A Comparison of Machine-Learned Survival Models for Predicting Tenure from Unstructured Résumés

Corné de Ruijt
Faculty of Science
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands
Email: c.a.m.de.ruijt@vu.nl

Vladimer Kobayashi
Faculty of Economics and Business
University of Amsterdam
Amsterdam, the Netherlands
Email: v.kobayashi@uva.nl

Sandjai Bhulai
Faculty of Science
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands
Email: s.bhulai@vu.nl

***Abstract*—This paper explores to what extent job seekers' future job tenures can be predicted using only the information contained in their own résumés. Here, job tenure is interpreted as the time spent in a single job occupation. To do so, we compare the performance of several machine-learned survival models in terms of multiple error measures, including the Brier score and the C-index. The results suggest that ensemble methods, such as random survival forest and Cox boosting, work well for this purpose. We further find that in particular time-related features, such as the time a person has already worked in a particular field, are predictive when predicting the person's future tenure. However, the results also show that this prediction task is difficult. There is substantial subjectivity in both how job seekers define their jobs, and at what level of granularity they indicate their job tenures. As a result, the best performing models (survival ensemble methods) only perform marginally better than the used benchmark (a Kaplan-Meier estimate).**

***Keywords*—Human resource management; turnover prediction; résumé mining; machine-learned survival models; job churn.**

I. Introduction

Given the high Internet penetration of job seekers, one could expect it to become easier for recruiters to find and select potential candidates. The reality, however, sometimes turns out to be different. Early studies on online recruitment (also known as e-recruitment) reported profitable benefits for recruiters, including an increased speed of hiring, or an improved quality applicants. However, they also reported the problem of having to sift through a (sometimes) overwhelming number of candidates [1].

Many of the methods proposed in the literature that assist in matching job seekers and vacancies online, use the semantic overlap between the résumé and the vacancy as a proxy for the quality of this match [2]. This, however, neglects other types of information contained in résumés that could provide information about the quality of the match. In this paper, we will instead use the temporal data often contained in résumés. Most job seekers indicate their job history in their résumé, in which their previous occupations are listed, along with a start and end date for each job. Our aim is to predict job tenures, defined as the time difference between these start and end dates, using other data that is contained in the résumé. This data includes features such as the type of job, education history, and the number of years of experience.

Predicting future job tenure from résumés is not a new problem. In fact, it has been the subject of many studies in personnel psychology in the last few decades [3, Ch. 12]. The problem we consider, however, differs from these studies in two ways: 1) we automatize the processes of extracting features from the résumé, both using a pre-trained résumé parser, and by using word2vec. 2) We focus on methods that emphasize on making accurate predictions, mostly by incorporating a large number of second or larger order interaction terms, rather than models that emphasize on explanations.

This paper has the following structure. Section II discusses related work. Sections III-A and III-B discuss properties of the résumé dataset, with a focus on properties of such unstructured datasets that may lead to biased results, and it considers how to avoid such bias. Section III-C introduces the survival models used in this study, and discusses how to measure the error of these models. Section IV presents the outcomes of the survival model comparison from different perspectives. Section V draws a final conclusion and provides directions for further research.

II. Related work

With the digitization of résumés, the automatic extraction of features from (manually written) résumés has become more common practice (e.g., [4], [5]). Apart from its application in job or candidate search engines, such features can also be used in job or candidate recommender systems. In both applications, the semantic overlap between the résumé and vacancy is often used as a proxy to evaluate how well the job seeker and vacancy match [2]. We will refer to the problem of matching job seekers and vacancies as *job seeker - vacancy matching*. Hence, since most literature considers the problem from a semantic perspective, we deliberately focused on non-semantic methods. In particular, we consider the potential of predicting how long someone will stay in a new job position, given that the candidate would be hired for the position, as a measure for the quality of the match.

