Is It Time For a Career Switch?

Jian Wang, Yi Zhang
University of California, Santa Cruz
Santa Cruz, CA 95060 USA
{jwang30, yiz}@soe.ucsc.edu

Christian Posse, Anmol Bhasin
LinkedIn Corp
Mountain View, CA 94043 USA
{cposse, abhasin}@linkedin.com

ABSTRACT

Tenure is a critical factor for an individual to consider when making a job transition. For instance, *software engineers* make a job transition to *senior software engineers* in a span of 2 years on average, or it takes for approximately 3 years for *realtors* to switch to *brokers*. While most existing work on recommender systems focuses on finding *what* to recommend to a user, this paper places emphasis on *when* to make appropriate recommendations and its impact on the item selection in the context of a job recommender system. The approach we propose, however, is general and can be applied to any recommendation scenario where the decision-making process is dependent on the tenure (i.e., the time interval) between successive decisions.

Our approach is inspired by the proportional hazards model in statistics. It models the tenure between two successive decisions and related factors. We further extend the model with a hierarchical Bayesian framework to address the problem of data sparsity. The proposed model estimates the likelihood of a user's decision to make a job transition at a certain time, which is denoted as the **tenure-based decision probability**. New and appropriate evaluation metrics are designed to analyze the model's performance on deciding when is the right time to recommend a job to a user. We validate the soundness of our approach by evaluating it with an anonymous job application dataset across 140+ industries on LinkedIn. Experimental results show that the hierarchical proportional hazards model has better predictability of the user's decision time, which in turn helps the recommender system to achieve higher utility/user satisfaction.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Design, Experimentation

Keywords

Recommender System, Hazards Model, Tenure

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

1. INTRODUCTION

Recommender systems are popular research topics in the information retrieval community. Host of academic and industrial incarnations of recommender systems exists in domains such as movies (Netflix), music (Pandora), e-commerce product recommendations (eBay, Amazon). Traditional research in recommender systems aims to find right item(s) to recommend to a user. For example, a job recommender system would recommend a *senior software engineer* position to a user working in an engineering function in the software domain. This discovery can be powered by content-based models, collaborative filtering, or hybrid systems. In addition to finding the right item to recommend, another key aspect is the timeliness aspect of making the appropriate recommendation. This aspect becomes critical in settings where the decision-making process is dependent on the tenure (i.e., the time interval) between successive decisions, such as in a job transiting scenario. To make recommendations at the right time helps the system to achieve higher utility. Utility is defined as the satisfaction or value a user gets. To motivate the discussion, consider the following question - when should the recommender system recommend a *senior software engineer* position to *software engineers*? The system is likely to achieve positive utility when a *software engineer* who works for 2 years receives such a recommendation. Yet the system might achieve negative utility when a *software engineer* who works for 2 months or 5+ years receives such a recommendation. In this paper, we focus on building models to find the right time to make appropriate recommendations.

We start tackling the problem by assuming that the job transiting process follows the Markov renewal process, i.e. the sequence of making decisions is a Markov chain. In addition the waiting time depends only on the last decision and the current decision. Inspired by the survival model in statistics, waiting time between successive decisions can be modeled by the Weibull distribution. In reality however, the waiting time is not only dependent on the last decision and the current decision, but also dependent on other factors. In the job domain these factors include the user's profile and behavioral characteristics, the nature of the current position and potential job opportunities, the interaction patterns between the user and the job or the functional area, the global economic environment, and a host of other externalities like location, time of the year, etc. To illustrate the point, let us consider the scenario of a potential job from a company with a high reputation. In this case, the user may change to this new job earlier than average. To incorporate these covariates (i.e., factors or features) into the model, we use

the proportional hazards model to model the tenure before a job transition. We further extend the model with the hierarchical Bayesian framework to solve the data sparsity issue. The proposed model predicts the probability of a user making a decision at time t, given that the user did not make the decision before time t. We denote this probability as the **tenure-based decision probability**. This could be used by a hybrid recommender system in two ways. To determine whether to present the recommendation at a certain time in the push-based scenario, the system can treat the probability as a threshold in the filtering process. To determine which items to recommend in the pull-based scenario, the system can use the probability as the item's additional feature in the ranking process.

We perform experiments with an anonymous job application dataset from millions of users in 140+ industries from LinkedIn. New evaluation metrics are designed to analyze the hazards model's performance on predicting the tenure-based decision probability. Metrics include the perplexity/likelihood of the model, the accuracy of the estimated decision time/tenure, the utility of the recommender system, etc. Experiments demonstrate that the hierarchical proportional hazards model has the better predictability of the decision time, which in turn improves the utility of the recommender system.

The major contribution of this paper includes the following:

- Analyze the problem of finding the **right time** to make recommendations in the job domain.
- Propose using the **proportional hazards model** to tackle the problem and extend it with a hierarchical Bayesian framework.
- Evaluate the model with a real-world job application data from LinkedIn to demonstrate the better predictability of the proposed model, as well as the effect of the **tenure-based decision probability** in improving the utility of recommender systems.

2. RELATED WORK

A major task of the recommender system is to present recommendations to the user. The task is usually conducted by first predicting a user's ratings for each item and then ranking all items in the descending order. There are two major recommendation approaches: content-based filtering and collaborative filtering. Content-based filtering [16, 19] assumes that descriptive features of an item indicate a user's preferences. Thus, a recommender system makes a decision for a user based on the descriptive features of other items the user likes or dislikes. Usually, the system recommends items that are similar to what the user liked before. Collaborative filtering [9, 25, 10, 15, 27, 18, 8] on the other hand assumes that users with similar tastes on some items may also have similar preferences on other items. Thus, the main idea is to use the behavior history from other like-minded users to provide the current user with good recommendations. Research on collaborative filtering algorithms reached a peak due to the 1 million dollar Netflix movie recommendation competition [1]. Factorization-based collaborative filtering approaches [4, 11, 28, 23], such as the regularized Singular Value Decomposition performed well on this competition, possibly better than Netflix's own well-tuned Pearson correlation coefficient algorithm. A common characteristic of these models is the introduction of user latent factors or/and item latent factors to solve the data sparsity issue.

Factoring time [12, 27, 34, 22, 24, 14, 33, 5, 13, 31, 26] has received much research attention. In the field of recommender systems, one focus is about the drift of the user's preference over time [12, 32, 21]. Koren [12] revamped two popular collaborative filtering methods by modeling the time drifting factor of user preferences. Rendel et. al [21] proposed a factorized personalized model that subsumes both a common Markov chain and the normal matrix factorization model. Compared to their work, our work explicitly models the tenure and the user's interest at each time as a white box. On one hand, the resulting tenure-based decision probability can be used in the hybrid system to find relevant items while on the other, it can be used as a signal to determine when is the right time to recommend an item. Another research focus is modeling the tenure between purchase orders in the e-commerce domain [27, 34]. Wang et. al [27] discovered different post-purchase behavior in different time windows after purchase. Zhao et. al [34] used the purchasing tenure to improve the temporal diversity [21]. The tenure and the corresponding purchase probability is modeled inside the framework of a utility-based recommender system [28]. The hybrid system takes the tenure into consideration when ranking all candidate items. Different from their work, we propose a more generalized model to explicitly predict the tenure-based decision probability. This probability can be leveraged in the filtering process of recommendation items from any system in the domain where the time interval between successive decisions is an important factor.

