Learning from Internal Mobility: A Machine Learning Approach to Augmenting Human Capital

Nikhil George, Ramayya Krishnan, Rahul Telang

Abstract

In companies today, work is organized in fluid teams whose composition changes as certain (new) skills become important while others become less relevant. Employees in-turn are primarily responsible for navigating their careers through positions that match their expertise and ambition. As retention and advancement of good employees is vital, many organizations provide employees the opportunity to apply for new jobs listed in the organization. Facilitating and supporting this internal mobility is a novel matching problem faced by many firms. We partnered with the IT division of a large financial services firm to develop an algorithmic model of an (internal) applicant's probability of selection to a new position. We show that the informational content in the applicant's current job's description and the firm's past selection decisions contain valuable signal to predict selection of internal applicants. The model enables the firm to identify cases where the applying employee's chances of selection was predictably low. This allows the firm to design training programs that are tailored to improve employees' selection prospects to their observed choices among new positions. We verify that these training opportunities are consistent with the firm's incentives and summarize them to an interpretable set via clustering. Our machine learning approach integrates three different sources of information available to firms, to identify training opportunities directly linked to new positions and observed employee choices.

Keywords: People Analytics, Internal Mobility, IT Talent Management

I. Introduction

Internal Mobility and HR Analytics

Firms today operate in a complex environment that requires them to update their systems, processes and even strategies regularly. A consequence is that work is organized in relatively fluid teams whose composition changes from time-time as new skills become important and some others less relevant. Employees do not face a traditional career ladder and are instead primarily responsible for navigating their careers through positions that match their expertise and ambition. To retain and progress employees, firms allow them to apply to new positions generated across the firm. Several large firms have internal job boards and encourage employees to seek out new positions within the firm. The efficacy of this internal matching process is today an important part of the firm's talent management strategy and naturally brings up how data analytics can help? The matching process can be improved in two ways. The first is by addressing some frictions or inefficiencies in the current process like identifying employees who would both like and fit a new position but does not apply. The second route for improvement, is by supporting employees to move to their desired roles with focused training support. Developing a widely applicable data analytic method to inform the latter, is the focus of this paper.

Many large firms today have data science teams focused on HR (Chamorro-Premuzic and Bailie 2020) functions, which has seen the emergence of terms like people analytics and people research to describe HR and its sub-functions. The emergence of HR analytics within organizations follows a transformation in how prospective employees seek opportunities and how firms advertise to prospective employees. Today, professional networking websites, job portals and employer review websites all play an important role in matching employees to jobs -- platforms that are powered by data and algorithms. A relative success in the area of HR analytics is also part of the recruitment process -- the adoption of pre-employment screening systems. Several large firms use software to screen which applicants to interview. In this case the organization has i) past data on which resumes got selected and ii) a well specified task – order resumes by selection probability. These conditions make applying machine learning to improve processes relatively easy (Gentzkow 2018). The structure of this problem has similarities to functions elsewhere in the firm where

machine learning has been successfully adopted. For e.g., marketing functions like pricing and product recommendations that are today delegated to algorithms (trained on data harnessed from past choices), have the same structure. However, as Tambe, Cappelli and Yakubovich (2019) highlight, most problems faced by an analytics team in HR do not have this structure. The requirement in HR analytics is not one of automating an existing process, but generating actionable insight from the data generated by the different processes within the firm. An aspect which makes HR analytics problems challenging.

An internal job board comprises descriptions of all posts newly created in the firm. When it is linked to administrative data that tracks employee applications to the listed positions and whether it resulted in selection, we have the HR data on internal mobility. Our objective to identify employee training opportunities that will improve selection prospects to desired jobs, falls in the challenging category of HR analytics problems. Kleinberg, Ludwig, et al. (2015) brought attention to a class of problems termed 'prediction policy problems', where the goal is to improve decisions and processes by (machine) learning from data, like in our case. The framework involves formulating a prediction problem that can be empirically tackled using the data available. The prediction, then becomes a previously unavailable input for policy formulation. In some settings, the new information allows the decision-maker to make better choices. More often, the previously unavailable model predictions expand the set of actions available to the decision-maker (firm management here). In the latter case, two parts of the problem require analytical attention. Formulating and implementing the prediction model requires creative application of algorithms to data. Turning the model predictions to actionable insight is a second part, which also require attention. This is the notable difference between resume screening and the problem considered here. In the former, once we can predict probability of selection, mapping it to action (select top k) is straightforward.

Paper Contribution

We present a machine learning approach to identify training opportunities from internal mobility data that is commonly available in most large firms. Like the prediction policy problems discussed above, we first build a prediction model. We build a machine learning model of the firm's selection decision of an internal applicant. The model accesses the job description of the new position and

the job description of the applicant's current position (in the firm) to estimate their probability of selection. We show on a validation set, that the applicant's current job has valuable informative signal for predicting their selection to a new position. Our predictive model is built on two sources of information – the information contained in the job descriptions including technical skills and functional responsibilities and the information in firm's past selection decisions. The latter allows us to not just match the competencies of the current and new (sought) job but also learn weights so that mismatches on important features contribute more to dissimilarity than mismatches on less important features.

With the ability to estimate an applicant's probability of selection, we can identify in the mobility data, employee applications to new positions where the candidate has a predictably low probability of selection. These cases are informative – they are observed employee choices to new positions and there is a shortcoming in their skills (assessed by the model) for the desired post. A training program to bridge the gap and improve selection prospects to similar positions, should be in the employee's interest and also contribute to the efficacy of matching employees to new positions in the firm. We additionally verify that employees applying to new positions with predictably low probability of success have few new posts in the firm where they have a high success. So, a training strategy based on employees' exploratory applications to new positions is not targeting employees whose current skills are in demand within the firm. The training paths identified are summarized to a small interpretable set for management consumption via clustering. In our setting, we assess close to 10% of the total workforce as potential beneficiaries from the eight broad paths of skill training identified. In sum, we present a machine learning framework that allows firms to tap their internal mobility data to support and augment human capital, consistent with their observed choices. In doing so, we also convey that most firms readily have the data for impactful HR analytics.

The structure of the paper is as follows. We describe the setting, data, and analytical framework in section II. Section III presents the details of our prediction model. In section IV, we illustrate how we identify up-skilling opportunities in the firm and discuss their scope, before concluding with a summary and discussion in section V.

II. Context, Data and Framework

II.A. Context

Our setting is the Information Technology division of a large financial services institution. The division is organized as multiple units and subunits that cater to the IT needs of a large corporation. There are units who cater to the requirements of building and supporting IT systems that serve different line functions of the business -- say trading, wealth management or credit. Other units are constituted to work across business functions to serve objectives like retiring old systems or generating reports for regulatory compliance. Our area of focus is employees who constitute these teams and are categorized as individual contributors by the firm. These are roles requiring knowledge of specific technical skills and/or processes and whose primary job function is not managing other staff members.

With units engaged in building new or improving existing systems, they often require individual contributors with previously unavailable skills. The organization expects new positions to require technical skills that are in high demand in the external labor market making hiring there costly. The organization looks to support internal mobility that allows employees to find new positions matching their ambitions and competencies. When a unit require the services of a new staff member, they generate a job description which is posted to an internal job board. The job board is accessible to all staff members. A staff member who believes their skills and experience would be better utilized in an advertised new position may apply to be considered.

The organizational policy supports internal moves in principle -- barring few practical restrictions, all employees are eligible to apply against a listed position. They do not require seeking the permission of their reporting manager for applying to the new position advertised within the firm. Internal applications received are reviewed by the recruiting unit (a screening step) and subsequently interviews are conducted to select an applicant. The job is advertised externally only after an internal candidate is not found in a desired time window. The role of HR in the recruitment process is limited to ensuring compliance with organizational policies and does not extend to inputs in the actual selection decision. The HR leadership wants to improve internal mobility levels

in the organization and is willing to undertake initiatives regarding the same. They want to tap the readily available organizational data to draw insight that is relevant to supporting their goals.

II.B. Data

We obtain information about the new positions generated in the firm between 2018 and 2021. Along with the date of posting and the job description text we observe the following about the applications received against each posting – i) We observe whether an application received against the listing was from an internal or external candidate, ii) whether they were offered the position or not and iii) their employee id if available[1]. We observe close to 8000 new positions for individual contributors that were generated in the organization. For clarity, we have illustrated the data available in Figure below. The informative part of the data is simply job descriptions of the positions generated. If the position received an internal application against it, then we know the applying employee's ID (see last column in Figure) and the outcome of the application. Notably, we do not observe any additional details (like the applicant's resume) although it could be available to organizations in other settings. The data available to us, we believe should be easily available to most firms with similarly organized internal labor markets or can be collected from their systems relatively easily.

The ability to track employees through this corpus of applications allows to match the job description of an employee's current job to the description of a future position sought by them. An example is illustrated in Figure 2. Employee **XA321** was selected to and joined position JD 332 on 1/1/2019. We then observe they applied to JD 6345 on 2/3/21 which was not offered and subsequently to JD 7801 which was offered. This allows us to match the job description of **XA321**'s position at the time of application (JD 332) to the job descriptions of the positions they applied to. In short, a job description corpus when matched to employees, provide information of employee flow and the ability to view descriptions of their source and destinations.

[1] An employee id is available against an internal applicant. It is also available against an external applicant who was selected to the position and subsequently issued an employee id on joining the firm.