 ISBN: 978-1-61208-891-4

Few studies consider the sequential and time-based elements in a résumé, which in particular present themselves in the job history section. In résumés, it is common to write down one's previous jobs in chronological order, and indicate the start and end date of each job. This information could be used to infer a job seeker's most likely next job, given a sequence of previous jobs. Such perspective to job seeker - vacancy matching has been considered by various contributions in the literature [6]–[9]. Li et al. [9] compared several sequential models for this task, where in particular a Long Short-Term Memory (LSTM) recurrent neural network worked well, especially when additional contextual data was included in the model.

From the start and end dates, one could also infer how long job seekers will be likely to remain in their jobs. For this problem, survival analysis has been a frequently used method in personnel psychology, in particular in the form of Cox regression [10]. Survival models often allow for censored data, making these models attractive for studying turnover. I.e., job seekers included in a study might still be "alive", or in other words still occupying the job under study, at the end of the study. Even though there has been a substantial increase in the number of studies applying machine learning methods for predicting employee turnover, only a few consider combining machine learning methods with survival analysis [11].

Wang et al. [12] propose a survival model that is fitted using a Bayesian model. The Bayesian model was chosen to cope with the high dispersion in the number of observations for a job transition from some job a to job b. I.e., most transitions have little to no observations, whereas some self-transitions may be very frequent. The authors show that if one is indifferent about to which job the job seeker switches, and only considers how long the job was occupied, the Bayesian model outperforms a model ignoring covariates in terms of perplexity. Though, this difference between the with/without covariates models evaporated when considering more passive job seekers.

Li et al. [8] discretize time and predict a value proportional to the survival function using a squared loss function. As the predicted values are only proportional to the probability of remaining in a job, the study considers the correct order of turnover events, rather than predicting tenure. The method outperformed typical parametric or semi-parametric survival methods such as a Cox regression or the log-logistic model.

III. Methods

A. Feature extraction from résumés

The data used in this study was extracted from résumés, which were uploaded to the Dutch job board Gus [13] between 2005-01-01 and 2016-10-17. The jobs to which these applicants applied were temporary jobs. In total, the dataset contains 50,000 unique job seekers, which we split into a training, validation and test set according to a 70/10/20 split. I.e., all jobs from one job seeker are either completely in the training, test, or validation set.

In total, the dataset encompasses 131,059 unique jobs. Note that the start and end dates of these jobs may be outside of the 2005-2016 range. E.g., if the job seeker applied in 2005, he/she will be likely to have jobs in his/her resume before 2005. To avoid data dredging, all statistics presented in this section are based on the training set.

Since the résumés are plain text documents, we used technology from Textkernel [14] to extract information from the text. Here, we make use of a common convention in résumés to include job history in a table, where each record includes a (textual) description of the job, and the start and end date of the job. From this data, we extracted the variables *transition lustrum* (the year in which job seeker j started job h, grouped into clusters of 5 years), *order* (the number of previous jobs job seeker j had occupied just after starting job h), and *expdays* (the total observed work experience of candidate j, just before the start of job h, in days).

We also extracted the *edu_lvl* (the candidate's highest education level, mapped to the Dutch education system), *age* (candidate's age at the start of the job), *gender*, and the *job description* given by the candidate. The job description was mapped to a vector space in two ways. The first approach used a classification model from Textkernel, which maps the job to a three-layer hierarchical classification (of which the upper two were used as covariates) and classifies the *industry* of the job.

We also trained a *word2vec* model on the candidates' previous job description. Before training the *word2vec* model, we removed (Dutch) stop words and stemmed the words using the *Snowball* stemmer [15]. We used a vocabulary of 20,000 unique words having the largest *tf-idf* values. We used negative sampling with a sampling factor of 10^{-5}, from which word pairs were constructed using skip grams with a window size of 3, as larger window sizes did not improve the results.

The *word2vec* model was trained using Keras with a Tensorflow backend [16], [17]. We used an embedding size of 64; 100 training epochs; a batch size of 65,536; an initial learning rate of 0.1; and we used *rmsprop* [18] to update the learning rate in subsequent epochs. To obtain document vectors from word vectors, we computed a weighted average over the word vectors for each job description. As weights we used the *tf-idf* value of each word. We did experiment with different epochs, batch sizes, and initial learning rates; these did not improve the results.