3. HIERARCHICAL PROPORTIONAL HAZARDS MODEL

3.1 Problem Definition

In this paper, we aim to answer the question: When is the right time to make a job recommendation and how do we leverage it in a job recommender system? The following notations are used in this paper. The relationship between different variables is shown in Figure 1.

- $u = 1, 2, ..., U$: the index of the user.
- $a, b = 1, 2, ..., C$: the index of the item category. In the job domain, it is the title of the job, such as *software engineer*, *realtor*, *lawyer*, etc.
- $j_a = 1_a, 2_a, ..., J_a$: the index of the item in category a. In our setup, an item is a job. This item has metadata like industry, seniority, function, company, etc. For example, a job in category *Software Engineer* belongs to the *computer software* industry, *Information Technology* function, *Google* company, and *Entry* seniority level.
- $m = 1, 2, ..., M$: the index of the decision transition between items in category a to items in category b. If a user with job j_a applies to a job j_b, these two job categories $\{a \rightarrow b\}$ form a decision transition m. Note that horizontal transitions such as $\{a \rightarrow a\}$ is included in the model as well.

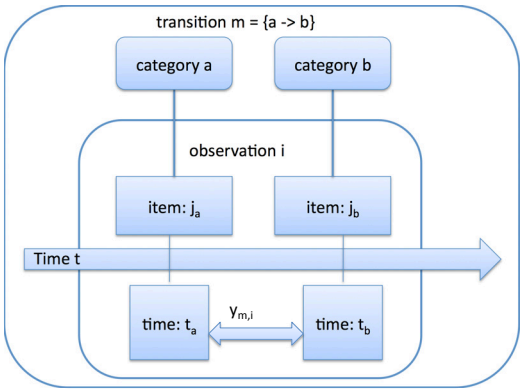

Figure 1: Illustration of the relationship of variables. The user first makes a job decision of item j_a in category a at time t_a and then makes a decision of item j_b in category b at time t_b. This transition from category a to category b is the i^{th} observation in transition $m = \{a \to b\}$. This observation is associated with two parts: 1) tenure $y_{m,i}$ and 2) covariates $\mathbf{x_{m,i}}$ (which is not shown in the plot).

- $D = \{D_1, ..., D_m, ..., D_M\}$: The observed data of all transitions from all users.
- $D_m = \{y_{m,i}, \mathbf{x_{m,i}}\}$: A set of observed data associated with transition m. Each transition m has N_m data observations from all users. Each observation $i = 1, ..., N_m$ in transition m is associated with two parts: the tenure $y_{m,i}$ and covariates $\mathbf{x_{m,i}}$.
- $y_{m,i}$: the tenure with the i^{th} observation in transition m. It is the tenure between the user's decision time t_b of item j_b and the user's decision time t_a of item j_a. $y_{m,i} = t_b - t_a$.
- $\mathbf{x_{m,i}}$: the k-dimensional vector of covariates that associate with the i^{th} observation in transition m. Covariates could be associated with user u who makes the decision transition, the source item j_a, the destination item j_b, the interaction between j_a and j_b, the global environment, etc.

The goal of the model is to predict the probability that a user makes a decision of item j_b at current time t_b, given that she made the last decision of j_a at time t_a and she did not make the transition decision up to time t_b. It is the same as predicting the probability that a user makes a decision of item j_b at tenure $y_{m,i} = t_b - t_a$ with covariates $\mathbf{x_{m,i}}$ being associated with the transition.

3.2 Review of Proportional Hazards Model

Before describing the hierarchical model that we propose, we first briefly review the basic proportional hazards model. In survival analysis, the survival function determines the time of a particular event, often the failure of a machine or the death of a subject. Here we consider *failure* as a user making a decision to transit to a new job. Let $p(y)$ denote the probability density function of such an event. The cumulative distribution function $P(y)$ and survival function $S(y)$ are then given by

$$P(y) = Pr(T \leq y) \quad (1)$$
$$S(y) = Pr(T > y) = 1 - P(y) \quad (2)$$

where T is a random variable denoting the survival time. In addition, the hazards function is defined as the event rate at tenure y, given that the event does not occur until tenure y or later. $h(y) = \frac{p(y)}{S(y)}$. In the real world, the hazards function is dependent on covariates. Two common approaches to incorporate covariates $\mathbf{x}$ in the hazards model are:

Cox proportional hazards model, which assumes that the covariates are multiplicatively related to the hazards [20]:

$$h(y) = h_0(y)exp(\beta^{\mathbf{T}}\mathbf{x}) \quad (3)$$

where $h_0(y)$ is the baseline hazards function and β is a vector of parameters.

Accelerated life model, which assumes that the covariates are multiplicatively related to the survival time [30], i.e., $T = T_0 exp\{-\beta^T \mathbf{x}\}$ where T_0 is the baseline survival time. Hence,

$$S(y|\mathbf{x}) = Pr(T > y|\mathbf{x}) \quad (4)$$
$$= Pr(T_0 exp\{-\beta^T \mathbf{x}\} > y) \quad (5)$$
$$= Pr(T_0 > y \cdot exp\{\beta^T \mathbf{x}\}) \quad (6)$$
$$= S_0(y \cdot exp\{\beta^{\mathbf{T}}\mathbf{x}\}) \quad (7)$$

where $S_0(y)$ is the baseline survival function.

Both approaches coincide if the Weibull distribution is used for $p(y)$[1]. Thus, we choose that distribution in this paper, which is given by

$$p(y) = \gamma\theta y^{\gamma-1} exp\{-\theta y^\gamma\} \quad (8)$$

where γ is the shape parameter and θ is the scale parameter. The corresponding baseline hazards function becomes:

$$h_0(y) = \gamma\theta y^{\gamma-1} = \gamma exp(\beta_0) y^{\gamma-1}. \quad (9)$$

- If $\gamma > 1$, $h_0(y)$ increases with time.
- If $\gamma < 1$, $h_0(y)$ decreases with time.
- If $\gamma = 1$, $h_0(y)$ is constant.

To incorporate covariates $\mathbf{x} = \{x_1, ..., x_k\}$, we extend θ from $exp(\beta_0)$ to $exp\{\beta_0\}exp\{\beta_1 x_1 + ... + \beta_k x_k\}$, i.e., $exp\{\beta^T \mathbf{x}\}$ where we extend $\beta = \{\beta_0, ..., \beta_k\}$ and $\mathbf{x}$ as $\{x_0 = 1, x_1, ..., x_k\}$. Thus, the probability density function becomes:

$$p(y) = \gamma exp\{\beta^T \mathbf{x}\} y^{\gamma-1} exp\{-exp\{\beta^T \mathbf{x}\} y^\gamma\} \quad (10)$$

This probability density function represents the basic proportional hazards model that models the tenure before a transition with associated covariates.

[1]Mathematical details are in http://data.princeton.edu/pop509/ParametricSurvival.pdf. γ in our notation is p in their notation. θ in our notation is λ^p in their notation.

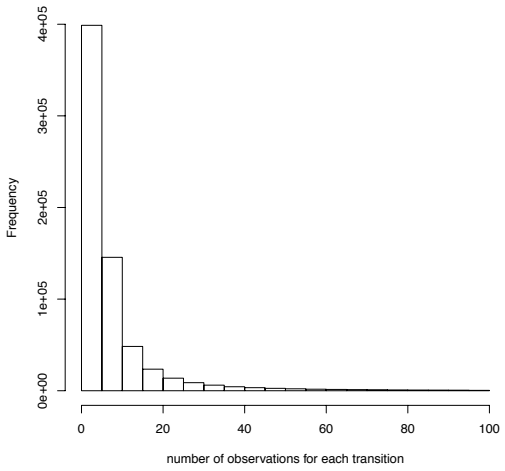

Figure 2: Histogram of number of observations (from 3 to 50) for each job transition $m = \{a \to b\}$.