Date	Job Description	App. No.	Int/Ext	Sel./Rej.	Emp. ID
1/1/2019	JD 1	1	I	S	XA32I
1/1/2019	JD 1	2	I	R	XD42I
1/2/2019	JD 2	1	E	S	YK31J
1/2/2019	JD 2	2	E	R	-
1/2/2019	JD 2	3	E	R	-
2/3/2019	JD 3	1	I	R	HJ67J
2/3/2019	JD 3	2	I	R	FG32M
2/3/2019	JD 3	3	E	R	-
2/3/2019	JD 3	4	E	S	JH56L
.					
.					
.					
3/1/2021	JD 8000	1	E	S	GK86B
3/2/2021	JD 8000	2	E	R	-

Figure 1: Data on new positions generated in the organization[2]

Date	Job Description	App. No.	Int/Ext	Sel./Rej.	Emp. ID
1/1/2019	JD 332	1	E	S	**XA32I**
1/1/2019	JD 332	2	I	R	XD42I
.					
.					
.					
2/3/2021	JD 6345	1	I	R	**XA32I**
2/3/2021	JD 6345	2	I	R	FG32M
2/3/2021	JD 6345	3	E	R	-
2/3/2021	JD 6345	4	E	S	JH56L
.					
.					
.					
3/1/2021	JD 7801	1	I	S	**XA32I**
3/1/2021	JD 7801	2	I	R	FG21K

Emp. ID	Curr. Job Lstg	Sought Job Lstg	Current JD	Sought JD	Sel./Rej.
XA32I	1/1/2019	2/3/2021	JD 332	JD 6345	R
XA32I	1/1/2019	3/1/2021	JD 332	JD 7801	S
.					
.					

Figure2: Extracting Employee Flow Information

[2] JD 1, JD 2, etc. are placeholders for job descriptions who have a modal length of above 500 words.

II.C. Framework

The data available comprises text descriptions of jobs and information on internal mobility decisions. Below we discuss, the informational content in the data in a decision-making framework. Let J: $\{j_i\}$ be the set comprising job descriptions (discussed previously). As a first step, we discuss the evaluation of a single internal applicant against a position. Let j_s be the job description of the sought position – a text document that lists the competencies required for the job, along with other relevant information. In our model, a job is characterized by a competency vector $X \in \mathbb{R}^l$ drawn from a fixed distribution over X. The decision-maker (say, the hiring manager) maps j_s to X_s. Here $l = \dim(X)$ is the count of all competencies required by positions in J. The hiring manager then turns to determine the competency vector of the applying candidate. The manager makes an assessment based on the applicant's current job - j_c and what is learned in a direct assessment, say through an interview. Let the competency vector of the applicant so assessed be $X_c^{\sim}$; where $X_c^{\sim} = f(X_c, Z)$ some function of the competency vector of their current job (X_c) and what is observed in the interview (Z). The decision maker computes a distance $d(X_s, X_c^{\sim})$ between the sought job and the assessed competency vector. The applicant is offered the position when the distance is less than some threshold τ. We can express the decision as follows:

$$S(j_s, (j_c, Z)) = \begin{cases} 1, & d(X_s, X_c^{\sim}) < \tau \\ 0, & d(X_s, X_c^{\sim}) \geq \tau \end{cases}$$

The decision regarding the applicant seeking position j_s and currently working in j_c is illustrated in Figure below. When there are multiple internal applicants, the offer is made to the candidate assessed to have the lowest competency distance to the job. The analyst's task is to approximate the competency distance of an internal candidate, given the sought job description. They have access to J the corpus and D: $\{(j_s,\ j_c,\ S)\}$ the set comprising past decisions (selection data). Note here that the analyst observes the sought and current job descriptions, but not Z; which was available to the decision maker. The analyst observes S the selection decision, but not all inputs that went into the decision.

With the goal to approximate $d(X_s, X_c^{\sim})$, a machine learning task is to learn a function that maps a job description text to a competency vector ($r: J \rightarrow \hat{X} \subset \mathbb{R}^l$)[3]. For a given new position j_s and an internal applicant working in job j_c, the learnt function allows to obtain $\widehat{X_s}, \widehat{X_c}$, stand-ins for the true competency vectors. The arguments in the decision maker's distance function use $(X_s, X_c^{\sim})$; using $(\widehat{X_s}, \widehat{X_c})$ instead brings two types of errors. The first may be thought of as measurement error in obtaining $\hat{X}$ from the job description text. The second source of error is due to factors unobservable to the analyst. While the decision maker has access to additional information, we expect them to attribute considerable weight to the current job competencies X_c, when formulating $X_c^{\sim}$, a candidate's competency vector. Competencies captured in X_c are observed by the firm in a professional setting within the firm, while additional information acquired via (say) interview has its limitations and is likely to be weighed accordingly in $X_c^{\sim}$.

This brings us to learn a distance function $\hat{d}$: $\mathbb{R}^l * \mathbb{R}^l \rightarrow \mathbb{R}_{\geq 0}$. While a standard distance metric can be used to compute the distance between $(\widehat{X_s}, \widehat{X_c})$, the setting provides valuable information that can be utilized. We have the set D comprising past decisions. Let D^0, D^1 be the sub-sets of D where the former comprises cases of rejection ($S = 0$) and the latter comprises cases of selection ($S = 1$)[4]. Instead of relying on a metric that equally weighs all features, we can learn a distance function that assigns weights such that the distance computed for the cases in D^0 where the applicant was rejected is higher than the distance computed for cases in D^1. A suitable optimization problem can be formed to learn the distance function. As we work with $(\widehat{X_s}, \widehat{X_c})$ and not the true competency vectors, the gains from learning a distance function from data (compared to using a standard metric) must be validated before adoption.

With the distance function and empirical counterparts for the feature vectors in place. We know from the setup that the probability of selection $P_{S=1}$ is a decreasing two-step function of $d(.)$ (see part A in Figure below). As the analyst, we would like $P_{S=1}$ to be a decreasing in function in $\hat{d}(\hat{.})$

[3] The following is a more accurate notation $r: J \rightarrow \hat{X} \subset \mathbb{R}^{l'}$. Here $\mathbb{R}^{l'}$ conveys that the dimension of $\hat{X}$ is picked as part of the empirical analysis (training) and is different from $dim(X) = l$.

[4] $D^0 \subset D|(j_s, j_c, S = 0)$ and $D^1 \subset D|(j_s, j_c, S = 1)$.

-- our approximation (like illustrated in part B of figure 3 below). The figure illustrates the gains from the ability to compute $\hat{d}(\hat{.})$. Before computing the distance measure, the firm had a single estimate for $P_{S=1}$, say the proportion of applications that resulted in selections (indicated by the horizontal grey line in figure 3 B). With held out data we can now estimate $\widehat{P_{S=1}}(\hat{d})$ – an empirical distribution of the probability of selection, based on observables -- j_s and j_c.

For the analyst, the success of the exercise to approximate $d(.)$ is illustrated by the range of $\widehat{P_{S=1}}(\hat{d})$ (indicated in figure 3 B by the grey right bracket). If the estimated distribution is flat, then the information contained in job descriptions is either inadequate (the selection is largely based on unobserved factors) or is not captured effectively in $\hat{d}(\hat{.})$ (high measurement error). However, if it is decreasing as illustrated by the black dashed line, then the exercise is of value. It indicates that the observable signal in the selection decision can be captured by the empirical apparatus in place.

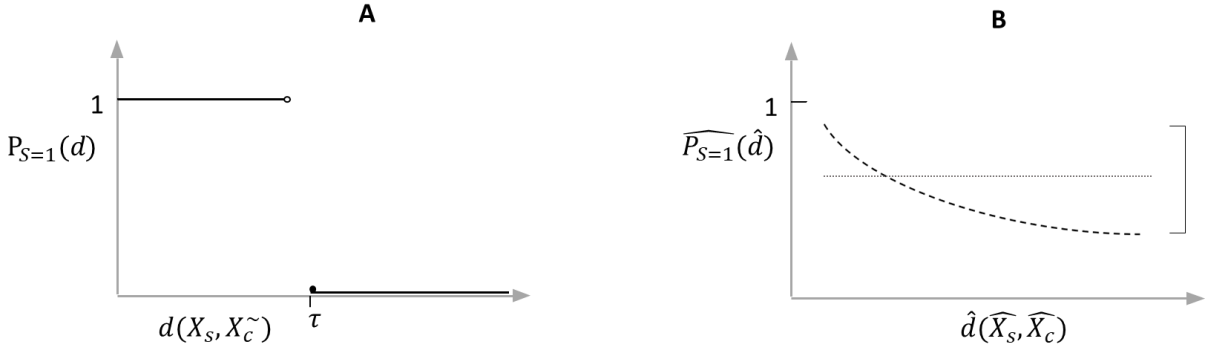

Figure 3 A. Job-candidate competency distance as assessed by firm. B. Competency distance (approximated) between job and applicant's current job.

It is useful to reiterate the role of selection data (set D) available here to the analyst. This information is not used to learn the vector representation of job descriptions. Which means we can approximate distances between job descriptions (say, using a standard metric) without using the selection data. However, the predictive value of the signal ($\widehat{X_c}$) contained in the applicant's current job description can be validated only by linking the distance measure to selection data. In sum, the

machine learning apparatus computes a distance from observed data, which in-turn is mapped to a probability of selection conditional on applying.

