        if $X$ fails any one of the tests specified in function check_np
            then continue;
        $L$=∅; // $L$ = list of ‘Service Line’ values found in sentence $S_i$
        for each noun $Y$ in $X$ do // parent node of $Y$ in parse tree $T_i$
                                    // has label NP, NNS, NNP etc.
            if $Y \in L_1$ then continue; // prohibited noun
            if $Y'$ is similar to a value in $L_2$ then // known value
                Add $Y'$ to $L$; continue; end if
            if $Y \in$ WordNet (e.g., budgeting) or $Y$ has a related noun
            then
                obtain root $Y'$ of $Y$ // testing→test, budgeting→budget
                Let $\mathbf{H} = \{H_1, H_2, \ldots, H_k\}$ denote the set of top $k$
                hypernym trees for $Y'$ obtained using WordNet;
                // use first k senses
                if no $H_i$ in $\mathbf{H}$ contains one of {“speech act”, group
                action”, “action”, “change”, “activity” } followed by
                “human activity” then continue; //  not a human action
                Remove articles and prepositions from $X$;
                Add $X$ to $L$; // $X$ is a possible value for ‘Service Line’
                break; // finished checking $X$; go to next NP
            end if
        end for
    end for
end for
Remove duplicate (or very similar) entries in $L$, if any;
// Keep at most $m$ NPs in $L$;
Select NPs which contain nouns from the list of known ‘Service
Line’ values; if the number of such NPs is less than m then keep
other NPs in order of appearance in the project description
return($L$)
```

Function check_np is used to filter out NPs which cannot be possible values for 'Service Line' entity.

```
Boolean check_np($X$) // check if given NP $X$ is acceptable
// e.g., 'Oracle Retail 12 Applications' or 'Tally 4DOT5'
if $X$ contains a number then return(0);
// e.g., 'Raymark Merchandising Systems' is ruled out because
// Raymark is not in WordNet.
if $X$ contains a word not present in WordNet then return(0);
if $X$ contains an acronym then return(0); // GIS, I.B.M.,
// 'GE Consumer Finance' etc.
if $X$ contains a cue word for an organization then return(0);
// e.g., systems, company, incorporated, inc, bank.
if $X$ contains a word having underscore then return(0);
// e.g., ‘SQL_Server’
// Following strong conditions can be turned off by user
if $X$ contains more than 2 nouns then return(0);
// e.g., 'Sales Order Management System'.
if $X$ contains ALL words with first letter capitalized
    then return(0); // e.g., 'Warehouse Management System'.
return(1); // OK
```

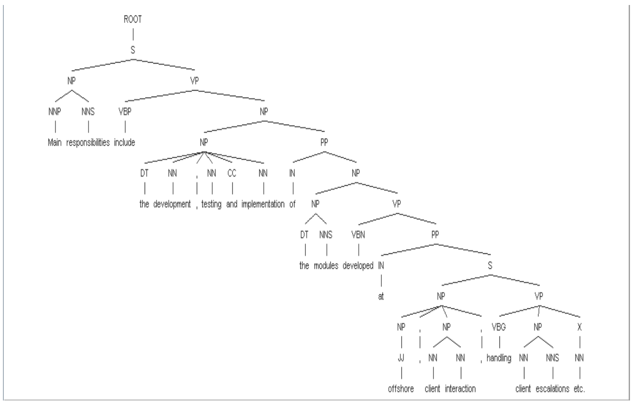

ROOT
S
NP
VP
NNP NNS VBP NP
Main responsibilities include
NP PP
DT NN , NN CC NN IN NP
the development , testing and implementation of
NP VP
DT NNS VBN PP
the modules developed IN S
at
NP VP
NP , NP , VBG NP X
JJ , NN NN , handling NN NNS NN
offshore client interaction client escalations etc.

Figure 3. Parse tree for an example sentence.

