# Automatic Shortlisting of Candidates in Recruitment

Girish Keshav Palshikar, Rajiv Srivastava, Mahek Shah, Sachin Pawar
TCS Research and Innovation, Tata Consultancy Services Limited
{gk.palshikar, rajiv.srivastava, shah.mahek, sachin7.p}@tcs.com

## Abstract

Talent acquisition is an important, complex and effort-intensive function within Human Resources (HR) management. We describe a system called TEAC that processes a set of given free-form textual resumes (in English for IT domain), creates a standardized profile for each candidate, and for a given job description, identifies a ranked shortlist of $k$ candidates, along with a matching score for each. The resume scoring function is hand-crafted, hierarchical, and uses domain-knowledge from recruitment experts. We describe a simple neural-network system that automatically learns some weights used in the scoring function, based on feedback about whether the candidate was SELECTED or REJECTED. The TEAC system is deployed in a large multinational IT services organization.

## 1 Introduction

Talent acquisition is an important, complex and effort-intensive function within Human Resources (HR) management. TCS currently employs about 400,000 people and as per the TCS Annual Report 2015-16, in the fiscal year 2015-2016, TCS hired 90,182 people (about 97% with IT background), 74,009 in India and 16,173 outside India. Assuming 20% selection rate, about 5 persons were interviewed for each selection. For creating the interview shortlist of 5 persons per post, HR executives may have read and evaluated at least 10 resumes per post. Assuming an average effort of 5 minutes per resume, this translates to approx. 75,000 person-hours p.a., spent only for resume evaluation and shortlist creation. Apart from efforts, the human evaluation of resumes is often subjective, error-prone, opaque, does not facilitate comparisons, and opportunities for improving quality of recruitment are difficult to spot.

In this paper, we describe a system called *TEAC (Talent Evaluation and Assessment of Candidates)* (Fig. 1). The main functionality of this system is as follows. In the *profile creation* step, TEAC processes a set of given free-form textual resumes (in English) and creates a *standardized profile* for each candidate. In this paper, we focus on resumes of IT professionals only. Since information for each candidate is in a fixed structure (called her *profile*) which is same for all candidates, it becomes easy to search, analyze and compare candidates. The standardized profiles of candidates also help in the interview process. In the *matching* (or *shortlisting*) step, TEAC accepts a *job description* (JD), identifies a shortlist of $k$ candidates for this JD, along with a *matching score* for each, from among a set of profiles (already created in the profile creation step), based on a specific matching (scoring) function. The resume scoring function is hand-crafted, hierarchical, and uses domain-knowledge from recruitment experts. TEAC system is deployed in a large multinational IT organization.

This paper is organized as follows. In Sections 2 and 3, we discuss the profile creation and automatic shortlist creation parts respectively. In Section 4 we evaluate a simple neural net approach to learn some of the weights

In: Joint Proceedings of the First International Workshop on Professional Search (ProfS2018); the Second Workshop on Knowledge Graphs and Semantics for Text Retrieval, Analysis, and Understanding (KG4IR); and the International Workshop on Data Search (DATA:SEARCH18). Co-located with SIGIR 2018, Ann Arbor, Michigan, USA – 12 July 2018, published at http://ceur-ws.org