While we can extract valuable information that predicts selection, we can't make much progress on analyzing selection decisions. If the setting provided more information – additional features of applicants or post selection performance of (selected) applicants, it would open opportunities to study firm's selection decisions. However, the competency distance (approximated here) is also a key factor in an employee's application decision. It is safe to assume that employees seeking new positions make a dynamic choice, factoring in their expectations about future opportunities, the cost of acquiring new competencies and the probability of selection, on applying. Applicant working in job j_c assess their probability of selection based on the requirements listed in j_s. Let $d_c(j_s, j_c)$ be the applicant's assessment of their distance to j_s. We assume the applicant's assessment ($d_c(j_s, j_c)$) is correlated with $\hat{d}(\widehat{X_s}, \widehat{X_c})$ -- the competency distance approximated by the analyst.

Note that $\hat{d}(\widehat{X_s}, \widehat{X_c})$'s ability to assess fit is validated, the correlation assumption allows the analyst to assess applicants' choices using the approximated competency distance. The core of the assumption is rather innocuous -- that the applicant has skill in ranking their fit to different positions available. The analyst observes the applicant's choice j_s and their choice set C – which comprises the descriptions of contemporaneous listings in the firm. They can assess the applicant's choice set – rank order the constituent jobs by predicted probability of selection. The assumption conveys that, the applicant's assessment (rank-ordering) which is unobserved, is consistent with analyst's ordering based on $\hat{d}(\hat{\cdot})$. For example, when the applicant is observed applying to a position with low probability of selection over others with better chance of success, we assume this ordering to be consistent with the applicant's own assessment and not a choice due to an erroneous assessment. Note that the assumption extends only to the applicant's assessment and not to their choice. The framework respects unobservable factors in both selection and application decisions and do not impute outcomes. Observed selection decisions are used in

computing $\widehat{P_{S=1}}(\hat{d})$. An internal applicant's (observed) choice is evaluated against this validated estimate.

III. Machine Learning Pipeline

III.A. A Model of Job Descriptions

We first take up the practical task of learning $r: J \rightarrow \hat{X}$, the function that maps a job description to its (approximate) competency vector. Henceforth, we use the term competency vector to refer to the approximation $\hat{X}$, unless specified explicitly. A job description j is a text document that describes tasks to be performed on the job and the technical skills (if any) required to perform them. In our setting, the average job description is about 550 words. Our objective is to capture the most important informational content in the text, the competencies required of an employee working in that position. We describe here a simple model of job description.

We model the tasks listed in a job description as being primarily associated with a function. We assess from a supplementary dataset that a high fraction of employees self-report their functional role as belonging to one of 'developer', 'tester', 'support'. These are well known roles in any IT division. In a job j, the envisaged role could be a mix of these functions and the tasks featured in it generated accordingly. Technical skills form the other significant part of a job description. In our model, the technical skills listed in a job description are generated by latent technical competencies. For clarity, take the case where a job requires the employee to be competent in 'web-frameworks'. This requirement may be represented in j by words like 'web framework', 'Spring', 'MVC' etc. Unlike in the functional case, we leave the competencies as latent and look to learn it form the data. In summary, the key information in a job description is generated by two distributions-- the first a mixture over the three functional roles that determines the tasks and responsibilities described; the second a mixture over k latent technical competencies that determine the technical skills. We illustrate the model in Figure 4 A below.

The model brings us two advantages. It facilitates our need to map the lengthy text to a vector of small dimension. Inferring the underlying functional and technical competency mixtures in j allows us to represent this information in a vector ($\hat{X}$, the competency vector) whose dimension is smaller than the length of text. The model also allows to incorporate our knowledge of the content and structure of IT job descriptions into the process that learns the competency vector. This brings up the discussion on inference algorithms. Algorithms that will learn from the corpus, some global characteristics that allows us to elicit the latent features—functional distribution and technical competencies from each individual job description. In the following discussion we reverse the order and first discuss inferring technical competencies before discussing the algorithm to obtain the functional distribution from tasks and responsibilities.

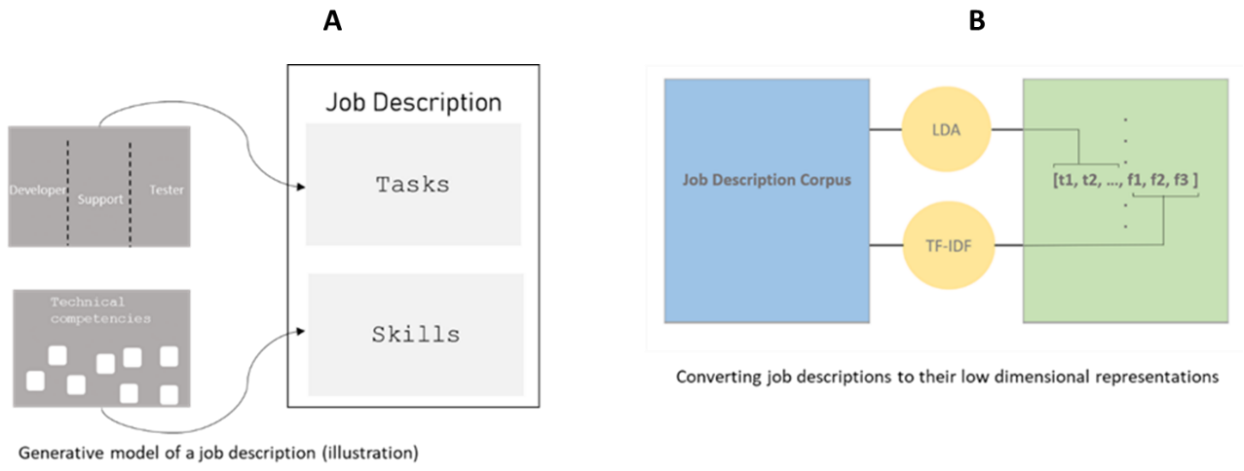

Figure 4: A. Illustration of the model of job description. B. Illustration of the constituent algorithms that convert a job description to a competency vector.

Topic Modeling (LDA)

While we have a conceptual idea that skill words in a job description is determined by a distribution over a latent set of competencies, to infer this distribution from a job description we require a fully specified probability model of how they are generated. The LDA (Latent Dirichlet Allocation) algorithm (Blei, Ng and Jordan 2003) is an inference algorithm based on a probability model for

word generation in a document. In LDA, the words that constitute a document is determined by the underlying topics covered in the document. In our context, the job listing is the document and the technology skills listed in them the words. Given a sufficiently large corpus of job descriptions, the LDA algorithm allows to estimate a topic mixture (distribution) for each listing.

The LDA algorithm assumes the following model of how the documents are generated[5]. Any job $j \in J$ can be represented as a mixture of k topics. All skill related keywords in the corpus J come from a fixed corpus of size v, the cardinality of the vocabulary. Make k draws from the Dirichlet distribution parameterized by β_v where $dim(\beta_v) = v$. Each of these draws give a Multinomial distribution with parameter vector ϕ_z, where $dim(\phi_z) = k$. Each of these distributions, correspond to word distributions (skill words) for each topic (competency). A job requirement j, is generated with a draw from a second Dirichlet distribution. This distribution is parameterized by α_k, where $dim(\alpha_k) = k$. The draw from this, gives a Multinomial distribution with parameter vector – θ, which corresponds to the competency distribution in the document. Recall that the k draws have already provided the skill word distributions for each competency.

Each skill word s in j is generated as follows:

Draw a competency z, according to θ; $z \sim Mult(\theta)$

Draw a skill word s, according to ϕ_z; $s \sim Mult(\phi_z)$

The training of the model involves estimating corpus level parameters. These include α_k the Dirichlet that determines the competency mixtures in each job, and ϕ_z which represents the k draws from $Dir(\beta_v)$. A desirable α_k would mean job descriptions are a mixture of few (2-3) competencies rather than several, which would make distinguishing jobs by this representation hard. With the corpus level parameters estimated, given $\{s\}$, the technical skills in j, we can obtain an estimate for θ the underlying competency distribution.

[5] As the algorithm's model and inference is well elucidated in (Blei, Ng and Jordan 2003) our explanation is brief.

The model was trained using Gensim, a popular open-source library for natural language processing in Python. The practical aspects in our setting include drawing a list of technical keywords to extract the skill related words from each j in our corpus[6]. The list comprised v = 287 words[7]. To train an LDA model we need to specify k, the count of latent competencies. We pick $k = 14$. The key evaluation of the LDA model is inspection. The ability of the model to learn meaningful latent topics. We present the 5-6 highest probability skill words associated with each competency, in Table 1 below. We have assigned a name to each topic for convenience.