Figure 3 shows the parse tree for the first sentence in Figure 2. The following are the lowest-level NPs in this sentence: *main responsibilities*, *the development*, *testing and implementation*, *the modules*, *client interaction*, *client escalations*. The NP *offshore* does not include a noun and so it is eliminated. The NP *the modules* is eliminated (i.e., it is not a possible value for 'Service Line' entity) because none of the hypernym trees for (any sense of) the noun *module* in this NP contains *human activity* (Figure 4). The NP *client interactions* is selected as a possible value because one hypernym tree for *interactions* contains *human activities* (Figure 5). Note that no hypernym tree for *client* contains *human activity*. Similarly, the hypernym trees for *support*, *production*, *development* contain *human activity*. Hence NPs containing *support* would be considered as candidates for 'Service Line'; e.g., *customer support*. The algorithm Service_Line contains several tests to eliminate some NPs as possible values for 'Service Line'. 'Service Line' may sometimes be mentioned as a verb (*designed ….*), rather than as a noun. To obtain 'Service Line' from values specified as verbs, we use WordNet to obtain the nouns related to the given verb; e.g., we can get the noun *inclusion* related to the verb *included* using WordNet.

module
=> component, constituent, element
=> part, portion
=> object, physical object
=> physical entity
=> entity

Figure 4. Hypernym tree for sense 4 of noun *module*.

interaction
=> action
=> act, deed, human action, human activity
=> event
=> psychological feature
=> abstraction, abstract entity
=> entity

Figure 5. Hypernym tree for sense 1 of noun *interaction*.

3.3 Extraction of 'Role'

Considering that one goal for IE from resumes is in understanding the employee associate profile, it is important to analyze the *role* or *position* in which an associate has worked in various projects. For example, if a project in a resume mentions a technology name 'Java', then before assigning 'Java' competency to that person, it is important to identify the role played by the person in the project. If the role is that of 'tester' then the competency 'Java' is not applicable. But if the role is that of 'developer' then the competency 'Java' becomes applicable. The role of an associate needs to be identified from the 'Project Details' section for various projects in the resume. Figure 6 shows some example sentences from actual resumes where a 'Role' is mentioned.

```
My role as a Business Intelligence Expert
is responsible for Managing Business
Intelligence Architecture.
Role         : Software tester
Designation  : Test Associate
ROLE: Worked as the Team member in this
project
Role & Responsibility: Lead Analyst.
Current Employer (Designation): headstrong
[software engineer]
Position: Senior Team Member
Position: Software Engineer (GAP Analysis)
```

Figure 6. Example sentences indicating 'Role'.

We extract 'Role' ('Designation', 'Position' etc.) occurrences only from project (or work) descriptions. We aim for at least 90% overall accuracy for extracting 'Role' values. Algorithm Role described below extracts 'Role' (and similar entities like 'Designation', 'Position') is explained below. Input to the algorithm is the sentences from a project description.

```
algorithm Role
input   set of $N$ sentences $\{S_1, S_2, \ldots, S_N\}$
input   $k$ // max number of senses to consider (default $k = 3$)
input   $m$ // max no. of 'Role' values per project
            // description (default $m = 3$)
input   $R$ // list of known 'Role' values
output  $L$ // set of matched and extracted 'Role' entities
for each sentence $S_i$ and its parse tree $T_i$ do
  $L = \emptyset$; // $L$ = list of gender values found in this sentence
  if $S_i$ starts with cue phrase/word in {role, designation, position} then
    Z = {immediately next NP $X$}
  else if $S_i$ contains cue phrase/word in {role, designation, position} then
    Z = {all lowest-levels NPs in S$i$}
  else
    if $S_i$ contains verb in (work | involve | perform)
    followed by (in | as) followed by a NP then
      add matches to L; end if
    continue;
  end if
  for each element $X \in$ Z do
      if $X$ contains a known role in $R$ then add $X$ to $L$; continue;
      else
        Find headword $Y'$ of $X$;
        Let H = $\{H_1, H_2, \ldots, H_k\}$ denote the set of top $k$ hypernym trees
        for $Y'$ obtained using WordNet;  // use first $k$ senses
        if no $H_i$ in H contains one of {creator, expert, specialist,
        specializer, planner, person, individual} then continue;
        else
          Remove articles and prepositions from $X$;
          Add $X$ to $L$; // $X$ is a possible value for 'Role'
          break;
        end if
      end if
    end for
  end if
Remove duplicates (or similar) entries in $L$, if any;
return($L$);
```

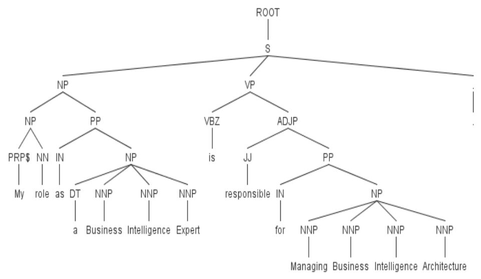


Figure 7. Parse tree for the first sentence in Figure 6.

