# Learning to Rank Resumes

Sangameshwar Patil, Girish K. Palshikar, Rajiv Srivastava, Indrajit Das

Tata Research Development and Design Center (TRDDC)
Tata Consultancy Services, 54-B, Hadapsar I.E., Pune, 411013, India

{sangameshwar.patil, gk.palshikar, rajiv.srivastava, indrajit.das}@tcs.com

## ABSTRACT

Recruiting and assigning right candidate for right job consumes significant time and efforts in an organization. Automated resume information retrieval systems are increasingly looked upon as the solution to this problem. In this paper, we focus on problem of learning an end-user specific ranking in such a resume search engine. We provide experimental results of SVMrank algorithm [2] for this task.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6 [**Learning**]: Concept Learning; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*User Profiles*

## Keywords

Learning to rank, Resume information retrieval

## 1. INTRODUCTION

Resumes contain vital information that aids the decision making process during recruiment in an organization. The project managers/HR regularly face the daunting task of selecting "right person for right job" from hundreds or even thousands of candidate resumes. Incorrect recruitment or task assignment decisions often have costly business impact.

In the simplest retrieval scenario, values for a set of structured attributes are extracted from each free-form textual resume; thus each resume is represented as a record in a table. Also, the job description can be specified as a structured condition ("query") over the same attributes. In more complex situations, the structured query itself could be supplemented in terms of a free-form textual job description.

Given a query, the search process retrieves and presents top K closely matching resumes to the end-user, who then manually filters and selects a subset (say, of size $M < K$) of these resumes for further processing (e.g. interviews). The selected resumes are not necessarily the top M resumes in the retrieved list. Some domain knowledge is used in making such a selection, which can often be explained in terms of background quality attributes. For example, the end-user may prefer persons with higher academic qualifications, higher marks, or those with additional certifications, or those who have won any awards etc. Thus the end-user implicitly re-ranks the K retrieved resume using an unknown ranking function. This ranking function may vary from user to user or may depend on the nature of the recruitment target position. The question that we address here is: how can this unknown resume ranking function be learnt after observing a specific user's selections from the resumes retrieved for different queries.

## 2. RESUME RANKING PROBLEM

Let $A = \{A_1, A_2, \ldots, A_m\}$ be a set of $m$ structured attributes for resumes; let $DOM(A_i)$ denote the set of possible values for attribute $A_i$. Let $D = \{D_1, D_2, \ldots, D_N\}$ be the given set of $N$ resumes, where each $D_j$ is a m-tuple in $DOM(A_1) \times \cdots \times DOM(A_m)$. A basic query has the form $A_i \in X_i$ where $X_i \subseteq DOM(A_i)$. A query is a conjunction of basic queries such that each attribute in $A$ occurs in at most one basic query. Let $K$ and $M \leq K$ be two given positive integer constants. Let $S$ denote the non-empty set of at most $K$ resumes retrieved from $D$ for the given query $Q$. A ranking function $f$ returns an ordered subset $L \subseteq S$ of size at most M, from the given set S of at most K resumes. Let $\{Q_1, Q_2, \ldots, Q_n\}$ denote a given set of n queries, along with the sets $S_i$ of retrieved resumes for each query $Q_i$. Let $L_1, \ldots, L_n (L_i \subseteq S_i)$ be the corresponding ordered subsets of the retrieved resumes, as ranked by the user. The ranking function $f$ used by the user is assumed to be fixed but unknown and needs to be estimated using this training data.

Learning such ranking functions is an active area of research [3]. In this paper, we report results obtained by a well known learning algorithm SVMrank [1] to learn a user-specific resume ranking function. SVMrank is a pairwise ranking method which learns an unbiased linear classification rule. Let $D_i \succ D_j$ denote that the user prefers $D_i$ over $D_j$. Training data for SVMrank contains set $R = \{(D_1, r_1), (D_2, r_2), \ldots, (D_M, r_M)\}$ where $r_i$ is the ranking of resume (or document) $D_i$ i.e. if $D_i \succ D_j$ then $r_i < r_j$. From a given ordering of the result-set per query, a set of pair-wise constraints are generated. Let $\xi_{ij}$ denote the slack variables in soft-margin formulation of SVM. Goal for SVMrank is to learn a ranking function $\hat{f}$ minimizes the objective function: $L_1(\mathbf{w}, \xi_{ij}) = \frac{1}{2}\mathbf{w}.\mathbf{w} + C \sum \xi_{ij}$ subject to constraints: $\forall (D_i, D_j) : r_i < r_j \mathbf{w}.D_i \geq \mathbf{w}.D_j + 1 - \xi_{ij}$ and $\xi_{ij} \geq 0 \ \forall (i, j)$.

## 3. EXPERIMENTS

Using the resume information extration engine built in-house, we created a repository of more than 2000 resumes of IT professionals. Each resume is represented using a standardized set of 28 features such as work experience, top 3 technology skills, top 3 business domains, number of job changes done, average job duration with a company, number of professional certifications (e.g. CCNA, SCJP etc.), academic qualification, average marks obtained and so on. We then created a benchmark set of 396 queries representing the typical information needs of HR and workforce management teams. Some of the sample queries are: $Q_1$ = (Java professionals with 0-2 years of experience), $Q_2$ = (Computer Science graduates with Java and Oracle skills and at least 2 years of project management experience).