Table 1: Top skill words for each competency

Topic	Topic – Keyword distribution	Assigned name
0	Java, Spring, Webservices, WebSphere, SQL, J2ee, …	Java-Webservices
1	Hadoop, Hive, Spark, Scala, Python, Scoop, Cloudera, …	Hadoop developer
2	Python, Java, Quartz, Multi-threading, Linux/Unix, …	Python developer
3	SQL, ASP.net, MVC, WCF, Webservices, C#, …	.Net - Webservices
4	Linux/Unix, Python, Java, C++, Perl, Devops, …	Devops
5	DB2, Mainframe, JCL, COBOL, CICS, VSAM, …	Mainframe
6	SOAP, Webservices, Mule, REST, Json, XML, …	API Handling
7	Pega, PRPC, RPA, CI/CD, Openspan, SOAP,…	Process Automation
8	SQL, Oracle, Linux/Unix, Informatica, Teradata, …	Database - ETL
9	Excel, SharePoint, Incident, ITIL, Management tools,..	Management
10	Java, J2ee, Spring, SQL, Hibernate, Oracle, …	Java
11	Linux/Unix, SQL, Windows, Java, Shell, Oracle, …	Sys. Admin
12	Testing, Selenium, QA, QC, UFT, Cucumber, …	Testing
13	Javascript, HTML, Angular, Jquery, AJAX, CSS,…	Front End - JS

[6] We parsed the section of job description that listed the key technical skills required by the job and extracted the nouns in them using a dependency parser. The set of extracted nouns from all descriptions in the job description corpus formed the core of our list of technical keywords.

[7] The list was then adjusted to handle quirks in how these keywords are expressed in job descriptions – for example ‘Javascript’ and ‘Java Script’ or ‘Linux/Unix’ and ‘Unix/Linux’ convey the same skill.

Screening for words that predict functional roles

We can now obtain a 14-dimensional vector from $\{s\}$ the skill words in j. We turn attention to the task words in a job description. As discussed earlier, we will follow a slightly different approach to summarize the functional role described in j, leveraging on our contextual knowledge. IT jobs may be readily classified into one of the following three functional roles -- developer, support and testing; roles with distinguishing tasks. In a large firm with projects at different stages of their lifecycle, some roles are likely to be a mixture of the three roles. Our goal is to infer this mixture from j. Our approach is based on the observation that, for a subset of job descriptions- $J' \subset J$, one of the phrases 'developer role', 'tester role', 'support role' is included within the first line of j. The phrase is easy to detect and conveys a clear indication of $j's$ envisaged functional role.

From J', we can isolate words that are most associated with tasks and responsibilities related to the three canonical functional roles. The procedure is intuitive. Let D, S, T be non-overlapping subsets of J', where the elements in D are job descriptions with the text label developer (role); S, T are analogously defined subsets. For all non-skill words $w \epsilon J'$ we compute $f_w^D = \frac{|\{jobs\ with\ w\ in\ D\}|}{|D|}$, the proportion of developer jobs containing w. Similarly, the proportion of support and test jobs containing w is computed. We now have a triplet $[f_w^D, f_w^S, f_w^T]$ for all $w \in J'$. With the proportions computed for all terms, we need to select words with the ability to predict the three functional roles. Set an upper and lower threshold (α^u, α^l) for picking out words that needs to be included in the (non-overlapping) sets $\widetilde{D}$, $\tilde{S}$, $\tilde{T}$. The sets comprise words that predict the functional roles of developer, support and testing respectively. A task word w that occurs in a large proportion of developer jobs and in a low proportion of support and testing jobs is included in $\widetilde{D}$[8].

$$\widetilde{D} = \{w: f_w^D > \alpha^u\} \cap \{w: f_w^S < \alpha^l\} \cap \{w: f_w^T < \alpha^l\}$$

The sets $\widetilde{S}$ and $\tilde{T}$ are composed analogously to $\widetilde{D}$. The upper and lower thresholds, like the number of latent competencies k are hyper-parameters that need to be tuned. Our method of isolating the terms is adapted to the context from the idea of marginal screening, a method to select variables in a high dimensional setting (Fan and Lv 2007). f_w^D is a measure of the strength of association

[8] Sets D˜, S˜, T˜ are enumerated in Appendix A.

between a word and the functional role of developer. This allows us to ignore words without predictive signal about the job's functional role. The approach is also similar to the idea of TF-IDF (from the information retrieval literature -- where words appearing frequently in several documents are down-weighed).

Now $\forall\, j \in J$ we can extract the words belonging to $\tilde{D}$, $\tilde{S}$ and $\tilde{T}$ to compute their respective proportions in j. We now have a mapping from j to a triplet that can be interpreted as a probability distribution over the three functional roles of interest[9]. Our approach which relies only on words, discards the informational content in the structure and construction of the text in j. We additionally discard words without predictive signal for the function. With this, we gain simplicity and the ability to use our knowledge of IT jobs and set up a supervised screening of words. The approach can be adapted to extract key information encoded in job descriptions based on other useful criteria like supervision and management responsibilities if relevant label can be extracted easily from a subset of J. With the two algorithms, we now have our function $r: j \rightarrow \hat{X}$ that can map a job description to a competency vector of 17-dimensions. The first 14 dimensions contain information on the technical competency distribution, while the last three dimensions contain information on the functional distribution indicated by the tasks and responsibilities listed in j. Like we have a ready interpretation for the first 14 dimensions (see Table 1), the last three dimensions too have a ready interpretation as the probability mixture over [developer, test, support].

III.B. Distance Function Learning

We now come to the second of the machine learning tasks. We need a distance function, given a pair of competency vectors -- $\hat{d}(\widehat{X_s}, \widehat{X_c})$, which in-turn is predictive of whether the internal applicant would be selected on applying. With the vectors available, we can use one of many standard distance functions including Cosine distance or the Euclidean distance. However, as

[9] In forming the triplet, all screened terms are assigned the same weight here. As we are in a super-vised setting, we can learn a weighting for how the isolated term predicts the different roles, which could improve the representation

briefly discussed in the framework section, with set D: $\{(j_s,\ j_c,\ S)\}$ (which has the firm's selection decisions) available we can leverage it to learn a distance function between the competency vectors. The case for learning a custom distance function is that the standard measures assign equal weight to all the features when computing the distance. Mismatches on important features should contribute more to the distance than mismatches on less important features. As the features are inferred from text, they are determined by cooccurrence patterns of words. We want the competency distance to capture the difference in job related competencies and not merely the semantic difference, which determines the competency vectors $\widehat{X_s}$ and $\widehat{X_c}$.

Recall sets D^0, D^1 partitions of D where the former comprises cases of rejection and the latter comprises cases of selection. The true distance between sought job and candidate (as assessed) are higher in D^0 compared to cases in D^1. We abuse notation and from now refer to D^0, D^1 as sets where j_s, j_c from the original are replaced by $\widehat{X_s}$, $\widehat{X_c}$ their respective competency vectors. Our goal is to learn a distance function $\hat{d}$ such that most distances computed on cases in D^0 is larger than the distances computed on cases in D^1. We need to convert this idea to a suitable objective function. The optimization problem solved in Xing, et al. (2002) can be adopted to the problem. Verbally, the objective function is as follows: maximize the sum of distances in D^0, under the constraint that sum of squared distances between the pairs in D^1 is less than some constant c. Our idea of a distance function which will return greater distances for cases in D^0 is satisfied here. We state it more formally below.

The goal is to learn a distance metric of the form

$$\hat{d}(\widehat{X_s}, \widehat{X_c}) = \sqrt{(\widehat{X_s} - \widehat{X_c})^T M (\widehat{X_s} - \widehat{X_c})}$$

The distance is referred to as the Mahalanobis distance and $M \in S_+^d$ is a symmetric positive definite matrix. It is equivalent to Euclidean distance after linear projection $\hat{d}(\widehat{X_s}, \widehat{X_c}) = \sqrt{(L\widehat{X_s} - L\widehat{X_c})^T (L\widehat{X_s} - L\widehat{X_c})}$[10]. Our objective function is expressed as follows:

$$\max_{M \in S_+^d} \sum_{(\widehat{X_s}, \widehat{X_c} \in D^0)} \hat{d}\,(\widehat{X_s}, \widehat{X_c})$$

$$s.t. \sum_{(\widehat{X_s}, \widehat{X_c} \in D^1)} \hat{d}\,(\widehat{X_s}, \widehat{X_c}) \leq 1$$

Note that if M is an identity matrix, the distance would then simply be the Euclidean metric. If M is restricted to be a diagonal matrix, then it is a re-weighting of the axes. The PSD constraint ensures that $\hat{d}(\hat{\cdot})$ is non-negative and triangle inequality holds. Details of the information on the algorithm to solve the problem can be found in the original paper (Xing, et al. 2002). In our data we have 726 and 590 elements in sets D^0 and D^1 respectively. We take an 80% random sample to estimate M. The optimization problem is solved using the algorithm's implementation available in Scikit-learn, a popular and free library of machine learning algorithms in Python. With the matrix M available, we can compute the distance between two competency vectors weighted based on the selection decisions in the firm.

III.C. Validation

The machine learning pipeline is now in place. We can map a job description *j* to its competency vector $\hat{X}$ and have a distance function $\hat{d}(\hat{\cdot})$ to evaluate distance between competency vectors. Note that our interest is the ability of the distance measure to predict probability of selection $P_{S=1}$. In the sub-section we will take up two tasks. We first verify whether $\hat{d}(\hat{\cdot})$ has valuable signal of $P_{S=1}$, the primary objective of our machine learning apparatus. Then verify whether our choices in learning the competency vector $\hat{X}$ and $\hat{d}(\hat{\cdot})$ brought predictive gains. Specifically, we will check

[10] L is the lower triangular matrix with positive diagonal entries. We can obtain an estimate for L by the Cholesky decomposition of M.

whether including information on the functional role and whether learning a custom distance function brings predictive gains.