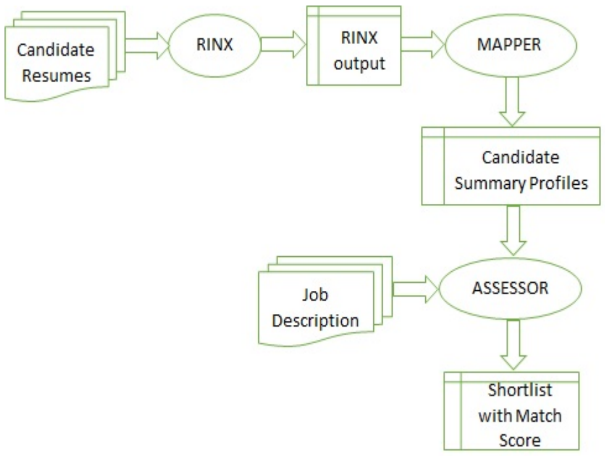
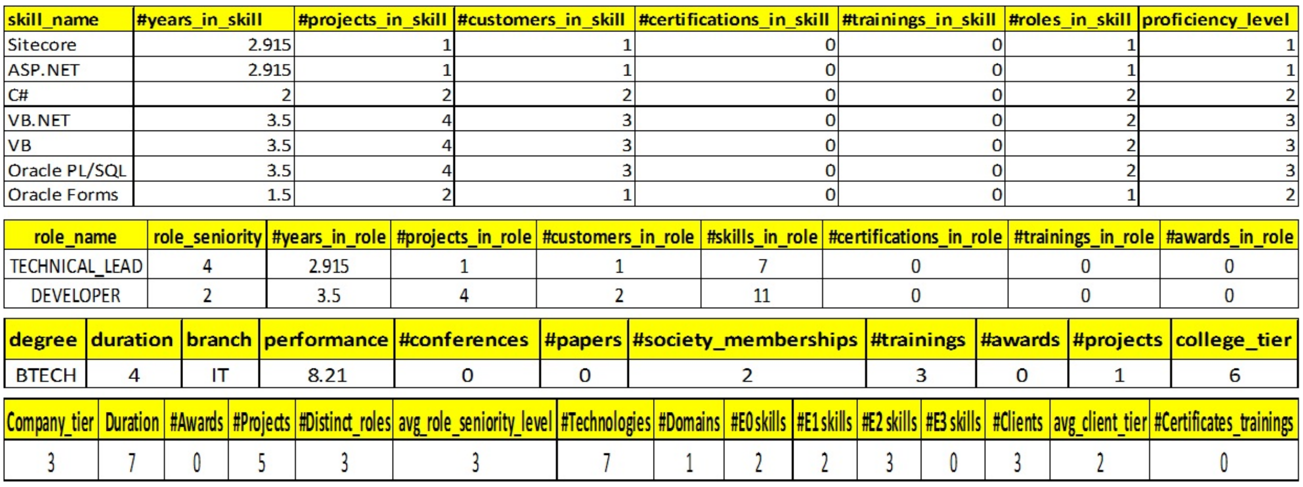


Figure 1: The TEAC system.



| skill_name | #years_in_skill | #projects_in_skill | #customers_in_skill | #certifications_in_skill | #trainings_in_skill | #roles_in_skill | proficiency_level |
|---|---|---|---|---|---|---|---|
| Sitecore | 2.915 | 1 | 1 | 0 | 0 | 1 | 1 |
| ASP.NET | 2.915 | 1 | 1 | 0 | 0 | 1 | 1 |
| C# | 2 | 2 | 2 | 0 | 0 | 2 | 2 |
| VB.NET | 3.5 | 4 | 3 | 0 | 0 | 2 | 3 |
| VB | 3.5 | 4 | 3 | 0 | 0 | 2 | 3 |
| Oracle PL/SQL | 3.5 | 4 | 3 | 0 | 0 | 2 | 3 |
| Oracle Forms | 1.5 | 2 | 1 | 0 | 0 | 1 | 2 |

| role_name | role_seniority | #years_in_role | #projects_in_role | #customers_in_role | #skills_in_role | #certifications_in_role | #trainings_in_role | #awards_in_role |
|---|---|---|---|---|---|---|---|---|
| TECHNICAL_LEAD | 4 | 2.915 | 1 | 1 | 7 | 0 | 0 | 0 |
| DEVELOPER | 2 | 3.5 | 4 | 2 | 11 | 0 | 0 | 0 |

| degree | duration | branch | performance | #conferences | #papers | #society_memberships | #trainings | #awards | #projects | college_tier |
|---|---|---|---|---|---|---|---|---|---|---|
| BTECH | 4 | IT | 8.21 | 0 | 0 | 2 | 3 | 0 | 1 | 6 |

| Company_tier | Duration | #Awards | #Projects | #Distinct_roles | avg_role_seniority_level | #Technologies | #Domains | #E0skills | #E1skills | #E2skills | #E3skills | #Clients | avg_client_tier | #Certificates_trainings |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 0 | 5 | 3 | 3 | 7 | 1 | 2 | 2 | 3 | 0 | 3 | 2 | 0 |

Figure 2: Sub-profiles created for each candidate.

involved in our resume scoring function based on feedback about whether the candidate was SELECTED or REJECTED. In Section 5 we empirically evaluate the proposed system. Section 6 contains related work; Section 7 gives conclusions and future work.