Depending on the target role/position for a query, the end-users (e.g. HR personnel, Project managers) will use additional domain knowledge and/or prior experience to form a preference list from the result-set of a query. For instance, an HR manager may give more preference to the average percentage marks received in college and the candidates from top-tier colleges, but a seasoned project manager may prefer candidates who have not done frequent job changes.

In case of *learning to rank* problems, the relevance judgements are typically done by human assessors. However, the feature of *learning to rank resumes* is still in-process of being added to our resume information extraction tool. Hence the human judgement data regarding relevance is not available currently. To model the end-user specific ranking behavior, we used following three reprentative functions $(f_1, f_2, f_3)$ as approximations of three end-user categories. These would be used to simulate the human assessors and obtain preference lists for experimental verification.
$f_1 = 0.4\,csGrad + 0.8\,experience - 0.7\,nJobChanges$,
$f_2 = (0.4\,(experience^2 + avgMarks))/(nJobChanges + 1)$,
$f3 =$ if (technology is searched) then
(0.75 nRelevantCertifications + 0.25 nOtherCertifications)
else (0.25 nOtherCertifications);

Note that the constants (e.g. 0.4, 0.8 etc.) used in definitions of $f_1, f_2, f_3$ are chosen to indicate relative importance given to a specific attribute vis-a-vis other attributes in a resume. One could try out different values of such constants. Further, we also assume that there is already a pre-specified database of top-tier colleges, desirable certifications for a technology etc. available to the end-users.

We evaluated SVMrank algorithm using 10-fold cross validation. For training, we randomly selected 80% queries from the query set. The result-set for each query was ranked using the ranking functions $f_1, f_2, f_3$ described above. Top 20 resumes from this ranked set per query were given to SVMrank as training data. The evaluation was done using the remaining 20% queries from the query set. The evaluation procedure is outlined in the Figure 1. To measure how well SVMrank has learned the original ranking function $f_i$, we use Jaccard coefficient (JC) and Kendall's $\tau$ rank correlation coefficient. JC between two sets $A, B$ is defined as $J(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|}$ and it measures the similarity between A and B without regard to any ordering of the elements. Kendall's $\tau$ takes into account the ordering of the elements within A and B. It is defined as $\tau = \frac{\text{(number of concordant pairs - number of discordant pairs)}}{\frac{1}{2}n(n-1)}$. We compare the set of top-M resumes as ranked by SVMrank with the original list of top-M resumes by each $f_i$. Figure 2 summarizes the experimental evaluation. As we increase $M$, the number of top-M resumes used for comparison, the Kendall's $\tau$ increases with marginal decrease in JC. For $f_1$, the relatively low values of $\tau$ and JC are obtained because of relatively large number of ties that due to similar ranking scores assigned by $f_1$ to resumes. The reasonably high values of both measures (for $f_2$ and $f_3$) give us the confidence that SVMrank is able to identify and rank most of the resumes in the top-M list as per original ordering.

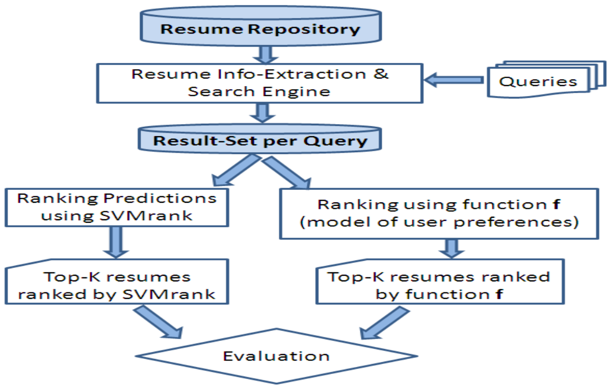
Resume Repository
Resume Info-Extraction & Search Engine
Queries
Result-Set per Query
Ranking Predictions using SVMrank
Ranking using function f (model of user preferences)
Top-K resumes ranked by SVMrank
Top-K resumes ranked by function f
Evaluation

**Figure 1: Experiment evaluation setup**

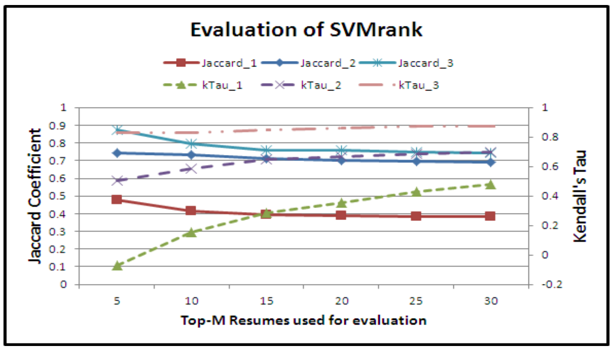

**Figure 2: Experimental Results**

## 4. CONCLUSIONS AND FURTHER WORK

In this paper we have identified the important problem with significant business impact: learning to rank the resumes. Experiments using approximate models of human relevance judgement showed that the SVMrank algorithm is able to get predict and rank the set of top-M resumes with good accuracy. Experiments with real-life relevance judgements from humans and comparison with other algorithms for *learning to rank* problems is planned as future work.

## 5. REFERENCES

[1] T. Joachims. Svmrank implementation,`http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`.

[2] T. Joachims. Training linear svms in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2006.

[3] T. Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(33):225–331, 2009.