Recall from the framework, that our objective is to extract information from j_c, the applicant's current job which can predict their probability of selection (on applying) to the sought job. If our exercise gives a positive range of values for $\widehat{P_{S=1}}(\hat{d})$ (the estimated probability of selection, given competency distance) then the exercise is fruitful. On a held-out validation set V: $\{(\widehat{X_s}, \widehat{X_c}, S)\}$ with 254 observations we compute $\widehat{P_{S=1}}(\hat{d})$. We divide the competency distance values for cases in V into quintiles and compute $\widehat{P_{S=1}}$ for each quintile. The results are presented in Fig. 5 below. Note that the probability of selection is 0.66 in the first quintile and 0.3 in the last quintile. In other words, our machine learning pipeline can capture valuable signal that can predict probability of selection. In the absence of the ability to compute competency distances from job description texts, the analyst resorts to modeling selection as a Bernoulli variable to obtain an estimate of selection probability, which here is 0.4.

Another test popular in machine learning for assessing the accuracy of a binary classifier is the Area Under the Curve (AUC)[11]. We can evaluate the potential of $\hat{d}(\widehat{X_s}, \widehat{X_c})$ to classify whether an applicant is selected or not given (j_s, j_c). We compute the AUC curve on the cases in V, which is presented in Figure ZZ below. It returns an AUC score of 0.69. The AUC score is a U-statistic and can be interpreted in the following manner. Let V^0, V^1 be the sub-sets of V where the former comprises cases of rejection ($S = 0$) and the latter comprises cases of selection ($S = 1$). The AUC score is the probability that $\hat{d}(\hat{.})$ computed on a case drawn at random one from V^1 will (correctly) return a smaller value than the $\hat{d}(\hat{.})$ computed on a case drawn at random from V^0[12].

[11] An AUC score is the area under the Receiver Operating Characteristic (ROC) curve (see Figure 3), which in turn is a plot of the performance (true positive rate -TPR and false positive rate -FPR) of a binary classifier at all classification thresholds. The ROC may be thought of as a frontier and movement along it results in trading off either true positive rate or false positive rate.

[12] So an AUC score of 0.5 communicates a classifier with no predictive ability and score of 1 indicates perfect classification ability.

In computing the AUC, our objective is to communicate the predictive ability of $\hat{d}(\hat{\cdot})$ in a manner that is widely used and well understood. 0.69 is not a high AUC value for a binary classifier that used in practice. However, recall that our use of $\hat{d}(\hat{\cdot})$ is not to make selection decisions, but simply to predict selection decisions from the observables at hand. AUC scores reported in the literature rarely have this objective. Li, Raymond and Bergman (2022) predicts whether a candidate interviewed by a firm is selected from the candidate's resume; Kleinberg, Lakkaraju, et al. (2018) predict whether a bail applicant is released, based on recorded information available at time of bail hearing. Like our setting, the focus in these studies too is on the predictive signal in the observables and not designing a classifier to be used in practice. The studies report AUC scores of .64 and .70 respectively on their classifiers.

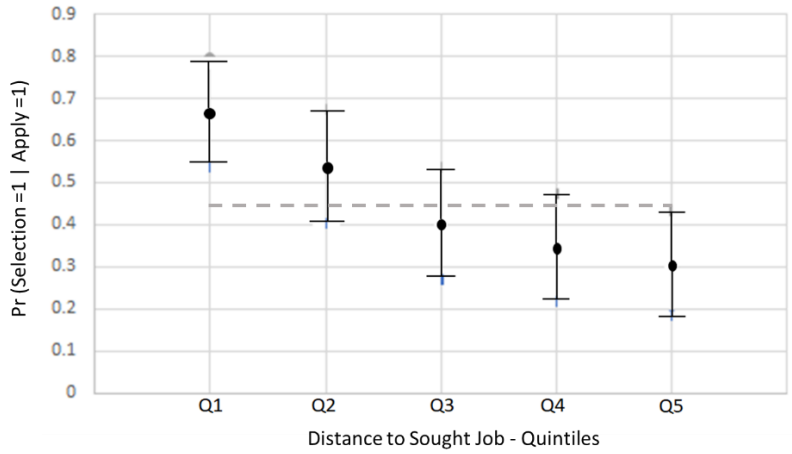

Figure 5: The plot shows the probability of selection (on applying), by quintile bins of competency distance to the sought job. The horizontal grey dashed line is the average probability of selection for the full sample. Estimates computed on the validation set.

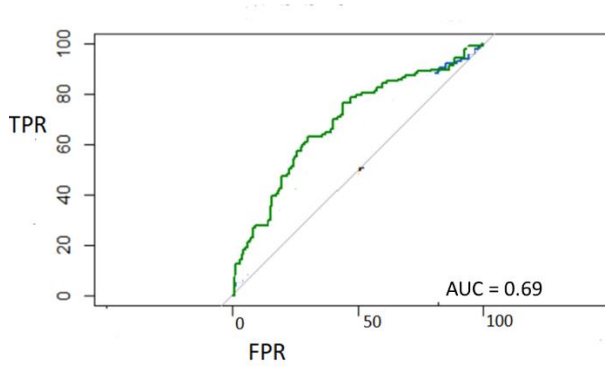

Figure 6: ROC Curve: Predicting selection of an internal applicant, conditional on applying

We will now turn to verifying whether two choices in our empirical approach i) to include function information in our competency vector and ii) learn a distance function instead of using a standard (Euclidean) metric, has brought predictive gains.

Let $r_1: j \rightarrow \widehat{X_{14}}$ and $r_2: j \rightarrow \widehat{X_{17}}$ be the functions that map j to competency vectors without function information and with function information respectively. Note that in our machine learning pipeline, learning $r: j \rightarrow \hat{X}$ comes before learning the distance function $\hat{d}(\hat{\cdot})$. So, when testing whether r_2 has better predictive ability, we use the Euclidean metric as the distance which predicts selection on application. The AUCs for the two classifiers are reported in Fig. 7 below.

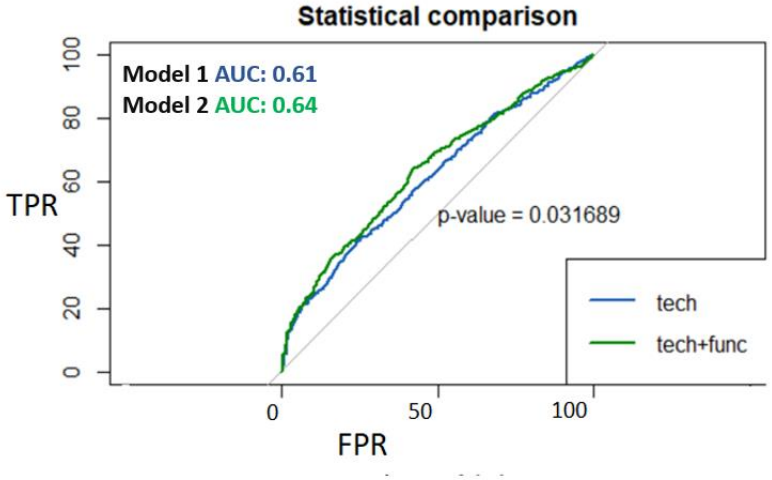

Figure 7: Comparison: AUC classifier1 (distance between 14 dim. Competency vectors – tech. skills only). AUC classifier2 (distance between 17 dim. Competency vectors). The p-value computed using the De Long test.

Including the functional information in the competency vector does improve its ability to predict selection. We use the DeLong test (DeLong, DeLong and Clarke-Pearson 1988) to verify whether the higher AUC score obtained here for competency distance computed on r_2 is statistically different[13]. The null distribution returns a z score of 2.15 ($p < 0.04$) suggesting the rejection of null.

We now take up the second verification question, whether we get predictive gains from using the learnt distance function in-lieu of a standard metric like the Euclidean metric. We already know the answer from the earlier exercises. We know that the learnt distance function returns an AUC of 0.69. From the preceding exercise, we know that using the Euclidean distance returns an AUC of 0.64. In Figure 8 below, we compare the two AUCs directly, i) A Euclidean distance-based classifier (blue) and ii) a learnt distance function-based classifier. The null hypothesis test that the metric learnt AUC and the Euclidean AUC are same, returns a z score of 2.86 ($p < 0.01$) suggesting the superiority of the metric learnt distance.

[13] Delong test is based on recognizing that the empirical AUC value is a U-Statistic. Theory of U-statistics allows to obtain the asymptotic distribution of two correlated empirical AUC curves. The hypothesis test for contrasting two AUCs follows.

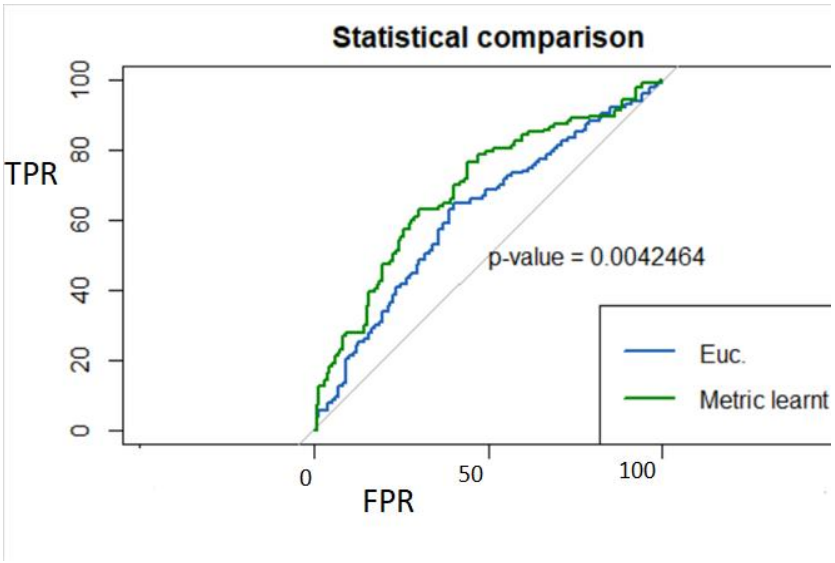

Figure 8: AUC comparison. Model 1 uses Euclidean distance on competency vectors (blue), Model 2 uses distance function learnt on selection data (green).