## 2 Profile Creation

Resumes of IT professionals have a somewhat common conceptual structure. Such a resume typically contains several *sections*, such as personal information, education, employment history, project details, trainings and certifications, awards and achievements, publications, professional summary etc. Each resume need not contain all these sections and some resumes may contain additional sections. The order of sections may vary across resumes, and there is a great variety in the contents and expression of each section.

We have built a machine-learning based information extraction tool called RINX to identify various sections and to extract various types of *entities*, their *attributes* and *relations* among these entities from the given resume. Entity type SKILL refers to IT technology platforms, systems, products, frameworks, programming languages, operating systems etc.; examples of *mentions* of this entity type are `Java, ASP.NET, SAP, Linux, Oracle, J2EE`. Entity type ROLE describes the type of tasks handled and work done in an IT project; examples of *mentions* of this entity type are `developer, technical architect, test engineer, project leader`. Mentions of the entity type PROJECT typically refer to the title of an IT project in a resume; e.g., `Utilities and scripts for production support`. A binary *relation type* connects two entity types; e.g., the relation *used_in* connects an entity of type SKILL and an entity of type PROJECT. Thus, if a specific project $p$ uses a particular skill $s$, then RINX will extract a *relation mention* *used_in*$(s, p)$. Some attributes of an entity type are directly extracted by RINX and some are derived from relations between two entities. For example, RINX extracts mentions of the relation *used_in* and the attribute #skills for a particular project is the count of these relation mentions within that specific project section. This extracted information about a candidate is arranged as a set of *sub-profiles*. Fig. 2 shows 4 sub-profiles for a candidate; e.g., the skill sub-profile for a candidate has one row for each of her skills, with the columns as listed. Using simple rules, each actual role mentioned in a resume is mapped to one role in a standard set of roles (along with a *seniority level* - a number between 1 to 15 - for each role) in our organization; e.g., `EDI-SAP Analyst` is mapped to the standard role ANALYST.

We have defined a scale for assigning *proficiency level* to a particular skill in a particular candidate: $E0, E1, E2, E3, E4$, with $E0$ denoting a basic familiarity and $E4$ denoting the highest level expertise; e.g., a candidate may be assigned $E2$ in `SAP`, $E1$ in `Java`, and $E1$ in `Oracle RDBMS`. We have defined a simple rule-based model to compute the proficiency level of a candidate in a given skill, using values of the corresponding attributes of that skill in the skill profile (Fig. 2). *Tier* of an employer company $c$ is derived using simple rules (higher values are better); an example of such a rule could be: IF $c \in Fortune1000 \vee c$ is a government department $\vee$ $c$.turnover $>$ USD 5 billion THEN tier = 5. *Tier* of a university is derived using some public-domain ranking. *Trainings* refer to internal trainings within an organization; *certification* is based on an external examination, such as `Microsoft Certified Technology Specialist (MCTS) 3.0 Web application`.

RINX has 3 different methods of extracting entities, attributes and relations. In a *supervised* method, RINX uses labeled training data to learn a binary classifier model (e.g., CRF or SVM) which can then be used for extracting mentions of a particular entity type from an unlabeled resume. In a *gazette*-based method, RINX starts with a small seed list of mentions of a particular entity type and iteratively discovers a list of other mentions in an unlabeled corpus of resumes. Thus starting with 5 seed examples of the entity type DEGREE, RINX discovered about 400 other degree names in a corpus of resumes. Extraction of mentions of an entity type using a gazette $G$ is straightforward and involves identifying all strings which approximately match any mention in $G$. RINX also contains hand-crafted rule-based methods to extract entities such as ROLE and DATE. After RINX extracts all the entities, attributes and relations from a resume, the MAPPER component in TEAC creates the various sub-profiles for that candidate.