3.3 Model Extension with Bayesian Framework

In real use cases, the number of observations for each transition tends to follow the power law distribution. In other words, few transitions are often observed while most transitions are rare events, making it hard to learn parameters of the corresponding hazards model. To illustrate this, we show the histogram of the number of observations for each transition in the job application data in Figure 2. To solve the data sparsity issue, we extend the proportional hazards model with a hierarchical Bayesian framework.

The goal of the hierarchical Bayesian framework is to borrow information from other transitions when learning the parameters for transition m. We derive the following hierarchical model, which is illustrated in Figure 3:

- For each transition m, β_m is sampled from the Gaussian distribution: $\beta_m \sim N(\mu_\beta, \mathbf{\Sigma}_\beta)$ and γ_m is sampled from the Gaussian distribution: $\gamma_m \sim N(\mu_\gamma, \sigma_\gamma^2)$. Note that μ_β is a $(k+1)$-dimensional vector and $\mathbf{\Sigma}_\beta$ is a $(k+1)\times(k+1)$ matrix, where k is the number of covariates in the model. We denote $\phi = (\mu_\beta, \mathbf{\Sigma}_\beta, \mu_\gamma, \sigma_\gamma^2)$.
- μ_β and μ_γ are sampled from $N(0, a\mathbf{I})$ and $N(0, b)$, respectively, and Σ_β and σ_γ^2 are sampled from the inverse Wishart Distribution $\mathbf{W^{-1}}(\mathbf{I}, c)$ and the inverse Gamma distribution $\Gamma^{-1}(1, d)$, respectively, where $\mathbf{I}$ is the $(k+1)\times(k+1)$ identity matrix, and $a, b, c, d > 0$.
- For each i^{th} observation of transition m with its covariates $\mathbf{x_{m,i}}$, its tenure $y_{m,i}$ is sampled from the proportional hazards model

$$p(y_{m,i}|\mathbf{x_{m,i}}, \beta_m, \gamma_m) \quad (11)$$
$$= \gamma_m exp\{\beta_m^T \mathbf{x_{m,i}}\} y_{m,i}^{\gamma_m - 1} exp\{-exp\{\beta_m^T \mathbf{x_{m,i}}\} y_{m,i}^{\gamma_m}\}$$

Let $\sigma = (\phi, \beta_\mathbf{1}, \gamma_1, ..., \beta_M, \gamma_M)$ represent parameters that need to be estimated. The joint likelihood for all variables in the probabilistic model is:

$$L(D, \sigma) = p(\phi) \prod_{m=1}^{M} p(\beta_m, \gamma_m|\phi) \prod_{i}^{N_m} p(y_{m,i}|\beta_m, \gamma_m, \mathbf{x_{m,i}}) \quad (12)$$

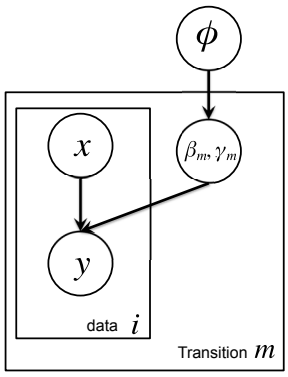

Figure 3: Illustration of dependencies of variables in the hierarchical proportional hazards model. It shows the i^{th} observation of transition m. $y_{m,i}$ is the tenure which is conditioned on covariates $\mathbf{x_{m,i}}$ that are related to this transition and the proportional hazards model. Each transition m has its own parameters of the hazards model β_m, γ_m. Models of each transition share information through the prior, $\phi = (\mu_\beta, \mathbf{\Sigma}_\beta, \mu_\gamma, \sigma_\gamma^2)$.

3.4 Parameter Estimation

Our model contains many hidden variables, some of them being high-dimensional vectors (μ_β and all β_m). Hence, the traditional Bayesian method might be too computationally expensive to learn the model. Instead we propose an iterative method with a point estimation in each step. We first introduce constants $c_i, (i = 1, 2, 3, 4)$ to replace functions of $a, b, \mathbf{\Sigma}_\beta, \sigma_\gamma^2$, respectively, with the same model effect. They can be viewed as regularization factors to avoid overfitting and can be set by cross-validation in the experiment. The maximum likelihood estimation of the remaining parameters is shown in Equation 13.

$$\begin{aligned}(\hat{\mu_\beta}, \hat{\mu_\gamma}, \hat{\beta_1}, \hat{\gamma_1}, ..., \hat{\beta_M}, \hat{\gamma_M}) &= \arg\max L(D, \sigma) \quad (13)\\ &= \arg\min\{c_1||\mu_\beta||^2 + c_2\mu_\gamma^2\}\\ &+ \sum_{m=1}^{M}\{c_3||\beta_m - \mu_\beta||^2 + c_4(\gamma_m - \mu_\gamma)^2\}\\ &+ \sum_{m=1}^{M}\{\sum_{i=1}^{N_m} -\log(p(y_{m,i}|\beta_m, \gamma_m, \mathbf{x_{m,i}}))\}\end{aligned}$$

The steps to solve the previous equation are shown in Algorithm 1. We first initialize μ^0 and update parameters by following steps 3-5 and step 6 iteratively until convergence.

In steps 3-5, the goal is to estimate parameters of the hazards model $\beta_1^n, \gamma_1^n, ..., \beta_M^n, \gamma_M^n$, based on the current estimation of the prior $\mu^n = (\mu_\beta^n, \mu_\gamma^n)$, where n denotes the iteration index. Because the parameters β_m^n, γ_m^n for each transition m are independent from each other, we estimate them one by

Algorithm 1 The parameter learning algorithm

1: Initialize $\mu^0 = \{\mu_\beta^0, \mu_\gamma^0\}, n \leftarrow 0$
2: **repeat**
3: **for** <m = 1,...,M> **do**
4: Compute the parameters of the hazards model, β_m^n and γ_m^n, based on μ^n for each transition m.
5: **end for**
6: Compute μ^{n+1} based on the hazards model β_m^n, γ_m^n of each transition m with conjugate gradient descent.
7: $n \leftarrow n + 1$
8: **until** <Convergence>
9: **return** $\mu = \{\mu_\beta, \mu_\gamma\}, \beta_1, \gamma_1, ..., \beta_M, \gamma_M$

one as:

$$(\hat{\beta_m^n}, \hat{\gamma_m^n}) = \arg\min\{c_3||\beta_m - \mu_\beta^n||^2 + c_4(\gamma_m - \mu_\gamma^n)^2\} - \sum_{i=1}^{N_m}\{\log p(y_{m,i}|\beta_m, \gamma_m, \mathbf{x_{m,i}})\} \quad (14)$$

We use the following steps in Algorithm 2 iteratively to estimate β_m^n, γ_m^n.