IV. Application

We have demonstrated in this setting that the competency distance approximated has predictive signal of selection on applying. With this ability, the firm can now bring more nuance to their review of internal transitions. For example, if previously the firm asked, 'what proportion of applications to teams supporting wealth management resulted in selection?' it can now ask, 'what proportion of applicants to wealth management teams with a high predicted probability of selection ($\widehat{P_{s=1}} > 0.5$, say) were selected?'. So, based on what complementary data is available the firm can conduct more nuanced review and analysis compared to before.

The management's objective is to use the data and machine learning apparatus to identify actions that would enhance internal mobility in the firm. The objective brings forth questions like – 'addition of what skill t would allow employee working in j_c improve their probability of being selected to a set of identified jobs?'. Such a question involves answering smaller questions like – by how much can a database engineer improve their chances of getting a process automation job, if they acquire 'Selenium' certification? Consistent with the objective, but a counterfactual question outside the scope of the data and predictive tool at hand. There is no record of training

(or skill acquisition) in the data, even to impute an answer to this question. There is emerging academic interest in approaches that look to modify j_c *i.e.*, generate a synthetic j_c' that incorporates the information of acquired skill t and subsequently compare probability of selection to j_s -- $\hat{d}(\widehat{X_C}, \widehat{X_S})$ and $\hat{d}(\widehat{X_c'}, \widehat{X_S})$; see Bana (2021) and Sisodia, Burnap and Kumar (2022). In such an approach, even if the difficult first part can be achieved, one cannot validate the predictions with the data at hand. Here, we take a simple approach to identify up-skilling opportunities. One that is based on observed choices (by firm and by applicants) and not counterfactual assertions.

Job Outlook and Exploratory Choices

In Fig. 9 below we plot the histogram for the competency distance $\hat{d}(\hat{\cdot})$ for the applications we observe in the validation set. The average probability of selection for applications falling in the regions are reported in the figure. Our interest is in applications that have a low predicted probability of selection. Note that, our prediction is based on observable difference in sought and current job competencies. We will refer applications to farther (larger competency distance) jobs as exploratory applications.

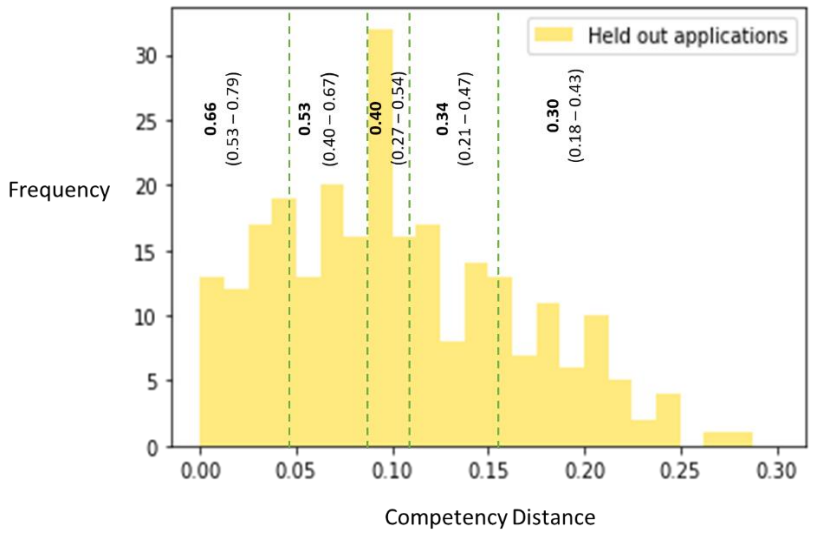

Figure 9: The histogram of competency distances between current and sought jobs in the held-out set. The vertical bands convey the estimated probability of selection in each quintile.

The ability to identify these exploratory applications allows us to think about programs that support exploratory applications. Two characteristics make exploratory applications and the ability to identify them, interesting. The first is that they have low probability of selection under current patterns of evaluation. From an up-skilling and mobility perspective, this is an opportunity. Support, to bridge the competency gap can help fulfill mobility aspirations of employees. The second is that we can observe the applicant's choice. When it comes to exploration -- seeking a job that requires notably different competencies from the current one there is usually more than one path available. The data has observed choices made by applicants. In sum, the ability to identify explorative choices (algorithmically from data) provides management with a list of job transitions that has low probability of success but is desired by employees and can improve mobility within the firm.

This brings the question of who makes exploratory choices? The firm perhaps do not want to support exploratory choices if it is made by applicants who have opportunities within the firm that utilizes their current job competencies. The firm may not want to institute a program (for example) that supports Webservices engineers to train as Hadoop engineers if there are several opportunities for webservices engineers to productively use their skills within the firm. In other words, such choices are not incentive compatible with the firm's talent management goals.

The data at hand allows to compare applicants along one observable dimension – the available internal positions at the time of their application. For each applicant (represented by their current job) j_c the analyst observes what other opportunities were available at the time they applied to the job. We can define this as an applicant's choice set $C_{j_c} =: \{j_s\}$ comprising jobs that were listed within d days prior or after their observed application.

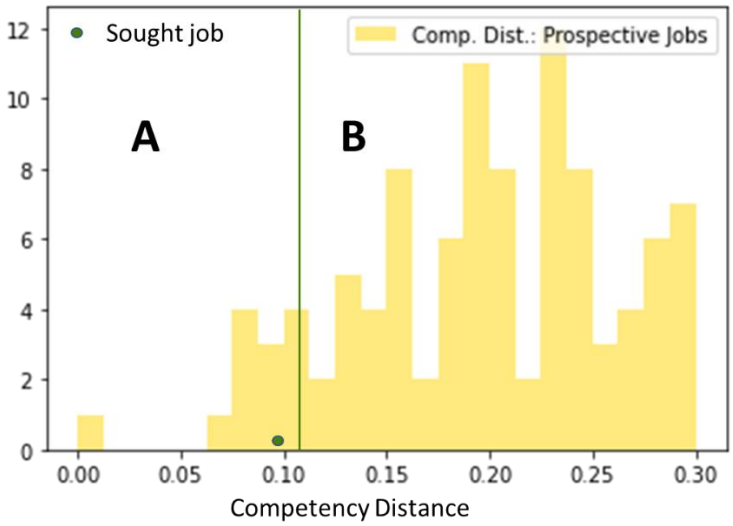

Figure 10: Competency distance histogram of an applicant to the positions in their choice set. The green dot indicates they applied to a position with distance equal to 0.09.

In Figure 10 above we have a histogram of the competency distances between an applicant j_c and the constituent jobs in its choice set C_{j_c}; (we picked $d = 20$). The distribution conveys information about the opportunities available to j_c and how their competencies compare to the current job. For supporting mobility, we want to identify applicants who face relatively fewer opportunities that are closer to their current job competencies. We define outlook as a ratio of jobs in C_{j_c}: count of jobs that are closer in competencies, to count of jobs farther away in competencies, as a summary measure of the distribution depicted in Figure 10 above. To proceed more precisely, we define all jobs with a competency distance less than 0.11 as close and others as far – depicted in Figure 10 as jobs in regions A and B respectively. The candidate's outlook O_{j_c} is the ratio of close jobs to far jobs[14]. The threshold value is the 3rd quintile of the competency distances observed between current and sought jobs in the validation set. For applications that had a competency distance greater than 0.11 the probability of selection was approximately 0.32 (see Figure 9). A higher outlook number conveys that among the jobs available to j_c a higher proportion are closer to their current competencies. With the ability to summarize information about the opportunities available to each applicant, we can compare the choice sets of applicants.

[14] $A_{j_c} =: \{j_s \in C_{j_c} \mid \widehat{d}(\widehat{X_s}, \widehat{X_c}) < 0.11\}$, $B_{j_c} =: \{j_s \in C_{j_c} \mid \widehat{d}(\widehat{X_s}, \widehat{X_c}) \geq 0.11\}$ and $O_{j_c} = |A_{j_c}|/|B_{j_c}|$.

Job Outlook and Exploratory Choice

In Figure 3A, for applications in the validation set, the outlook (O_{j_c}) is on the x-axis of the scatter diagram and competency distance to the sought job on the y-axis. We want to explore the link between the two. The question of interest to us is whether employees' exploratory choices are at random or are they linked to job outlook faced by the applicant (our only other observable about the applicant)? Notice in Figure 11 A below that the scatter takes the form of a right triangle. This is indicative that candidates with high outlook are less likely to apply to exploratory positions compared to candidates facing a low outlook.