## 3 Shortlist Creation

After the sub-profiles are created from the uploaded resumes of some candidates, TEAC accepts a *job description (JD)* from a user and creates a shortlist of $k$ candidates, sorted on a similarity (or matching) score for each candidate, indicating how well the candidate matches the JD. A JD consists of a list of skills (e.g., `SiteCore, ASP.NET`, intended role (e.g., DEVELOPER), total experience (e.g., $3-6$ years) etc. We now describe a hand-crafted hierarchical matching function that computes a similarity score between a JD and a candidate (i.e., her sub-profiles). We have defined an unsupervised algorithm that computes similarity score $sim_1(x, y)$ between any two given skills $x$ and $y$, based on, say, the similarity between their Wikipedia pages. $sim_2$ computes the similarity between a skill $x$ from a JD and a row for skill $y$ from the skill sub-profile for a given candidate. Here, $w_1, \ldots, w_7$ are user-defined non-negative weights for the 7 variables in the skill sub-profile.

$$sim_2(x, (y, y_1, y_2, y_3, y_4, y_5, y_6, y_7)) = sim_1(x, y) \cdot \frac{(w_1 y_1 + \ldots + w_7 y_7)}{(w_1 + \ldots + w_7)} \tag{1}$$

The function $sim_3$ gives the overall similarity between a skill $x$ in the given JD and all the skills in the skill sub-profile of the candidate. $sim_3$ takes the top match similarity as it is and adds the average of $sim_2$ values of the remaining skills in the skill sub-profile of the candidate whose similarity with $x$ is above some user-specified threshold $\theta_0$. Function $sim_4$ computes the total matching score between all skills in the given JD and all skills in the skill sub-profile of the given candidate. Suppose there are at most $K_1$ skills specified in a JD (we use $K_1 = 5$). Let $u_1, \ldots, u_5$ denote the user-defined weights of the first, second, third etc. skills in a JD. Then $sim_4$ computes the weighted average of the $n$ $sim_3$ values, where $n$ is the no. of skills in the JD.

Similarity between a role in JD and a given role sub-profile is computed as follows. Similarity function $rsim_1(a, b)$ two roles $a$ (from JD) and $b$ (from role profile) is as follows, where $sen(x)$ denotes the seniority of role $x$ and $maxsen$ is 15.

$$\begin{aligned} rsim_1(a, b) &= 1 - \frac{sen(a) - sen(b)}{maxsen} \text{if sen(b) > sen(a)} \\ &= \frac{sen(a)}{sen(a) - sen(b)} \text{ otherwise} \end{aligned} \tag{2}$$

Function $rsim_2$ computes the similarity between a role $x$ from a JD and an entire row for role $y$ from the role sub-profile for a given candidate. The formula is similar to that of $sim_2$. Function $rsim_3$ gives the overall similarity between the role in a given JD and all the roles in the role sub-profile of the candidate. Its formula is similar to that of $sim_3$. Function $expsim$ computes the similarity between the total experience of a candidate and the experience range in a JD using a triangle function .

In addition to how well the skills, roles and experience of a candidate match those specified in a JD, we consider the *quality* of a candidate when preparing the shortlist, so that higher quality candidates come up. We measure the quality of a candidate (independent of any JD) in terms of the quality of her education and quality of her work experience. We have defined a set of rules for assessing the quality of a candidate's education profile. For example, the performance of a candidate during a degree is typically given in terms of % of marks, GPA, or CGPA (averaged over all semesters). After bringing the performance number from a resume to a common range $(0-10)$, we define a simple rule that gives an assessment score of performance: IF $performance \leq 6.0$ THEN $score = 2$ ELSE IF $6.0 < performance \leq 8.0$ THEN $score = 3$ ELSE IF $performance > 8.0$ THEN $score = 4$. The overall quality score for the education sub-profile of a candidate is the average of the assessment scores for each of the columns in it.

We have also defined a set of rules for assessing the quality of a candidate's work experience sub-profile. A rule like IF $company\_tier == 1$ THEN $score = 1$ ELSE IF $company\_tier == 2$ THEN $score = 2.5$ ELSE IF $company\_tier == 3$ THEN $score = 4$ gives preference to a candidate who has worked for a high tier employer. In a similar manner, rules compute the score for other attributes in the work experience sub-profile, such as $\#awards$, $\#certifications\_trainings$ etc. The overall quality score for the work experience sub-profile of a candidate is the average of the assessment scores for each of the columns in it.