Figure 7 shows the parse tree for the first sentence in Figure 6. The lowest level NPs in this sentence are: *My Role*, *a Business Intelligence Expert*, *Managing Business Intelligence Architecture*. The corresponding headwords for these NPs are: *Role*, *Expert*, *Architecture*. The hypernym trees for all these three words are attached below. The NPs *My Role* and *Managing Business Intelligence Architecture* are eliminated since none of the hypernym tree for nouns *Role* and *Architecture* contains any word from the given list {*creator*, *expert*, *specialist*, *specializer*, *planner*, *person*, *individual*} (Figure 8). The NP *a Business Intelligence Expert* is selected as a 'Role' value because it contains the role indicating noun *expert* from the list.

architecture
=> building, edifice
=> structure, construction
=> artifact, artefact
=> whole, unit
=> object, physical object
=> physical entity
=> entity

Figure 8. Hypernym tree for the noun *architecture*.

Some nouns such as *provider*, *controller* may appear in the text not as a 'Role' but as a part of software or technology. Since these satisfy the conditions set above, these are marked as 'Role' and hence add to the number of false positives. We maintain a list of such false 'Role' indicating nouns to exclude NPs which contain them.

As another post-processing step, we map each extracted 'Role' value to a standardized 'Role Category' value. For example, 'Role' values like *developer*, *senior programmer analyst*, *technical associate*, *software development engineer*, *programmer*, *and coder* are all mapped to the 'Role Category' value *Developer*. This mapping R_to_RC of 'Role' values to 'Role Category' values is user-defined.

3.4 Extraction of Other Entities

RINX extracts a large number of entities from resumes: name, address, phone number, email address, gender, date of birth, career profile (e.g. period, organization, designation), educational qualifications (degree, branch, year of passing, college, score etc.), project worked on (e.g. duration, role, services performed for a project, technologies used etc.). We describe a few of these entities now.

Period of a Project

The date range, if mentioned in a project description, usually indicates the period of that project. It always consists of two dates. The individual date is mentioned in various formats. We designed several complex patterns using regular expressions to convert a date in a commonly occurring format to a standard date format. Some of the sample examples are as follows:

- `Jan 06 – Till date`
- `August 04 – Dec 05`
- `Oct 2001-Dec 2002`

Technology and Tools

Within a given business domain, a technology refers to an organized body of formal knowledge and techniques that help the organization in providing better service or in producing better products. A technology is often implemented (and made useful in practice) as a machine, device, or a computer program. In IT domain, a technology is often indicated in the form of software tools (e.g., the database technology is indicated by tools such as Oracle, Sybase and SQL Server). The names of tools mentioned in project descriptions in a resume are considered for deriving competencies, as these are the technologies used by the associate in actual work. Some examples of the technologies from IT domain are databases, graphics, programming languages, compilers, image processing, data-mining and model-driven development. Following list gives sample statements from resumes, which contain mentions of technology and tools.

- `Solution Environment: WINDOWS-2000/XP`
- `Tools:      Eclipse 3.0, MS Visual Source Safe 6.0`
- `Support for Queue implementation in Message Driven Beans in MasterCraft 6.5.2 and 7.0`
- `Support for Weblogic 9.0 appserver in MasterCraft 6.5.2`

RINX has complex patterns (as well as extensive gazettes of known tool names) to recognize technology and tools.

3.5 Experimental results

For verification a set consisting of one hundred and sixty free form resumes are taken belonging to candidates from Information Technology domain. The extraction of entities like data of birth, e-mail, project dates, service line and role were extracted and various measures like precision, recall and F-measure are provided in the following table.