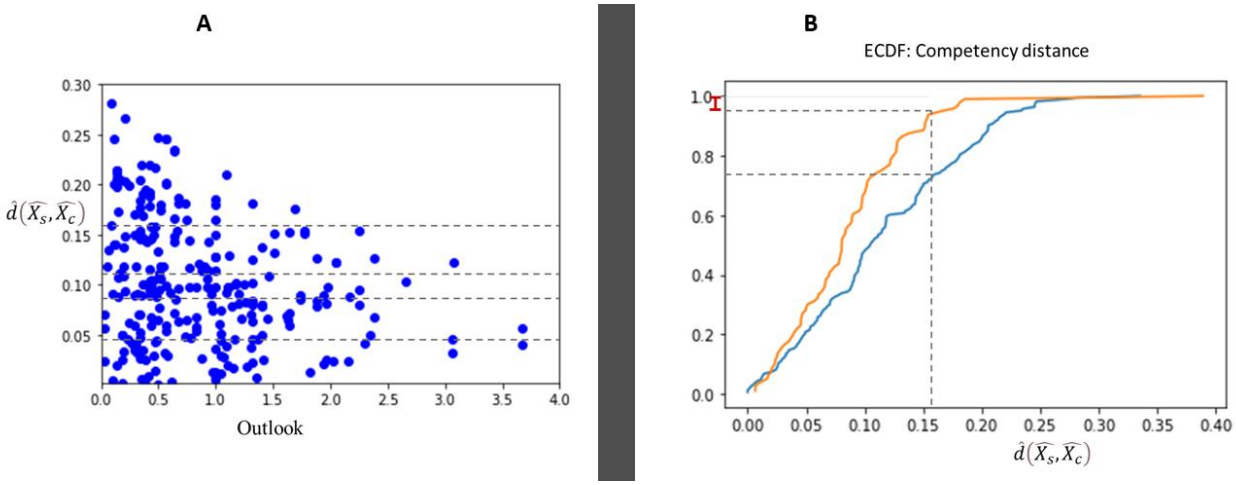

Figure 11: A. Scatter Diagram of outlook and competency distance. Horizontal grey lines indicate quintiles of competency distance in the validation set. B. Empirical CDF of competency distance to sought job for applicants with outlook < 1 (orange) and outlook > 1 (blue). The grey lines aid to check the probability of applying to an exploratory job (distance > 0.155).

To clarify the idea, we estimate separate empirical CDFs of the competency distance to the sought job, for applicants whose outlook were less than 1 ($O_{j_c} < 1$) and those whose outlook values were greater. We plot (in Figure 11 B) the former, the ECDF of applicants with low outlook in blue and the latter in orange. The orange plot, of applicants with $O_{j_c} > 1$, conveys that the probability that an applicant in this group seeks a position where $\hat{d}(\widehat{X_s}, \widehat{X_c}) \geq 0.15$ is small, in fact less than 5%. The corresponding probability of applying to an exploratory position when $O_{j_c} < 1$ is close to 24%. This conveys that exploratory choices come from employees facing an internal job market with fewer positions that require competencies closer to that required by their current jobs.

The observation brings strength to our idea of exploratory choices as up-skilling opportunities. Along with the fact that these choices reflect career paths that are harder to achieve and are employees' observed choices, they are also almost exclusively chosen by employees who have few opportunities to exploit their current competencies. So, training programs instituted to support these paths are unlikely to veer employees who have skills that are sought by different positions away to other jobs.

Summarization

We have a set of exploratory choices and want to design mobility programs that support employees along these paths. In our setting here, there are 51 cases where the distance is greater than 0.15 ($\hat{d}(\hat{\cdot}) > 0.15$). To institute initiatives that would support employees to make these career moves, we need to summarize the information contained in the cases. As the current and sought jobs are represented by their respective competency vectors, we can turn to clustering algorithms. Our goal is to obtain current and sought job clusters. For clarity, we will define the set comprising explorative choices as U: $\{(j_s, j_c)\} \mid \hat{d}(\widehat{X_s}, \widehat{X_c}) > \alpha\}$. In our exercise we will set $\alpha = 0.155$, applications that fall in the fifth quintile of competency distances. Our goal with the summarization exercise is to map the choices in set U to set $U_{\neq}^{S}$: $\{(C(\widehat{X_S}),\ C(\widehat{X_C}))\}$, whose distinct elements are clusters that the sought job and current job belong to respectively. The goal is for $U_{\neq}^{S}$ to have significantly fewer elements (lower cardinality) compared to U.

We now need a clustering algorithm C: $\{\hat{X}\} \rightarrow \{1,2, \dots, k\}$ that maps a job's competency vector to a cluster. We use k-means, the canonical clustering algorithm. In implementing the clustering algorithm, we have to choose k[15]. We choose k to minimize the cardinality of $U_{\neq}^{S}$ subject to the constraint that the sought job and current job for any case in set U do not belong to the same cluster ($C(\widehat{X_S}) \neq C(\widehat{X_C})\ \forall\ (\widehat{X_S}, \widehat{X_C}) \in U$). In practice, we used a larger set of job descriptions (their competency vectors) for the clustering exercise and not just those that feature in U. In our context

[15] Note that we previously used k to refer to the parameter to be tuned in the LDA model -- section III. We are reusing k as the number of clusters. As both algorithms are very popular, and their respective tuning parameters referred to as k, we stick to standard nomenclature.

we have $k = 19$[16]. We can leverage the fact that our cluster-centroids are interpretable. They are competency vectors, where each dimension has a description available. For each cluster, we pick up to three features that have the highest weights against them, as descriptors for the cluster. The procedure is illustrated in Figure 12 and the result is summarized in Figure 13 below.

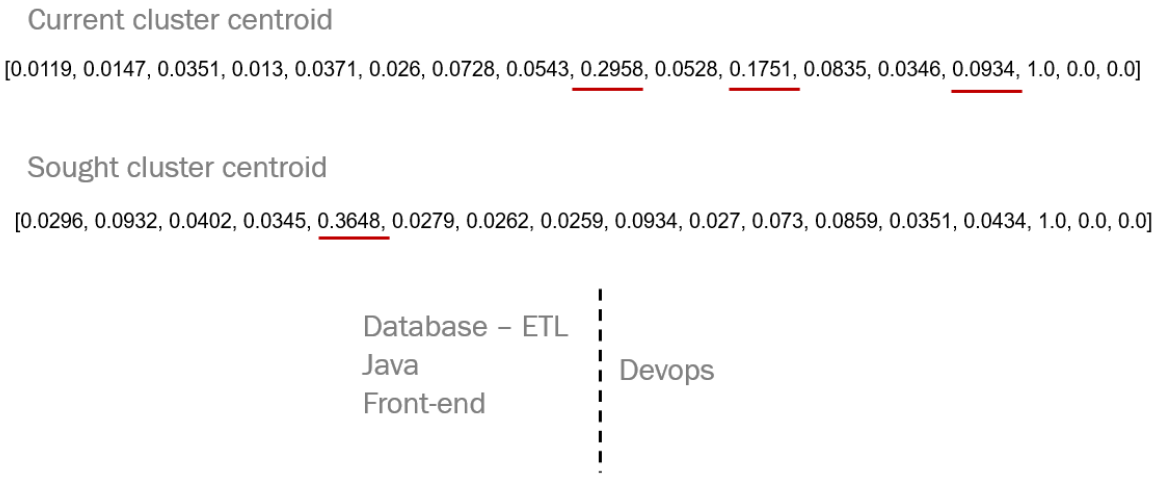

Figure 12: Interpreting cluster centroids

[16] In the clustering exercise we included the full sample of selection data which comprises close to 2000 job descriptions. This corpus was divided to 19 clusters satisfying the constraints described in text above.

Current cluster competencies	Sought cluster competencies
Database – ETL+ Java + Front-end	Devops
	Hadoop dev.
	Hadoop dev. + Database - ETL
	.NET - Webservices
Mainframe + Developer	Hadoop dev.
Database + Sys. Adm. + **Support**	Devops
Mainframe + **Support**	Mainframe + Developer
.NET + **Support**/**Test**	Database-ETL + Java + Front-end

Figure 13: Source and (sought) destination clusters represented by their key competencies. The functional role is highlighted only when they are either support or tester.

Note that, the interpretability of clusters is not an integral part of our approach. The users can retrieve the full job descriptions for review. The benefit of having interpretable clusters is rather more practical. It enables easy communication to decision makers and employees who are the intended beneficiaries of any training programs instituted based on this approach. It is easier to communicate – "the data conveys that 'mainframe support' employees have few new opportunities in the firm and apply to 'devops' positions with low probability of success", than present the same as "employees in cluster 1 … and apply to positions in cluster 6". On a cautionary note, there is information loss in generating the labels and the actual program design should evaluate the jobs that map on to these clusters. While our chief purpose in this paper is to present a broadly applicable data analytic method, it is useful to note the patterns generated. Two current job clusters –from which employees are looking to move away, are also sought jobs by employees in other positions with low outlook. For example, employees working as 'Mainframe developer' are seeking 'Hadoop developer' positions, but 'Mainframe support' professionals are seeking 'Mainframe developer' positions for which they seem to have a low probability of selection. This indicates that there are seekers for jobs who themselves offer (relatively) low potential for mobility.