Algorithm 2 Step 4 in Algorithm 1

1: Initialize $\gamma_m^{n,0}, c \leftarrow 0$
2: **repeat**
3: Compute $\beta_m^{n,c}$ based on $\gamma_m^{n,c}$ with conjugate gradient descent.
4: Compute $\gamma_m^{n,c+1}$ based on $\beta_m^{n,c}$ with conjugate gradient descent.
5: $c \leftarrow c + 1$
6: **until** <Convergence>
7: **return** β_m^n, γ_m^n

4. TENURE-BASED DECISION PROBABILITY

4.1 Definition

The **tenure-based decision probability** of item j_b for user u is defined as: the probability that user u would make a job transition to j_b at time between t_b and $t_b + \Delta t$, given that the user starts her current position j_a at time t_a and she did not make the decision up to time t_b. In other words, it is the probability that the survival time T would be between $y_{m,i}$ and $y_{m,i} + \Delta t$, given that T is not less than $y_{m,i}$. We denote the prediction of the tenure-based decision probability by model q as $q(y_{m,i}, \mathbf{x_{m,i}})$. It is given by

$$q(y_{m,i}, \mathbf{x_{m,i}}) = Pr(y_{m,i} < T \leq y_{m,i} + \Delta t | T > y_{m,i}) \quad (15)$$

$$= \frac{Pr(T \leq y_{m,i} + \Delta t) - Pr(T \leq y_{m,i})}{1 - Pr(T \leq y_{m,i})} \quad (16)$$

where

$$Pr(T \leq y_{m,i}) = 1 - exp\{-y_{m,i}^{\gamma_m} exp\{\beta_m^T \mathbf{x_{m,i}}\}\} \quad (17)$$

and each transition m has its own parameters (β_m, γ_m) in model q.

Table 1: The utility set of the recommender system. There are four types of utilities, depending on whether the system shows the item to the user and whether the user accepts the item.

	show:Y	show:N
accept:Y	u_{TP}	u_{FN}
accept:N	u_{FP}	u_{TN}

4.2 Usage

The major goal of the recommender system is to achieve high utility/user satisfaction. The user satisfaction is dependent on both the relevance and the time of the recommendation. While an irrelevant recommendation results in a negative utility, a relevant item could also lead to a negative utility due to the wrong time. We consider relevant items at the right time as *good* recommendations. On the other hand, we consider irrelevant items or relevant item at the wrong time as *bad* recommendations. We explore how to use this tenure-based decision probability in two scenarios.

Push-based Scenario In this scenario, the recommender system pushes items to the user proactively regardless of whether the user comes to the website. The recommendations could be sent to the user by email or other campaign methods. The challenge is to determine the right time to make relevant recommendations to maximize the user utility/satisfaction.

In Table 1, we show the utility set of the recommender system for different types of recommendation. The utility of model q for a set of recommendation items g is calculated with Equation 18:

$$utility_g(q) = \sum(u_{TP} I_{show,accept} + u_{FP} I_{show,\bar{accept}} + u_{FN} I_{\bar{show},accept} + u_{TN} I_{\bar{show},\bar{accept}}) \quad (18)$$

I_* is the indicator function where $I_* = 1$ if $*$ is true. The recommendation threshold [7] $Thres_{rec}$ is automatically determined as [2]

$$Thres_{rec} = \frac{u_{FP} - u_{TN}}{u_{FP} - u_{TN} + u_{FN} - u_{TP}} \quad (19)$$

where "reasonableness conditions" assume $u_{FP} < u_{TN}$ and $u_{FN} < u_{TP}$. It indicates that the utility of a right label is always higher than the utility of a wrong label.

The system could leverage the tenure-based decision probability as a signal in the process. If the tenure-based decision probability of an item is greater than the threshold $Thres_{rec}$, the recommendation is presented to the user. Otherwise, it is not presented.

Pull-based Scenario In this scenario, the recommender systems selects a set of items to recommend to the user when the user comes to the website. The goal is to select most relevant items to present to the user. A natural approach is to incorporate the tenure-based decision probability of an item for a user as an additional feature in the hybrid recommender system. This feature indicates a user's aspiration to make the job transition to the item at the recommendation time. The hybrid system first ranks all candidate items based on

[2]Mathematical details can be referred to the paper [7].

all features that are associated with the user and the item, then selects the top items to present to a user.

5. EVALUATION OF HAZARDS MODEL

In this section, we evaluate the performance of the hierarchical proportional hazards model in predicting the transition time. We first list all research questions that we intend to answer, followed by the experimental setup and performance analysis. Major research questions include:

- How accurate is the tenure-based decision probability that is predicted by the hazards model? Will the model predict a higher probability when the user is likely to transit to new job and a lower probability when the user is not likely to make transitions?
- How accurate is the predicted decision time, compared to the actual decision time in the data?
- Is it important to consider covariates that are associated with the transition? Does the hierarchical proportional hazards model make more accurate prediction by taking covariates into consideration?

5.1 Evaluation Metrics

Determining the right recommendation time is a relatively new research topic. We introduce two metrics to evaluate the accuracy of the tenure-based decision probability and its effect in the recommendation utility.

5.1.1 Perplexity/Likelihood

The first metric is the perplexity of the model. It is widely used in the evaluation of language models and speech recognition [3]. We assume that the testing data is drawn from the same probability distribution as the training data. After a probability model q is trained with the training data, the perplexity reflects how well model q predicts the testing data. The perplexity of the model q is defined as

$$perplexity(q) = \left(\prod_{m=1}^{M} \prod_{i=1}^{N_m} \frac{1}{q(y_{m,i}, \mathbf{x_{m,i}})} \right)^{\frac{1}{\sum_{m=1}^{M} N_m}} \quad (20)$$
$$= 2^{-\sum_{m=1}^{M} \sum_{i=1}^{N_m} \frac{1}{\sum_{m=1}^{M} N_m} log_2 q(y_{m,i}, \mathbf{x_{m,i}})}$$

where $q(y_{m,i}, \mathbf{x_{m,i}})$ is defined in Equation 15. As we can see, perplexity is the inverse of the probability. If model q gives higher data likelihood to transitions in the testing data, the corresponding $perplexity(q)$ would be lower. The lower the perplexity, the better the model.

5.1.2 Estimated Decision Time

The second metric is to compare the estimated decision time and the actual one. It is the same as comparing the estimated tenure $y_{m,i}$ and the actual tenure $\hat{y_{m,i}}$. After model q predicts the distribution of the tenure, we use the mean of the distribution as the estimated tenure $\hat{y}_{m,i}$. The absolute error of the estimation is given by $|\hat{y_{m,i}} - y_{m,i}|$. The mean absolute error (MAE) across all testing data can then be used for analysis and comparison. $MAE(q) = \frac{1}{\sum_{m=1}^{M} N_m} \sum_{m=1}^{M} \sum_{i=1}^{N_m} |\hat{y_{m,i}} - y_{m,i}|$. The smaller the MAE, the better the model.

5.2 Models to Compare

In our experiments, we compare the following models.

H-One is the hazards model that fits a **single** set of parameters with no covariates (i.e., the basic Weibull distribution) to the tenure data. All transitions $m = \{* \rightarrow *\}$ share the same parameters, regardless of the transition's source a and destination b.

H-Source is the hazards model that fits **multiple** sets of parameters with no covariates to the tenure data. All transitions $m = \{a \rightarrow *\}$ from source a share the same parameters, regardless of the destination b.

H-SourceDest is the hazards model that fits **multiple** sets of parameters with no covariates to the tenure data. All transitions $m = \{a \rightarrow b\}$ from source a and destination b share the same parameters. It only uses the tenure information with no covariates.