Entity	Recall	Precision	F-measure
Date of birth	0.99	0.95	0.97
e-mail	0.98	0.93	0.95
Phone	0.96	0.95	0.95
Project date range	0.92	0.9	0.91
Project Role	0.95	0.7	0.8
Project Service line	0.9	0.83	0.86
Job duration	0.74	0.94	0.83
Employer	0.94	0.91	0.93
Degree Name	0.86	0.9	0.88
Degree Marks	0.71	0.93	0.81

Degree Specialization	0.86	0.9	0.86
Degree University	0.77	0.89	0.82
Year of degree	0.74	0.94	0.83

The role accuracy is further improved by deriving it from the extracted service lines, as majority of service lines grouped together imply a role. The same is elaborated in the detailed case study presented in section 4.

3.6 Applications of the tool

In the last year the tool was used on the resume extraction from multiple domains like Information Technology, Business Process Outsourcing and Electronic Consumer Goods domain across India and US geographies. The patterns were adapted quickly for the various entities and satisfactory results were achieved in short time-frame of two to three weeks.

The issues faced in adapting patterns were due to the various reasons. The main differences found are in structure of resume where-in the work description is organized based on the complete job or projects, the domain specificity of the technical skills, role names and date patterns. The satisfactory results for few extraction applications and pilots for different domains and geographies have yielded satisfactory results. The patterns are being continuously tested and improved for deploying in career portal.

4. CASE-STUDY

RINX was used in a large IT organization to automatically extract technology skills for each employee and automatically assign a competency level for each skill. The current HR process expects employees to manually update the skills and proficiency levels in the HRMS databases, which was not being done by most employees. This poor compliance leaves the HRMS with an incomplete picture of the technical and experience profiles of individual employees as well as the organization as a whole. On the other hand, most employees regularly update their resumes. Thus resumes represented a more up-to-date source of information than the HRMS databases. Hence the main objective of this case-study was to use RINX to automatically extract the technical competencies (and estimate their proficiency levels based on the total time they spent in projects having that particular technology) from 65000 employee resumes, mainly from project descriptions. This case-study helped the HR managers and technical leaders to understand the current “technical and experience profile” of its work-force, which then lead to plan recruitment, create best project teams, plan training programs to augment skills etc. The data extracted by RINX from resumes also enabled intelligent and automatic search of project requirements with up-to-date employee profiles, thereby leading to improved utilization of the organization-wide talent pool.

The essential idea behind this task was to use RINX to extract ‘Role’, ‘Service Line’, ‘Technology’ and ‘Project Duration’ values from the given resumes and then perform a set of consistency checks to estimate the correct proficiency level for each technology skill. If no ‘Role’ is mentioned in a project, then use a user-specified mapping SL_to_R that maps ‘Service Line’ values to an appropriate ‘Role’; e.g., ‘Service Line’ values like *attending design*, *building*, *coding*, *conversion*, *definition*, *design*, *design development* map to the ‘Role’ value *Developer*. Essentially, ‘Service Line’ values are related to ‘Role’ values like *developer*, *tester*, *administrator*, *manager* etc. Similarly, we use another user-defined mapping T_to_C that maps a ‘technology’ value to a competency or skill; e.g., ‘Technology’ values *Portal*, *User Information*, *Weblogic Portal*, *Tibco Portal Builder* or *Enterprise Portal* are all mapped to the ‘Competency’ value *Enterprise Portal*. Similarly, the mapping C_to_RC specifies the ‘Role Category’ values for each ‘Competency’ value, at least one of which must be present to assign a ‘Competency’ to a person; e.g., ‘Competency’ *BEA_Weblogic_Application_Server* can be assigned to a person only if he has at least one ‘Role Category’ value from {*Developer*, *Designer*, *Technical Architect*}.

Following algorithm identifies all the ‘Competency’ values for a given resume.