Coverage

As an up-skilling initiative, we have identified current job and sought job clusters. A managerially relevant question is regarding the scope of the identified up-skilling paths? In other words, how many current employees are potential beneficiaries of the up-skilling paths identified? We have job descriptions and date of starting in the current positions for 6800 employees. We can see how many among them map onto identified current job clusters. If they map onto the current job clusters, then they face a low outlook and are potential beneficiaries of training programs along the identified paths. To conduct this exercise, we first ask how many current employees have been in their positions for at least 1000 days? This narrows down the list of potential beneficiaries, as employees who started more recently are less likely looking to move.

In our setting, we find 2186 employees who have served in their current positions for over 1000 days (32% of all employees). Out of which 673 employees fall into one of the five identified current job clusters. In other words, ~10% of current employees are potential beneficiaries from the identified opportunities. The numbers are computed here for a concrete illustration of the idea and has no relevance away from the setting. The coverage numbers are to be seen as a ceiling for the number of current employees who can potentially benefit from the opportunities identified. Our framework has picked out mobility paths – i) that have a low probability of success ii) are observed choices by iii) employees with a low outlook when it comes to internal opportunities. We do not offer any guidance on the contents of a program that can bridge the competency gap. It is outside the framework's scope and something for management and technical leadership to design.

In the absence of a supportive program, only a small proportion of employees make these choices[17]. What proportion of employees we count here would benefit from the program is a tempting question, but one we cannot predict using application patterns observed here. In sum,

[17] Note that even employees facing low outlook (< 1) who contributes the majority of exploratory choices observed, make non-exploratory applications ($\hat{d}(\hat{\cdot}) < 0.155$ in 74% of the cases) in most cases.

before instituting the programs we can only obtain a count of potential beneficiaries and not predictions about improvements in mobility outcomes.

V. Conclusion and Discussion

We have contributed an analytical framework that large firms with an internal labor market may adopt, to learn from their internal mobility data. We use administrative data that is commonly available or can be collected at little cost. It comprises a machine learning model which predicts whether the firm selects an internal applicant to a new position. The prediction ability is then used to review employee application patterns to identify training (skill-building) opportunities consistent with employee and firm interests. While our machine learning model is custom built for internal labor markets in large IT organizations, it can be adapted to be used in any large firm where current employees are encouraged to apply to new positions.

We illustrate that information encoded in job descriptions, past firm decisions and contextual knowledge can be combined using relatively well-known machine learning techniques to extract signal and build a prediction model. Our prediction model which has three training steps – the LDA, predictive screening and metric (distance function) learning can be accomplished on a laptop within few minutes, even for organizations with several thousands of observations. Alternatively, like in the setting studied here the model gives valuable performance even when the training data is only in the few thousands. Our model and the larger analytic framework comprising the clustering algorithm and supplementary analysis can be ported to different firm settings, with little starting costs.

We incorporated our knowledge of the context in extracting information from job descriptions - we separately incorporated technical and functional skills and avoided certain general information contained in JDs. Firms can have supplementary information, which was not available in our study setting. The relative simplicity of our approach should encourage potential adapters of this approach to incorporate their own supplementary information to the prediction model. There are today vastly sophisticated algorithms for language modeling. Using them, increases the complexity in incorporating contextual information and also in applying metric learning. Using a more

advanced approach to model the text information only sharpens the model predictions and does not affect its logic and use in any other way.

We do not extend the use our prediction apparatus to tackle causal questions. How much can an employee in job j_c predictably improve their probability of selection to job j_s if they acquire skill t? is a relevant question for human capital management, but one we can neither impute nor validate. Our prediction model is trained on actual job descriptions generated in the firm. Acquiring a new skill and communicating it during an interview etc. are part of the actual selection process, but unobserved and therefore not included in how the model is trained and validated here. Obtaining the model's estimate of selection probability by modifying j_c to j_{c+t} has no logical support or empirical validation.

A last point of discussion is regarding the use of machine learning. We illustrate how prediction problems can be formulated such that the value does not come by replacing a human-decision maker. Machine learning is used here, to identify career moves that are preferred by employees but is likely to be turned down by the firm on applying. They are identified with the objective of supporting and facilitating such moves. In other words, it is a machine learning approach that focuses on empowering the employees. HR analytics is a young field with big ambitions – use data to develop a nuanced understanding of modern organizations to support the firms and also people that constitute them. We illustrate the predictive potential in readily available HR data and how they may be used to support employees to improve their chances of securing a preferred job. Our approach is data driven but guided by organizational/behavioral thinking. Synthesis of methods and approaches open up in chasing the field's lofty ambitions.

References

Bana, Sara. 2021. "Using Language Model to Understand Wage Premia." Working Paper.

Blei, David M., Andrew Y. Ng, and Michael B. Jordan. 2003. "Latent Dirichlet Allocation." *Journal of Machine Learning Research* 993-1022.

Chamorro-Premuzic, Tomas, and Ian Bailie. 2020. "Tech Is Transforming People Analytics. Is That a Good Thing? ." *Harvard Business Review.*

DeLong, Elizabeth R., David M. DeLong, and Daniel L. Clarke-Pearson. 1988. "Comparing the Area Under Two or More Correlated Receive Operating Characteristic Curves: A Non parameteric Approach." *Biometrics.*

Fan, Jianqing, and Jinchi Lv. 2007. "Sure independence screening for ultrahigh dimensional feature space." *Journal of the Royal Statistical Society Statistical Methodology Series B.*

Fu, Runshan, Yan Huang, and Param Vir Singh. 2021. "Crowd, Machine, Lending and Bias." *Information Systems Research.*

Gentzkow, Matthew. 2018. *Media and Artificial Intelligence.* Working Paper, Toulouse School of Economics.

Kleinberg, John, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. 2018. "Human Decisions and Machine Predictions." *Quarterly Journal of Economics.*

Kleinberg, John, Jens Ludwig, Senthil Mullainathan, and Ziyad Obermeyer. 2015. "Prediction Policy Problems." *American Economic Review: Papers & Proceedings* 491–495.

Li, Danielle, Lindsey Raymond, and Peter Bergman. 2022. "Hiring as Exploration." Working Paper.

Sisodia, Ankit, Alex Burnap, and Vineet Kumar. 2022. "Automatically Discovering Visual Product Characteristics." Working Paper.

Tambe, Prasanna, Peter Cappelli, and Valery Yakubovich. 2019. "Artificial Intelligence in Human Resources Management: Challenges and a Path Forward." *California Management Review.*

Xing, Eric P., Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. 2002. "Distance metric learning, with application to clustering with side-information." *Advances in neural information processing systems 15.*

Appendix A. Sample job description

Overview

We are one of the world's leading financial institutions, serving individual consumers, small and middle-market businesses, and large corporations with a full range of banking, investing, asset management and other financial and risk management products and services …

Process Overview

Build and evolve a consistent Authorized Data Source within Consumer & Small Business Bank (CSBB), with an organization and operating processes to support both the strategic and tactical analytics needs of the Consumer Bank.

Job Description

Hadoop developer for multiple initiatives. Develop Big Data Strategy and Roadmap for the Enterprise. Experience in Capacity Planning, Cluster Designing and Deployment. Benchmark systems, analyze system bottlenecks, and propose solutions to eliminate them. Develop highly scalable and extensible Big Data platform, which enables collection, storage, modeling, and analysis of massive data sets from numerous channels. Continuously evaluate new technologies, innovate and deliver solution for business-critical applications.

Responsibilities

Assists the team with the design of the architect layer to ensure re-usable metrics and attributes within the reporting layer. Responsible for creating and maintaining necessary documentation (MDR) to ensure audit readiness where necessary. Prototype improvement ideas. Work effectively with the global team. Expected to play technical leadership as an individual contributor. Articulate challenges, propose and drive solutions to make entire India engagement successful.

Mandatory skills

Extensive knowledge of Hadoop stack and storage technologies HDFS, MapReduce, Yarn, HIVE, sqoop, Impala, spark, flume, kafka and oozie

Extensive Knowledge on Bigdata Enterprise architecture (Cloudera preferred)

Experience in No SQL Technologies (Cassandra, Hbase)

Desired skills

Experience in Real time streaming (Kafka)

Experience with Big Data Analytics & Business Intelligence and Industry standard tools integrated with Hadoop ecosystem. (R , Python)

Visual Analytics Tools knowledge (Tableau)

Data Integration, Data Security on Hadoop ecosystem. (Kerberos)

Awareness or experience with Data Lake with Cloudera ecosystem

Appendix B. Screened words for developer, support and testing

dev_words = ['development', 'design', 'practice', 'technology', 'experience', 'develop', 'enhance', 'unit', 'candidate', 'collaborate', 'skill', 'deliver', 'ability', 'write', 'software', 'learn', 'deliverable', 'domain', 'architecture', 'contribute', 'have', 'datum', 'line', 'solution', 'resource', 'understanding', 'requirement', 'program', 'analyze', 'quality']

support_words = ['support', 'individual', 'issue', 'production', 'change', 'service', 'provide', 'management', 'incident', 'review', 'process', 'request', 'cause', 'communication', 'monitoring', 'resolve', 'task', 'deployment', 'drive', 'take', 'triage', 'problem', 'call', 'manage', 'resolution', 'root', 'weekend', 'identify', 'risk', 'release']

test_words = ['automate', 'tool', 'testing', 'activity', 'automation', 'create', 'execute', 'skill', 'defect', 'framework', 'script', 'have', 'involve', 'delivery', 'document', 'adapt', 'assist', 'case', 'define', 'methodology', 'reporting', 'specification', 'suite', 'understand', 'member', 'test', 'standard', 'program', 'accountability']