Let $Q = (S, r, (t_1, t_2))$ denote a JD, where $S = \langle s_1, s_2, , s_n \rangle$ is an ordered list of skills, $r$ is a role, and $t_1, t_2$ denote the experience range. As an example, $Q = (\{\texttt{sitecore}, \texttt{ASP.NET}\}, DEVELOPER, (3, 6))$. Let $\sigma^{(skill)}, \sigma^{(role)}, \sigma^{(edu)}, \sigma^{(exp)}$ denote the sub-profiles for a particular candidate. Each sub-profile is a set of vectors. We denote $i$-th element of a sub-profile (which is a vector) $\sigma$ by $\sigma_i$; e.g., in Fig. 2, $\sigma^{(role)} = \{\sigma_1^{(role)}, \sigma_2^{(role)}\}$, where $\sigma_1^{(role)} = \langle TECHNICAL\_LEAD, 4, 2.915, 1, 1, 7, 0, 0, 0 \rangle$ and $\sigma_2^{(role)} = \langle DEVELOPER, 2, 3.5, 4, 2, 11, 0, 0, 0 \rangle$. Our skill similarity function gives $sim_1(\texttt{Sitecore}, \texttt{ASP.NET}) = 0.4667$, $sim_1(\texttt{Sitecore}, \texttt{C\#}) = 0.363$ etc. Using the weights $w_1 = 0.5, w_2 = 0.2, w_3 = 0.02, w_4 = 0.04, w_5 = 0.04, w_6 = 0.1, w_7 = 0.1$ for the 7 variables in each row of the sill sub-profile, we compute, using Eq. (1):

$$sim_2(\texttt{Sitecore}, \sigma_1^{(skill)}) = sim_2(\texttt{Sitecore}, (\texttt{Sitecore}, 2.915, 1, 1, 0, 0, 1, 1))$$
$$= sim_1(\texttt{Sitecore}, \texttt{Sitecore}) \cdot (0.5 \cdot 2.915 + 0.2 \cdot 1 + 0.02 \cdot 1 + 0.04 \cdot 0 + 0.04 \cdot 0 + 0.1 \cdot 1 + 0.1 \cdot 1)$$
$$= 1.0 \cdot (0.5 \cdot 2.915 + 0.2 \cdot 1 + 0.02 \cdot 1 + 0.04 \cdot 0 + 0.04 \cdot 0 + 0.1 \cdot 1 + 0.1 \cdot 1) = 1.8775$$

Similarly, we get
$sim_2(\texttt{Sitecore}, \sigma_2^{(skill)}) = 0.87622925$, $sim_2(\texttt{Sitecore}, \sigma_3^{(skill)}) = 0.66792$,
$sim_2(\texttt{Sitecore}, \sigma_4^{(skill)}) = 1.86289$, $sim_2(\texttt{Sitecore}, \sigma_5^{(skill)}) = 1.604138$,
$sim_2(\texttt{Sitecore}, \sigma_6^{(skill)}) = 0.729295$, and $sim_2(\texttt{Sitecore}, \sigma_7^{(skill)}) = 0.06909$.
Then the similarity between the first skill in the JD and the candidate skill profile is:
$sim_3(\texttt{Sitecore}, \sigma^{(skill)}) = 1.8775 + ((0.87622925 + 0.66792 + 1.86289 + 1.604138)/4) = 3.1302943125$

We did not consider `Oracle PL/SQL` and `Oracle Forms` in computing $sim_3$, because $sim_1(\texttt{Sitecore}, \texttt{OraclePL/SQL}) = 0.2345 < \theta_0$ and $sim_1(\texttt{Sitecore}, \texttt{OracleForms}) = 0.047 < \theta_0$, where we assume $\theta_0 = 0.3$. In a similar manner, we get $sim_3(\texttt{ASP.NET}, \sigma^{(skill)}) = 3.63678845$. The matching score function $sim_4$ between all the skills in the given JD and the skill sub-profile of the given candidate is:

$sim_4(Q, \sigma^{(skill)}) = (1.0 \cdot 3.1302943125 + 0.8 \cdot 3.63678845)/(1.0 + 0.8) = 3.3554028$