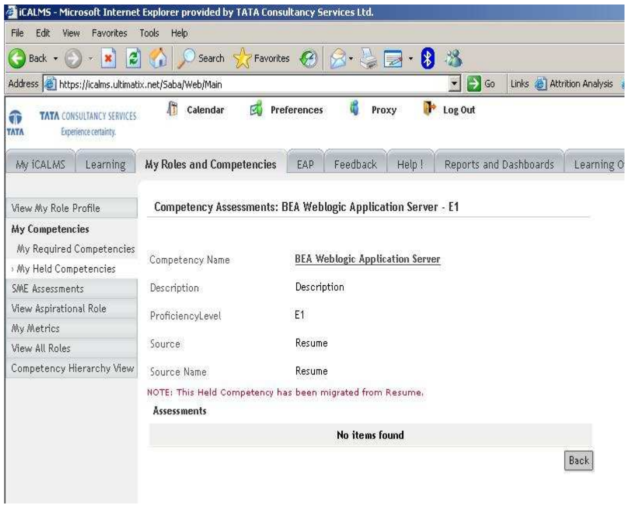
```
L = ∅; // list of competencies for the given resume
for each project P in the given resume do
    Let RL be the list of ‘Role Category’ values in P;
    Let SL be the list of ‘Service Line’ values in P;
    Let TL be the list of ‘technology’ values in P;
    if RL == ∅ then // no role mentioned in P
        use mapping SL_to_R to map values in SL to an appropriate
            role for P; // e.g., programming
    end if
    use mapping T_to_C to map values in TL to an appropriate
            ‘Competency’ values for P;
    Let CL be the resulting list of ‘Competency’ values.
    For each ‘Competency’ value c in CL do
        if all the mandatory ‘Technology’ values for c (as
            specified in mapping C_to_T) are present in TL and
            at least one mandatory ‘Role’ value for c is present in RL
        then
            add c to L;
        end if
    end for
end for
```

Project	MasterCraft Java 6.5.2 and 7.0 – Server side enhancements and support
Customer	TCS Internal
Period	Jan 06 – Till date
Description	As part of the MasterCraft Java – Server side, support and Training had to be given to the existing clients of MasterCraft 6.5.2 and 7.0 and bugs reported by Clients had to be fixed. Features enhancements were as follows. • Support for Queue implementaion in Message Driven Beans in MasterCraft 6.5.2 and 7.0 • Support for CLOB datatype in MasterCraft 6.5.2 • Support for WCHAR in MasterCraft 6.5.2 • Support for BEA Weblogic Application Server 9.0 in MasterCraft 6.5.2 Modification of Batch Framework in MasterCraft 7.0
Role	Developer and Team Lead
Solution Environment	WINDOWS-2000/XP
Tools	MasterCraft Versions 6.5.2 & 7.0, Adex 1.8.

Figure 9. Example project description.

Consider the example project description in Figure 9. The 'Role' values mentioned are *Developer* and *Team lead*, which maps (using mappoing R_to_RC) to the 'Role Category' value *Developer*. 'Technology' value *Stellant* is mapped (using mapping T_to_C) to the competency *Oracle_ECM_Stellent*. Similarly, 'Technology' value *bea weblogic application server* is mapped to the 'Competency' value *bea_weblogic_application_server*.

Using the mapping C_to_T, for 'Competency' value *bea_weblogic_application_server*, the mandatory 'Technology' values are *bea application* and *weblogic server*. Both these are present in the list of 'Technology' values extracted from the resume. Also, the 'Competency' *BEA_Weblogic_Application_Server* requires at least one 'Role Category' from {*Developer*, *Designer*, *Technical Architect*}, which is present in the resume. Hence, this competency is validated and it is assigned. Similarly for 'Competency' value *Oracle_ECM_Stellent,* both the mandatory 'technology' values *ECM* and *Stellant* should appear in the complete list of 'Technology' values in the resume. This step is needed to filter the correct competencies, because in the earlier step, multiple competencies may be selected based on the single technology keyword. Figure 10 shows the iCalms (Skill repository) screenshot for this example.


iCALMS - Microsoft Internet Explorer provided by TATA Consultancy Services Ltd.
File Edit View Favorites Tools Help
Back Search Favorites
Address https://icalms.ultimatix.net/Saba/Web/Main Go Links Attrition Analysis
TATA CONSULTANCY SERVICES Experience certainty.
Calendar Preferences Proxy Log Out
My iCALMS Learning My Roles and Competencies EAP Feedback Help ! Reports and Dashboards Learning O
View My Role Profile
My Competencies
My Required Competencies
My Held Competencies
SME Assessments
View Aspirational Role
My Metrics
View All Roles
Competency Hierarchy View
Back

Competency Assessments: BEA Weblogic Application Server - E1

Competency Name	BEA Weblogic Application Server
Description	Description
ProficiencyLevel	E1
Source	Resume
Source Name	Resume

NOTE: This Held Competency has been migrated from Resume.