H-SourceDestCov further incorporates covariates into the hazards model in *H-SourceDest*. In this case, the probability density function and the underlying hazards function change for each user u at time t if values of the associated covariates change. In this paper, we use the following covariates: 1) about the user u: the user's gender, age, number of connections, number of jobs that the user has changed, average months that the user changes a job; 2) about the item j_a or j_b: discretized company size, the company age, i.e., the current year minus the year the company was founded; 3) about the relationship between j_a and j_b: the ratio of the company size, the ratio of the company age; whether j_a and j_b are in the same function, whether they are in the same industry; 4) about the user's aspiration of category b: number of job applications from user u in category b in the last week, last month, last two months, and last three months. We choose these covariates to show the effect of the proportional hazards model. Extensive feature engineering can be applied here to include more useful features/covariates which could be explored in future work.

For all models, we learn parameters for a job transition if the transition is performed by at least k_u unique users in the training data. k_u is set to be 5 in the experiment. If parameters of transitions $m = \{a \rightarrow b\}$ are not learned in the training process, average values of parameters of transitions $m = \{a \rightarrow *\}$ are used to represent parameters of $m = \{a \rightarrow b\}$. If all parameters of transitions $m = \{a \rightarrow *\}$ are not learned in the training process, average values of parameters of transitions $m = \{* \rightarrow *\}$ are used to represent parameters of $m = \{a \rightarrow *\}$. In the prediction step, we smooth the probability estimation $q'(y_{m,i}, \mathbf{x_{m,i}}) = \max(\text{minThres}, q(y_{m,i}, \mathbf{x_{m,i}}))$ to avoid having a probability that is too low. minThres is set as 0.001. $\Delta(t)$ in Equation 15 is set as 3 (i.e., 3 months) for all models during the prediction step.

5.3 Dataset

As positioned earlier in the document, we apply our techniques in the context of a job recommender system and use a real-world dataset from LinkedIn to evaluate our models. We first analyze the user's changing job behavior in recent

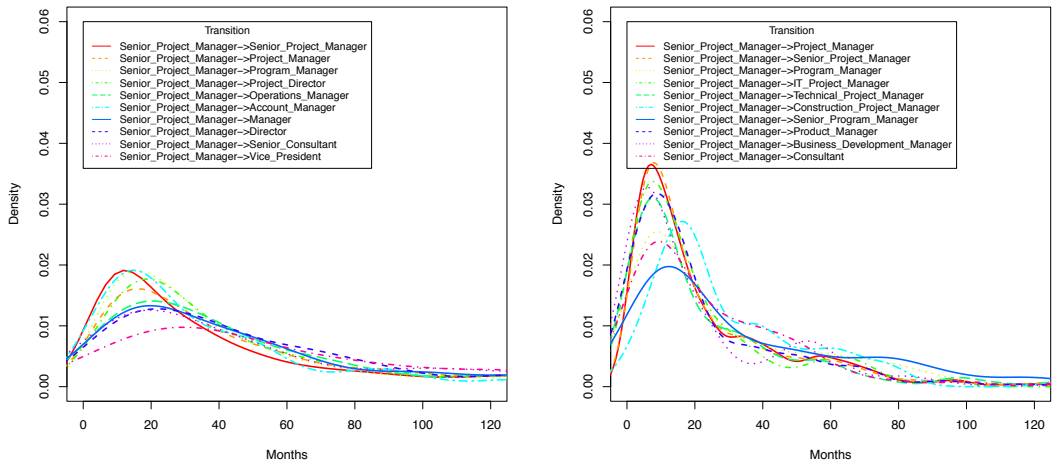

Figure 4: Density plot of the tenure before job transitions from *senior project managers*. (a) tenure before users make the job transition (b) tenure before users start to apply for the new job.

5 years to understand the role of the tenure in the job transiting process. Figure 4 shows the density plot of the tenure before a job transition from *senior project managers*. The first one shows the tenure before users make the transition and start the new job. It is clear that different job transitions usually happen at different tenures. For example, horizontal transitions from *senior project manager* to *senior project manager* are likely to happen at tenures of approximately 18 months while transitions from *senior project manager* to *vice president* are likely to happen at tenures of approximately 30 months. Users tend to transit to a higher position if they stay at the current position longer. The second plot in Figure 4 shows the tenure before users start to apply for new jobs. We notice that different job applications from the same job position also happen at different tenures. This justifies our motivation that we should take the time factor into consideration when making recommendations. The goal of the recommender system is to recommend jobs for users to apply. An actual job transition may or may not happen after a user applies to a job. Thus, we evaluate our models with the job application dataset and not necessarily the actual position transition data (which is also available). The dataset is composed of a sample of 11 million job applications over years. 10-fold cross validation is performed.

5.4 Performance Analysis

5.4.1 Perplexity/Likelihood

Here we compare the perplexity of all models in Section 5.2. As we describe before, the tenure-based decision probability $q(y_{m,i}, \mathbf{x_{m,i}}) = Pr(y_{m,i} < T \leq y_{m,i} + \Delta t | T > y_{m,i})$. The baseline model, *uniform*, assigns the uniform distribution to all tenures. The probability density function is $p(y_{m,i}) = \frac{1}{T_{uniform}}$ where $T_{uniform}$ is the number of tenures to consider. We set $T_{uniform} = 100$, i.e., 100 months.

First, we show the perplexity of all models from the 10-fold cross validation of the job application data in Table 2. In each job application, the user's job before the application and the job that the user applies to are available. It is clear that all hazards models have lower perplexity than the baseline *uniform*. It demonstrates the effect of modeling the tenure with the hazards model. Among all hazards models, *H-SourceDestCov* achieves the lowest perplexity, followed by *H-SourceDest*, *H-Source*, and *H-One*. *H-SourceDestCov* fits the parameters of the proportional hazards model for each $m = \{a \rightarrow b\}$ transition and incorporates related covariates. It shows the importance of considering covariates when modeling the tenure before a transition.

Second, we show the perplexity of different degrees of job seekers from a survey data. The data contains 9k LinkedIn users who were surveyed about their job seeking level in December 2011. All users categorized themselves into the following five categories from active job seekers to passive ones. 1)*Aggressively looking*: I'm actively looking for a new job and sharing my resume; 2) *Somewhat looking*: I'm casually looking for a new job 2-3 times per week to see what is available; 3) *Tiptoers*: I'm thinking about changing jobs and have reached out to close associates but am not actively looking; 4) *Explorers*: I'm not looking for a new job, but would discuss an opportunity with a recruiter to see if the job is interesting to me. 5) *Super passive*: I'm completely happy in my current job and am not interested in discussing any new job opportunities. The more passive the user, the lower the tenure-based decision probability. Because the perplexity is the inverse of the likelihood, a good model is expected to have higher perplexity for more passive users. In the survey data, the destination job or the job category that the user looked for is unknown. Thus, *H-SourceDest* and *H-SourceDestCov* that need the information of the destination are not included in the comparison. The perplexity of *H-One* and *H-Source* for different degrees of job seekers is shown in Figure 5. One can see that the perplexity increases for more passive job seekers and this trend is more pronounced with model *H-Source*. The figure also reveals that model *H-Source*, which learns multiple sets of parameters (one for each source category a), has better predictability for both job seekers and non-job seekers, compared to model *H-One*.