Assuming $sen(DEVELOPER) = 2$ and $sen(TECHNICAL\_LEAD) = 4$, using Eq. (2), we get $rsim_1(DEVELOPER, DEVELOPER) = 1$ and $rsim_1(DEVELOPER, TECHNICAL\_LEAD) = 0.867$. We assume the weights $u_1 = 0.6, u_2 = 0.1, u_3 = 0.05, u_4 = 0.1, u_5 = 0.05, u_6 = 0.05, u_7 = 0.05$. Then, the similarity between the JD role (DEVELOPER) and one of the roles (TECHNICAL_LEAD) in the role sub-profile is:

$$rsim_2(DEVELOPER, \sigma_1^{(role)}) = rsim_2(DEVELOPER, (TECHNICAL\_LEAD, 2.915, 1, 1, 7, 0, 0, 0))$$

$= rsim_1(DEVELOPER, TEHNICAL_LEAD) \cdot (0.6 \cdot 2.915 + 0.1 \cdot 1 + 0.05 \cdot 1 + 0.1 \cdot 7 + 0.05 \cdot 0 + 0.05 \cdot 0 + 0.05 \cdot 0)$
$= 0.867 \cdot (0.6 \cdot 2.915 + 0.1 \cdot 1 + 0.05 \cdot 1 + 0.1 \cdot 7 + 0.05 \cdot 0 + 0.05 \cdot 0 + 0.05 \cdot 0) = 2.253333$

Similarly, we get $rsim_2(DEVELOPER, \sigma_2^{(role)}) = 3.7$. Then the similarity between the JD role and the role sub-profile of the candidate is: $rsim_3(DEVELOPER, \sigma^{(role)}) = (2.253333 + 3.7)/2 = 2.9766665$. The match score for her experience is $expsim = 0$. The total match score for the example candidate with respect to the example JD is: $0 + 3.355 + 2.977 = 6.332$. Using various rules, we get the quality score for this candidate's education sub-profile as 6.5, for her work experience profile it is 1.396, giving the overall quality score as $6.5 + 1.396 = 7.896$. The overall matching score for this candidate with the given JD is: $6.332 + 7.896 = 14.228$.

## 4 Improving Scoring using Feedback

Our shortlist creation function is a hierarchical hand-crafted knowledge-rich function for ranking resumes with respect to a given job description. We came up with this candidate scoring function in close cooperation with HR professionals, who gave examples of candidates and their scores as well as guided us on the detailed aspects that they included when assessing candidates. Intuitively, their requirements for candidate filtering (with respect to a specific JD) were about close "match" with one or more skills, and roles. In addition, they often preferred "high quality" candidates. The problem was that, although there is very good understanding with each HR professional about matching and quality, this understanding is not quantified formally. We noted with interest that the HR professionals often had somewhat different evaluation criteria for matching and quality, and the candidate scoring function reported here is our attempt to consolidate the different subjective ways into a more objective automated scoring function. After the TEAC system became operational, they monitored the scores generated by the system for different candidates and satisfied themselves that the these scores were reasonable. We have not done a formal evaluation of the scores produced by TEAC for various candidates. However, we have evaluated the quality of the scores in another way which is more relevant to the HR professionals, and report the results in section 5.

The candidate scoring function involves many weights that are set initially by the users. We now report a simple learning algorithm that automatically adjusts these weights by observing the end result of recruitment (SELECTED or REJECTED). For a JD $J_i$ and the corresponding resume set $S_i$ of size $M_i$, we receive a feedback from the HR executives, which is in the form of a flag $F_j \in \{0, 1\}$ for every resume $R_j \in S_i, 1 \leq j \leq M_i$ (0 indicates REJECTED, 1 indicates SELECTED). We translate this into a pairwise preference data as follows: for every pair of resumes $R_j, R_k \in S_i, 1 \leq j, k \leq M_i$, such that their flags are different ($F_j \neq F_k$), we generate a preference record $R_j \prec R_k$ if $F_k > F_j$, and $R_k \prec R_j$ if $F_j > F_k$. We represent each resume $R_j$ as a vector $\mathbf{v_j}$ of 5 score values: scores for skill sub-profile, role sub-profile, education sub-profile, work experience sub-profile and total experience. 3 of these 5 scores are with respect to a given JD. A resume pair $R_j, R_k$ is presented as the tuple $(\mathbf{v_j}, \mathbf{v_k}, g_{jk})$, where $g_{jk} = 1$ if $R_k \prec R_j$, and 0 otherwise. We use such tuples to train a neural network. The goal of this ANN is to learn the weights in a linear score combination function $a_1 v_1 + \ldots + a_5 v_5$, where the 5 weights $a_i$ are unknown and $v_1, \ldots, v_5$ are the sub-profile scores for a resume. During the training phase, the ANN starts with some initial values for $a_i$ and keeps adjusting the weights $a_i$ to conform to the relative preference rank of the given pair of resumes. After the training phase is over, the final values of $a_i$ are used to compute the score for any resume, and to create a shortlist by selecting $k_0$ resumes having the highest scores.