Assessments

No items found

Figure 10. iCalms (Skill repository) screen-shot for the updated competency from the example resume.

The case-study was implemented by deploying RINX on a pool of 4 servers, processing resumes in parallel. Overall, 65675 resumes were processed and competencies extracted from each of them except about 400 resumes which were corrupted while saving due to incorrectly renaming as MS Word *.doc* files. The output of RINX was manually validated by HR experts on selected resumes.

5. CONCLUSIONS AND FURTHER WORK

In this paper, we described a tool called RINX, which uses deep linguistic and domain knowledge to extract various types of information elements from resumes. The results of applying RINX on an initial set of resumes indicated that an iterative approach is needed for generating and refining an exhaustive pattern set. The results show that the performance of the patterns does improve over iterations. We terminated the iterations upon reaching a sort of stable state where further refinement of patterns to improve performance seemed difficult. When applied on the test data, the final patterns are observed to give satisfactory results. Through the iterative process, the patterns increased considerably in their complexity.

A unique feature of our patterns is that we have combined external knowledge sources (WordNet, domain specific dictionaries etc.) with regular expressions. The resulting patterns seem to be robust enough to maintain similar levels of accuracy on the other resume datasets. We adopted a rather conservative approach when constructing the patterns for the entities, because of which we got 0% false positives on test dataset (very low false positive rate was mandated by the client).

Inconsistencies added during the PDF-to-text conversion process affected the performance (mostly resulted in false negatives). We used only minimal cues from the structure of the documents (font size, section header etc.). But we believe that the patterns can be improved by using structural information in the documents.

Constraints very specific to the IT domain were instrumental in choosing a manual and iterative approach for creating and refining patterns to identify the entities: 'Service Line', 'Role', ‘Competency’ etc. The patterns had to be robust in the face of extremely noisy text in the resume documents. Through this exercise, we have reiterated the strengths of manual analysis and effectiveness of linguistically oriented patterns for identifying domain-specific entities.

As mentioned, the IT organization mentioned in the case-study has millions of resumes in its repository. The main question is whether the patterns will work well in spite of the varieties. How does one evaluate the effectiveness of the patterns “on the fly”, given the fact that the client will not spend too much manual efforts in creating any further labeled training datasets? We are planning to design a linguistically oriented system to automatically identify candidate entities which were not identified by the patterns (i.e., were false negatives). Such an approach can use, for example, textual similarity and database of known entities. Output of such a system can be evaluated to see whether the patterns’ performance is degrading “too much”.

If the performance of the patterns is indeed detected to be deteriorating, then the next question is: how do we refine the patterns to improve their performance on the new data? The option of manually analyzing the failures and refine the patterns to cover them are likely to be too effort-intensive for the client. We are exploring a machine learning (ML) based approach to automatically come up with refinements of the patterns. We can term this as a hybrid approach, since the manually created initial patterns would now be automatically refined using ML techniques. We plan to investigate use of standard ML techniques like HMM, CRF for this purpose. We need to see how to adapt the standard ML techniques to handle special constraints present in this domain. As mentioned, we are also interested in using structural information a document can provide. We believe a sophisticated document classifier would also help in boosting our system’s performance. Finally, we plan to build a knowledge repository for this domain using the knowledge we have gained during our exercise.

6. ACKNOWLEDGEMENTS

The authors would like to thank the RINX team for their untiring efforts and enthusiasm in design, implementation, testing and case-study phases. We sincerely thank Prof. Harrick Vin for his constant support and encouragement. We also thank S Narasimhan, Vice President, Global Head - Learning & Development (L&D), Tata Consultancy Services (TCS), Murali Padmanabhan, Head, Learning Technology Solutions Group, L&D, TCS and associated HR teams for giving us an opportunity to use RINX in a real-life case-study. We would like to thank K. Ananth Krishnan, who directed the TCS L&D team to our group for helping them with the huge task of updating iCALMS with held competencies of associates.

7. REFERENCES

[1] D.E. Appelt, R.H. Jerry, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Myers, M. Tyson, 1995. SRI international FASTUS system MUC-6 test results and analysis, *Proc. 6th Message Understanding Conference (MUC-6)*, pp. 237–248, Morgan Kaufmann.