Table 2: Perplexity of different hazards models

Uniform	H-One	H-Source	H-SourceDest	H-SourceDestCov
22.47	9.17	8.11	7.27	**5.30**

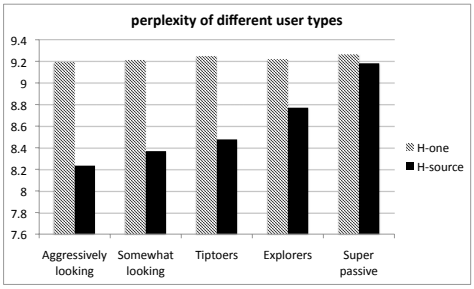

Figure 5: Perplexity of models for different degrees of job seekers.

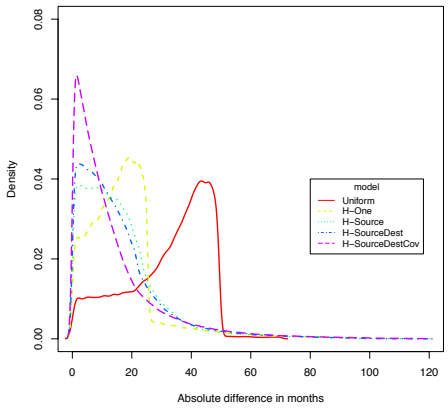

Figure 6: Density plot of the absolute difference between the estimated tenure and the actual tenure

5.4.2 *Estimated Decision Time/Tenure*

Here we compare the estimated tenure $\hat{y}$ with the actual tenure y before a transition. As before, the basic model *uniform* assigns the uniform distribution to all tenures. $p(y_{m,i}) = \frac{1}{T_{uniform}}$ where $T_{uniform}$ is set as 100 months. Hence, the corresponding estimated tenure is always 50 months.

In Figure 6, we show the density plot of the absolute error between the estimated tenure and the actual tenure, i.e., $|\hat{y} - y|$. The distribution of the absolute error of *H-SourceDestCov* is closest to 0. This is confirmed by Table 3 which reports the MAE between $\hat{y}$ and y. All hazards models perform better than the baseline *uniform* while *H-SourceDestCov* gives the most accurate estimation of the decision time/tenure, followed by *H-SourceDest*, *H-Source*, and *H-One*. In order to get a better estimated decision time, we plan to evaluate other estimators (such as the median) in the future work.

6. EVALUATION OF RECOMMENDATION MODEL

In this section, we incorporate the hazards model into the recommender system and evaluate its contribution in different scenarios.

6.1 In the Push-Based Scenario

6.1.1 *Experimental Setup*

In the push-based scenario, the goal is to determine the right time to make relevant recommendations to maximize the user utility. We use a sample of 6 million job impression data that were collected after the previous 11 million job application data. The hazards model is trained with the 11 million job application data and predicts the tenure-based decision probability for each impression in the 6 million impression dataset. An impression might lead to a job application or not, which corresponds to a *good* recommendation or a *bad* one. The impression item is selected by a hybrid recommender system with decent performance. The item is assumed to be relevant to the candidate user. Evaluations with two datasets are presented for each set. One dataset $Impression_{all}$ contains all sets of impressions, regardless of whether users applies to any of the impression in a set. The other dataset $Impression_{app}$ contains sets of impressions with at least one application, i.e., a user applies to at least one job in a set of impressions on that day.

Table 3: Mean absolute error (MAE) between the estimated tenure and the actual tenure

Uniform	H-One	H-Source	H-SourceDest	H-SourceDestCov
32.02	17.35	16.07	15.57	**14.25**

The evaluation metric is the utility of the testing data. The testing data consists of a set of item impressions that the system predicted to be relevant to user u at time t. The user then choose whether to accept the items presented by the system. Assume that there are G sets of impressions in the test data. The average utility $utility(q)$ can be calculated as following: $utility(q) = \frac{\sum_{g=1}^{G} utility_g(q)}{G}$ where $utility_g(q)$ for each set of impressions is calculated by Equation 18. Unlike traditional metrics such as precision@K and recall@K, $utility(q)$ considers both the positive effect for *good* recommendations and the negative effect for different types of *bad* recommendations. The higher the utility, the better the model. In addition, we compare the average number of recommendations and the recommendation coverage after filtering items with $q(y_{m,i}, \mathbf{x_{m,i}}) > Thres_{rec}$. Coverage is the percentage of sets of impressions that contain at least one recommendation. Models that have both high utility and high coverage are preferred.

6.1.2 *Performance Analysis*

Based on the tenure-based decision probability, the system decides whether or not to present an item (impression) to a user. The baseline model *AlwaysRec* shows all impressions to the user.

In the real world scenario, the customized utility set is determined by the application's usage and the users' tolerance for *bad* recommendations. In Table 4, we show three sets of utility that correspond to different scenarios.

In the first utility set, the utility u_{TP} is set to 20 when the system shows an impression and the user applies to it. When the system shows an impression and the user does not apply to it, the utility u_{FP} is -2. When the system does not show an impression but the user actually applies to it, the utility u_{FN} is set to -2. When the system does not show

Table 4: Utility of the model. $Impression_{all}$ contains all groups of impressions. $Impression_{app}$ contains sets of impressions with at least one application. In each utility set, the *first* line shows the average utility of the model for each set of impressions. The *second* line shows the lift of the model compared to the baseline *AlwaysRec*. The *third* line shows the average number of recommendations. The *fourth* line shows the coverage of recommendations.

u_{TP} = 20, u_{FN} = -2
u_{FP} = -2, u_{TN} = 0
Threshold = 0.083

Data	*AlwaysRec*	*H-One*	*H-Source*	*H-SourceDest*	*H-SourceDestCov*
$Impression_{all}$	-41.38	-41.38	-37.89	-38.02	-37.97
		(0.00)	**(3.49)**	(3.36)	(3.41)
	(24.12)	(24.12)	(23.11)	(22.79)	(22.62)
	(100%)	(100%)	(91.28%)	(97.20%)	(99.13%)
$Impression_{app}$	-31.45	-31.45	-29.06	-29.15	-29.07
		(0.00)	**(2.39)**	(2.30)	(2.38)
	(25.76)	(25.76)	(24.73)	(24.39)	(24.20)
	(100%)	(100%)	(91.59%)	(97.44%)	(99.23%)

u_{TP} = 20, u_{FN} = -10
u_{FP} = -2, u_{TN} = 0
Threshold = 0.063

Data	*AlwaysRec*	*H-One*	*H-Source*	*H-SourceDest*	*H-SourceDestCov*
$Impression_{all}$	-41.38	-41.38	-40.67	-40.30	-39.65
		(0.00)	(0.71)	(1.08)	**(1.73)**
	(24.12)	(24.12)	(23.89)	(23.73)	(23.55)
	(100%)	(100%)	(98.06%)	(98.88%)	(99.59%)
$Impression_{app}$	-31.45	-31.45	-31.12	-30.91	-30.51
		(0.00)	(0.33)	(0.54)	**(0.94)**
	(25.76)	(25.76)	(25.54)	(25.37)	(25.18)
	(100%)	(100%)	(98.18%)	(98.95%)	(99.62%)