## 5 Experimental Evaluation

We do not report a direct evaluation of the quality of the shortlists created by TEAC. We report an indirect evaluation, which is more useful for HR executives. In this experiment, we collected sets $S_1, \ldots, S_N$ of resumes, and corresponding $J_1, \ldots, J_N$ JDs. Candidates with resumes in set $S_i$ were interviewed and the HR executives also gave us the result flag (SELECTED or REJECTED) for each candidate in each $S_i$. To evaluate TEAC, for each set $S_i$, we used TEAC to create a shortlist of top $k_0$ resumes for each $S_i$, ordered in terms of their matching score for JD $J_i$. The effectiveness of TEAC is measured in terms of the fraction of SELECTED candidates in the TEAC shortlist of size $k_0$. We define a measure $avg\_precision@k_0$, which is simply the average value of this fraction for the $N$ sets. Note that the sizes of the sets $S_i$ are different and the numbers of SELECTED candidates in the sets $S_i$ also vary from set to set. For testing, we considered 21 resume sets, containing a total of 352 resumes, and a JD for each resume set. For each resume in each set, we also had a SELECTED/REJECTED flag (130 SELECTED, 222 REJECTED). Fig. 3 shows the $avg\_precision@k_0$ for various values of $k_0$, using both

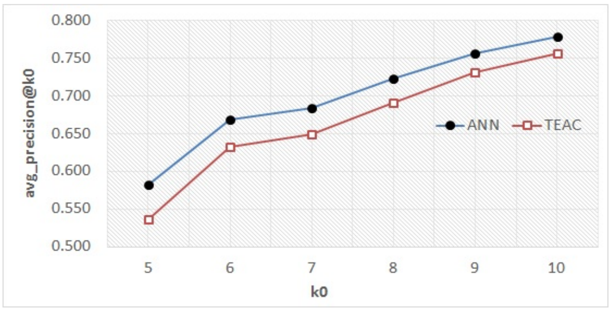


Figure 3: Evaluation of shortlists.

the default values of these 5 weights in TEAC (all values are 1) and the new weights learned by the ANN (0.946, 0.833, 1.019, 0.841, 0.865). The ANN shows an improvement in the quality of shortlists created.

## 6 Related Work

Recruitment is a rich source of problems for applying data mining and machine learning algorithms [7], [12]. [8] explores the use of soft set theory in screening, which is similar to our shortlist creation. [2] handle the candidate shortlisting problem in a different setting: recruiters searching a very large set of profiles on LinkedIn; they proposed a scalable machine learning algorithm that prunes irrelevant documents. Faliagka et al in [5], [4] use machine learning algorithms for learning weights assuming the linear or piece-wise linear ranking function defined over profile entities including skill, education and personality traits. Unlike in this paper, the learnt function is not hierarchical over profile attributes and hence much simpler. Also the models learnt using SVR or rankSVM are not amenable for user interpretation due to the kernel transformation [10]. Fazel-Zarandi et al in [6] use description logics (DL) to represent the profile features and the JD criteria. The features of competence, experience and education from a profile are written as logic rules and are used in matching. The simpler rule based approach does not use quantification of attributes nor does represent importance of attributes as weights. Some researchers are exploring the different impacts of social media platforms like FaceBook and LinkedIn on recruitment, not only to gather more information about candidates but for issues such as getting personality signals [3], [9] and impact on the workplace [11]. The approach of using scoring functions to evaluate objects or situations is quite common in practice; e.g., disease severity score for a patient [13] and credit risk score [1].