[2] M. Berland, E. Charniak, Finding parts in very large corpora, 1999, *Proc. 37th Annual Meeting of the Asso. For Computational Linguistics (ACL99)*.

[3] D.M. Bikel, R. Schwartz, R.M. Weischedel, 1999. An algorithm that learns what’s in a name, *Machine Learning*, 34, pp. 211 – 231.

[4] N. Collier, C. Nobata, J-i. Tsujii, 2000. Extracting the names of genes and gene products with a hidden Markov model, *Proc. 18th Int. Conf. Computational Linguistics (COLING2000)*, pp. 201 – 207.

[5] M. Collins, Y. Singer, 1999. Unsupervised models for named entity classification, *Proc. EMNLP*.

[6] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D.S. Weld, A. Yates, 2005. Unsupervised named-entity extraction from the Web: An experimental study, *Artificial Intelligence*, 165, pp. 91–134.

[7] A. Jimeno, E. Jimenez-Ruiz, V. Lee, S. Gaudan, R. Berlanga, D. Rebholz-Schuhmann, 2007. Assessment of disease named entity recognition on a corpus of annotated sentences, *Proc. 2nd Int. Sympo. Languages in Biology and Medicine (LBM)*.

[8] D. Hanisch, J. Fluck, H. Mevissen, R. Zimmer, 2003. Playing biology’s name game: identifying protein names in scientific text, Proc. PSB’03.

[9] J-i. Kazama, T. Makino, Y. Ohta, J-i. Tsujii, 2002. Tuning support vector machines for biomedical named entity recognition, *Proc. ACL-02 Workshop on Natural Language Processing in the Biomedical Domain*, pp. 1–8.

[10] D. Klein, J. Smarr, H. Nguyen, C.D. Manning, 2003. Named Entity Recognition with Character-Level Models, *Proc. 7th CoNLLL*, pp. 180-183.

[11] K.-J. Lee, Y.-S. Hwang, H.-C. Rim, 2003. Two-phase biomedical NER recognition based on SVMs, in ACL2003.

[12] W. Li, A. McCallum, 2003. Rapid development of Hindi named entity recognition using conditional random fields and feature induction, *ACM Tran. Asian language Info. Processing*, 2(3), Sep. 2003, pp. 290 – 294.

[13] Y.-F. Lin, T.-H. Tsai, W.-C. Chou, K.-P. Wu, T.-Y. Sung, W.-L. Hsu, 2004. A maximum entropy approach to biomedical named entity recognition, *Proc. 4th Workshop on Data Mining in Bioinformatics (BIOKDD04)*.

[14] D. Maynard, V. Tablan, C. Ursu, H. Cunningham, Y. Wilks, 2001. Named entity recognition from diverse text types, *Proc. Recent Advances in Natural Language Processing (RANLP2001) Conference*.

[15] A. McCallum, W. Li, 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons, *Proc. Conf. Natural Language Learning*, pages 188–191.

[16] D. Nadeau, P. Turney, S. Matwin, 2006. Unsupervised named-entity recognition: generating gazetteers and resolving ambiguity, *Proc. 19th Canadian Conf. Artificial Intelligence*.

[17] B. Settles, 2004. Biomedical named entity recognition using conditional random fields and rich feature sets, *Proc. COLING 2004 Int. Joint Workshop on Natural Language*

Processing in Biomedicine and its Applications (NLPBA), Geneva, Switzerland.

[18] B. Sundheim (ed.), 1998. *Proc. 7th Message Understanding Conference (MUC-7)*, ARPA, Morgan Kaufmann.

[19] Y. Tsuruoka, J. Tsujii, 2003. Boosting precision and recall of dictionary-based protein name recognition. in *ACL 2003*.

[20] K. Uchimoto, M. Murata, Q. Ma, H. Ozaku, H. Isahara, 2000. Named entity extraction based on a maximum entropy model and transformation rules, *Proc. of the 38th ACL*, pages 326–335.

[21] R. Weischedel, M. Meteer, R. Schwartz, L. Ramshaw, J. Palmucci, 1993. Coping with ambiguity and unknown words through probabilistic methods, *Computational Linguistics*, 19(2), pp. 359–382.