u_{TP} = 20, u_{FN} = -2
u_{FP} = -10, u_{TN} = 0
Threshold = 0.313

Data	*AlwaysRec*	*H-One*	*H-Source*	*H-SourceDest*	*H-SourceDestCov*
$Impression_{all}$	-231.83	-0.62	-15.79	-26.81	-71.37
		(231.21)	(216.04)	(205.02)	(160.46)
	(24.12)	(0.00)	(0.78)	(1.43)	(2.91)
	(100%)	(0%)	(6.45%)	(26.47%)	(66.93%)
$Impression_{app}$	-230.24	-1.82	-17.61	-28.90	-73.60
		(228.42)	(212.63)	(201.34)	(156.64)
	(25.76)	(0.00)	(0.87)	(1.59)	(3.21)
	(100%)	(0%)	(6.69%)	(28.15%)	(68.55%)

an impression and the user does not apply to it, the utility u_{TN} is 0. In this case, users are quite tolerant to *bad* recommendations with u_{TP} being much higher than u_{FP}. Model *H-One* achieves the same utility as *AlwaysRec*, while the other three models achieve better utility than the baseline. *H-Source* has a slightly better utility yet it has the lowest coverage. In *H-One*, a single Weibull distribution is fitted to all data from all transitions. The fitted Weibull distribution has shape $\gamma = 0.978$ and scale $\theta = 25.93$. As shown in Equation 3, the hazards function is $h(y) = \gamma\theta y^{\gamma-1}$, which is pretty stable for different tenure values y when $\gamma = 0.978$. In the experiment, Δt is set as 3 months in Equation 15. The resulting tenure-based decision probability of *H-one* is around 0.10 to 0.11 for different tenure values. Given that the recommendation threshold is 0.083 in this utility set, all impressions are therefore shown to the user in model *H-one*. Thus, it performs the same as the baseline model *AlwaysRec*.

In the second utility set, the utility u_{FN} is set to -10, indicating more utility penalization when the system does not show an impression yet the user would actually apply to it. In this scenario, *H-SourceDestCov* wins with the highest recommendation utility and coverage, followed by *H-SourceDest*, *H-Source*, *H-One*, and *AlwaysRec*. This indicates that *H-SourceDestCov* has better predictability of whether the user applies to a job at the recommendation time.

In the third utility set, the utility u_{FP} is set to -10, indicating that the user does not tolerate **bad** recommendations. In this case, the best model does not show any recommendation, as in the case of model *H-one* with coverage of 0%. All other models have higher utility than *AlwaysRec* but less utility than *H-one*. There is a tradeoff between the utility and the recommendation coverage, which can be tuned in a real-world application.

We discover that the overall utility of the best approach is still negative. There are two possible reasons behind it. 1) The ratio between *bad* items and *good* items is highly unbalanced in the impression dataset. It is challenging for the recommender system to keep *good* items while filtering out most *bad* items. 2) It is true that the user's decision to accept a relevant item is not purely dependent on the time. Recommender systems also take related factors [6] into account. A potentially better filtering signal is to combine the tenure-based decision probability with other probabilities, such as the probability of an item being relevant, the probability of an item being in the right location, etc.

6.2 In the Pull-Based Scenario

6.2.1 Experimental Setup

In the pull-based scenario, the goal is to select most relevant items to present to the user when the user comes to the site. The recommendation model is trained with a sample of millions of job application data in one month and tested with a sample of job application data in the following two weeks.

We evaluate the recommender system in the context of a ranking task [4]. For each job application, we first apply a heuristic filter to all open jobs available that day. First, only open jobs that are in the same geographical region as the user, for example, *San Francisco bay area*, are retained. Second, jobs must have a seniority level comparable to the user's current position seniority. In other words, the system won't recommend entry-level jobs to users in a senior position. All potential jobs that remain after the heuristic filter are used in the ranking step. Similar to other work in recommender systems [4, 28, 34], we use the commonly used IR metrics, precision@K of all testing cases, to compare models. The task focus is on evaluating a model's performance on ranking relevant items in top positions while ignoring the negative effect of different types of *bad* recommendations. The number of recommendations is fixed to be K.

In the experiment, **BasicModel** with basic features serves as the baseline. Basic features include similarity-related features between the user profile (including the user's working experience, education information, etc) and the job information (including the job's title, description, etc). The following feature groups are compared to the baseline. **Basic+TranProb** adds the smoothed transition probability in addition to basic features. The smoothed transition probability from j_a to j_b is calculated as following: $P(j_a \to j_b) = \frac{\#\ j_a \to j_b + \lambda}{\sum \#\ j_a \to * + J \cdot \lambda}$. $\#\ j_a \to j_b$ is the number of transitions from j_a to j_b. $\sum \#\ j_a \to *$ is the number of all transitions from j_a. J is the number of jobs and λ is the smoothing factor, which is set as 0.1. **Basic+TranProb+Tenure** further adds the pure tenure value (such as 8 for 8 months) as an additional feature. Instead of using the pure tenure value, **Ba-**

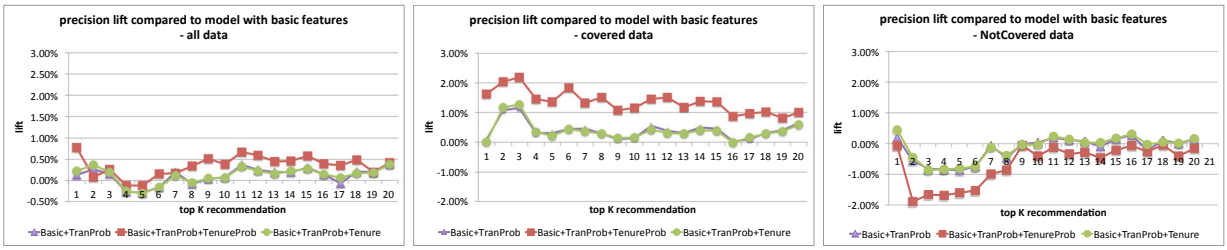

Figure 7: Precision lift of the hybrid recommender system with different feature groups. (a) performance of all testing data (6435 cases) (b) performance of testing cases that are covered by top transitions (47.7% of all cases) (c) performance of testing cases that are not covered by top transitions (52.3% of all cases)

sic+TranProb+TenureProb uses the tenure-based decision probability from hazards model *H-SourceDestCov*.

In practice, there are several mechanisms for building hybrid recommender systems [2, 29] to use. We use logistic regression, which is a common practice in industry [17]. It achieves decent performance with low computational complexity.

6.2.2 *Performance Analysis*

We compare the precision lift of different feature groups by using **BasicModel** as the baseline in Figure 7. We show the precision lift of all testing cases, cases that are covered by top transitions, and cases that are not covered. Suppose that the testing user is working as j_a and applies to j_b. This testing case is covered by top transitions if more than k_u unique users working with jobs in category a applied to jobs in category b in the training data. k_u is set to be 5. For example, common destinations of *software engineers* include *senior software engineers*, *technical leads*, *consultants*, etc. If a user as *software engineer* does apply to a job in one of these common destinations, the case is covered by the top transitions. Otherwise, it is not covered.

In the first plot in Figure 7, we observe a 0.5% to 1% lift of all testing cases by adding **TranProb** and **TenureProb** features. The difference between the model with basic features and the one with **Basic+TranProb+TenureProb** is significant after the top 9 recommendations ($p \leq 0.05$).