## 7 Conclusions

The TEAC system discussed here extracts information from free-form text resumes, and creates a shortlist for a given job description. The resume scoring function in TEAC is hand-crafted, hierarchical, and uses domain-knowledge from recruitment experts. As mentioned earlier, HR professionals often had somewhat different evaluation criteria for matching of a candidate with a JD and about measuring candidates' quality, and the candidate scoring function reported here is our attempt to consolidate the different subjective ways into a more objective automated scoring function. While we have described its application for candidates in the IT domain, this scoring function can be tailored for candidates in other domains. For example, when scoring non-IT candidates in the banking domain, we can remove the scoring component related to projects (since banking work is typically not project-oriented). We are working on devising a more general scoring function for non-IT candidates, which includes a component related to specific tasks. We also described a simple neural-network that automatically learns some of the weights used in the scoring function, based on feedback about whether the candidate was SELECTED or REJECTED. The TEAC system is deployed in a large multinational IT services organization. The ANN learns only some of the weights in the scoring function. We are working on extending the approach to automatically learn all the weights used in the scoring function. Firms specializing in recruitment usually have some software components to aid them, mostly focusing on generic search and retrieval of resumes. Our candidate scoring function is more transparent than such systems and is able to give concrete reasons for why a particular candidate is (or is not) included in the shortlist for a given JD. This function can also be customized in a limited way; e.g., by changing the weights.

## References

[1] Anderson, R.: The credit scoring toolkit: theory and practice for retail credit risk management and decision automation. Oxford University Press (2007)

[2] Borisyuk, F., Kenthapadi, K., Stein, D., Zhao, B.: Casmos: A framework for learning candidate selection models over structured queries and documents. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2016) 441–450

[3] Caers, R., Castelyns, V.: Linkedin and facebook in belgium: The influences and biases of social network sites in recruitment and selection procedures. Social Science Computer Review **29**(4) (2011) 437–448

[4] Faliagka, E., Iliadis, L., Karydis, I., Rigou, M., Sioutas, S., Tsakalidis, A., Tzimas, G.: On-line consistent ranking on e-recruitment: seeking the truth behind a well-formed cv. Artificial Intelligence Review **42**(3) (2014) 515–528

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

[6] Fazel-Zarandi, M., Fox, M.S.: Semantic matchmaking for job recruitment: an ontology-based hybrid approach. In: Proceedings of the 8th International Semantic Web Conference. Volume 525. (2009)

[7] Isson, J.P., Harriott, J.S.: People Analytics in the Era of Big Data: Changing the Way You Attract, Acquire, Develop, and Retain Talent. John Wiley & Sons (2016)

[8] Li, M.Y., Fan, Z.P., You, T.H.: Screening alternatives considering different evaluation index sets: A method based on soft set theory. Applied Soft Computing **64** (2018) 614 – 626

[9] Nikolaou, I.: Social networking web sites in job search and employee recruitment. International Journal of Selection and Assessment **22**(2) (2014) 179–189

[10] Patil, S., Palshikar, G., Srivastava, R., Das, I.: Learning to rank resumes. In: Working notes for the annual meeting of the Forum for Information Retrieval Evaluation (FIRE2012). (2012)

[11] Skeels, M.M., Grudin, J.: When social networks cross boundaries: A case study of workplace use of facebook and linkedin. In: Proceedings of the ACM 2009 International Conference on Supporting Group Work. (2009) 95–104

[12] Srivastava, R., Palshikar, G.K., Pawar, S.: Analytics for improving talent acquisition processes. In: Proc. 4th IIMA International Conference on Advanced Data Analysis, Business Analytics and Intelligence (ICADABAI 2015). (2015)

[13] Vincent, J., Moreno, R., Takala, J., Willatts, S., De Mendona, A., Bruining, H., Reinhart, C., Suter, P., Thijs, L.: The sofa (sepsis-related organ failure assessment) score to describe organ dysfunction/failure. Intensive care medicine **22**(7) (1996) 707–710