In the second plot in Figure 7 with testing cases that are covered by top transitions, it is clear that the model with **Basic+TranProb+TenureProb** performs the best. The model with **Basic+TranProb** features gives 0.21% lift in precision@5, compared with the baseline **BasicModel**. The model with **Basic+TranProb+TenureProb** gives 1.36% lift in precision@5. The difference between the model with basic features and the one with **Basic+TranProb+Tenure Prob** is significant for all top K positions ($p \leq 0.05$). On the other hand, the model with the pure tenure value **Basic+TranProb+Tenure** does not give further improvement, compared to the model with **Basic+TranProb**. We observe that the pure tenure value is noisy for hybrid systems that do not capture interactions among features. First, it is independent of the source job, which is the user's current job. The same tenure value for users working in different jobs does not indicate the same aspiration of changing jobs. Secondly, it is independent of the destination job. However, the reality is that the same tenure value indicates different level of aspiration for different destination jobs. Thus, it is essential to use the tenure-based decision probability instead of the pure tenure value in hybrid systems such as logistic regression models. If more advanced hybrid systems, such as gradient boosted tree algorithms are used to capture interactions among features, pure tenure values might perform similarly as the tenure-based decision probability.

In the third plot in Figure 7 with testing cases that are not covered by top transitions, we observe that incorporating **TranProb** and **TenureProb** features hurts the performance a little. It is not surprising because the transition probability and tenure-based decision probability reflect transitions that are shared by most users. If a user has her own career plan, such as transiting to be a *novel writer* after working as a *software engineer* for *two* years, it won't be captured by the transition probability or the tenure-based decision probability. Instead, these transitions need to be captured by the user's behavior signals, such as job searches, job clicks, etc. Besides such job applications are more likely to happen after a user proactively searches the system. This is an important problem to analyze and study, but beyond the scope of this paper.

7. CONCLUSION AND FUTURE WORK

We performed research to answer the following question: When is the right time to make a job recommendation and how do we use this inference to improve the utility of a job recommender system? We proposed using the hierarchical proportional hazards model. Experiments with the real-world job application data demonstrated the effectiveness of the hazards model and the importance of considering the time factor in the recommendation process. This was just the first step in exploring the right time to make the recommendation. More interesting models to leverage the tenure information could be studied and compared with the hazards model. We plan to also explore other approaches to use the tenure-based decision probability and evaluate it in other domains beyond job recommendations as well.

Acknowledgments

We would like to thank Ethan Zhang in LinkedIn for his help and comments. Part of this work was funded by National Science Foundation IIS-0953908 and CCF-1101741. Any opinions, findings, conclusions or recommendations expressed in this paper are the authors, and do not necessarily reflect those of the sponsors.

8. REFERENCES

[1] J. Bennett and S. Lanning. The netflix prize. 2007.

[2] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov. 2002.

[3] S. Chen, D. Beeferman, and R. Rosenfeld. Evaluation metrics for language models. In *DARPA Broadcast News Transcription and Understanding Workshop (BNTUW)*, Lansdowne, Virginia, USA, Feb. 1998.

[4] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM RecSys 2010*, pages 39–46, New York, NY, USA, 2010. ACM.

[5] M. Dubinko, R. Kumar, J. Magnani, J. Novak, P. Raghavan, and A. Tomkins. Visualizing tags over time. *ACM Trans. Web*, 1(2), Aug. 2007.

[6] P. Dütting, M. Henzinger, and I. Weber. Maximizing revenue from strategic recommendations under decaying trust. In *Proceedings of the 21st ACM international CIKM'12*.

[7] C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the 17th IJCAI'01*.

[8] N. Golbandi, Y. Koren, and R. Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *Proceedings of the fourth ACM WSDM'11*.

[9] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd ACM SIGIR*, pages 230–237, New York, NY, USA, 1999. ACM.

[10] R. Jin, L. Si, C. Zhai, and J. Callan. Collaborative filtering with decoupled models for preferences and ratings. In *Proceedings of the twelfth CIKM*, pages 309–316, New York, NY, USA, 2003. ACM.

[11] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[12] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD*, 2009.

[13] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Recommendation systems: A probabilistic analysis. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, FOCS '98, pages 664–, Washington, DC, USA, 1998. IEEE Computer Society.

[14] C. Liu, R. W. White, and S. Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *Proceedings of the 33rd ACM SIGIR'10*.

[15] B. Marlin and R. S. Zemel. The multiple multiplicative factor model for collaborative filtering. In *Proceedings of the 21st ICML '04*.

[16] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *DL '00*, pages 195–204, New York, NY, USA, 2000.

[17] D. Parra, A. Karatzoglou, X. Amatriain, and I. Yavuz. Implicit feedback recommendation via implicit-to-explicit ordinal logistic regression mapping. In *Proceedings of the CARS-2011*, 2011.

[18] D. Parra-Santander and P. Brusilovsky. Improving collaborative filtering in social tagging systems for the recommendation of scientific articles. *Web Intelligence and Intelligent Agent Technology*, 1:136–142, 2010.

[19] M. Pazzani, D. Billsus, S. Michalski, and J. Wnek. Learning and revising user profiles: The identification of interesting web sites. In *Machine Learning*, pages 313–331, 1997.

[20] C. D. R. Regression models and life tables. *Journal of the Royal Statistic Society*, B(34):187–202, 1972.

[21] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th WWW '10*.

[22] E. J. Ruiz, V. Hristidis, C. Castillo, A. Gionis, and A. Jaimes. Correlating financial time series with micro-blogging activity. In *Proceedings of the fifth ACM WSDM '12*.

[23] E. Shmueli, A. Kagian, Y. Koren, and R. Lempel. Care to comment?: recommendations for commenting on news stories. In *Proceedings of the 21st WWW'12*.

[24] M. Shokouhi and K. Radinsky. Time-sensitive query auto-completion. In *Proceedings of the 35th international ACM SIGIR'12*.

[25] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *Proc. of ICML*, pages 704–711. AAAI Press, 2003.

[26] M. D. Smucker and C. L. A. Clarke. Modeling user variance in time-biased gain. In *Proceedings of the Symposium on Human-Computer Interaction and Information Retrieval*, HCIR '12, pages 3:1–3:10, New York, NY, USA, 2012. ACM.

[27] J. Wang, B. Sarwar, and N. Sundaresan. Utilizing related products for post-purchase recommendation in e-commerce. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 329–332, New York, NY, USA, 2011. ACM.

[28] J. Wang and Y. Zhang. Utilizing marginal net utility for recommendation in e-commerce. In *Proceedings of the 34th international ACM SIGIR'11*, pages 1003–1012. ACM, 2011.

[29] J. Wang, Y. Zhang, and T. Chen. Unified recommendation and search in e-commerce. In *Information Retrieval Technology*, pages 296–305. Springer Berlin Heidelberg, 2012.

[30] L. J. Wei. The accelerated failure time model: A useful alternative to the cox regression model in survival analysis. *Statistics in Medicine*, 11(14-15):1871–1879, 1992.

[31] R. W. White, P. Bailey, and L. Chen. Predicting user interests from contextual information. In *Proceedings of the 32nd international ACM SIGIR'09*.

[32] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD*, KDD '10, pages 723–732, New York, NY, USA, 2010. ACM.

[33] D. Zhang, J. Lu, R. Mao, and J.-Y. Nie. Time-sensitive language modelling for online term recurrence prediction. In *Proceedings of the 2nd ICTIR '09*.

[34] G. Zhao, M. L. Lee, W. Hsu, and W. Chen. Increasing temporal diversity with purchase intervals. In *Proceedings of the 35th international ACM SIGIR'12*.