B. Computing tenure from parsed résumé data

From analyzing the job seekers' start and end dates, one can readily observe that job seekers tend to indicate the start and end date of each job at different levels of granularity. Some indicate the start and end dates on a monthly level, whereas the majority indicate these dates on a yearly level. In case candidates indicate their start and end dates on a yearly level, we considered this a case of interval-censored data. Besides the interval censoring, the start and end dates also may incorporate other types of censoring.

The survival models we will introduce in Section III-C only cope with right censoring. Hence, to cope with other types of censoring in the tenures, the following procedure was applied.

 ISBN: 978-1-61208-891-4

All observations with *Complete* (no start and end date; 1.62%) and *Left FJ* (no start date, and the job is the First Job of the candidate; 0.03%) censoring were removed. The former were removed because they are non-informative. The latter were removed because these only encompass a small number of jobs. *Year interval* (44.18%) indicates jobs with rounded start and/or end dates to entire years. If this was the case, a random number of months were added (subtracted) to the start (end) date, following the monthly turnover distribution. Since the observed turnover in the months January and December was inflated, due to the interval censoring, we did not use the observed turnover frequencies in these months to estimate the monthly turnover distribution. Instead, we estimated the turnover probability in these months by interpolation, using a cubic spline over the remaining months.

Right censoring, Not Last Job (*Right NLJ*; 10.3%), and Right censoring Last Job (*Right LJ*; 5.87%) indicate cases of right censoring. In case of *Right NLJ*, the start date of the next job was taken as end date of the job. In case this start date was again year interval-censored, the job was relabeled as year interval-censored and processed accordingly. *Right LJ* were treated as normal cases of right censoring, using the date of application as the date of censoring. 38% of all jobs did not have any type of censoring, hence remained in the dataset as-is. Although theoretically other types of censoring could have occurred (e.g., *Left NFJ*), these did not occur in the dataset.

Besides removing and correcting censored data, we also removed observations having occupations with tenures lasting longer than 50 years, occupations that started before the candidate's 18th birthday, occupations that started after the candidate's 67th birthday, and observations with negative tenures. To reduce the number of unique values for categorical attributes, we reassigned categorical values with fewer than 30 observations to a category "other". Missing data was imputed using adoptive tree imputation, as described by Ishwaran et al. [19], and which is implemented in the `RandomForestSRC` R package [20]. To fit this random forest model, the package's default parameters were used.

C. Survival estimation methods

1) Notation: Before discussing the survival models, we require some notation. Let T_i be the observed job tenure of job $i = 1, \ldots, I$, which is computed following the procedure described in Section III-B. Although we corrected for different types of censoring, T_i may still be right-censored. Whether this is the case, is indicated by δ_i (1 if not censored, 0 otherwise).

Furthermore, let $\tilde{T}_i$ be the full job tenure. That is, the job tenure we would have observed if no censoring had occurred. We are interested in estimating the survival function $S_i(t) = \mathbb{P}(\tilde{T}_i > t|\mathbf{x}_i)$. Here, $\mathbf{x}_i \in \mathbb{R}^P$ is some covariate vector. We assume independence between $\tilde{T}_i$ and $\tilde{T}_j$, given covariate vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. I.e., $\mathbb{P}(\tilde{T}_i, \tilde{T}_j|\mathbf{x}_i, \mathbf{x}_j) = \mathbb{P}(\tilde{T}_i|\mathbf{x}_i)\mathbb{P}(\tilde{T}_j|\mathbf{x}_j)$, for all pairs (i, j): $i \neq j$. Furthermore, we assume $\tilde{T}_i$ to be independent of the censoring time. Note that, due to right censoring, $\tilde{T}_i$ may not be completely observed. Hence, survival estimation methods use the (possibly right-censored) job tenure T_i, and censoring indicator δ_i, to estimate the uncensored survival distribution.

To estimate the survival function, we will frequently use the cumulative hazard function $\Lambda_i(t) = \int_{\tau=0}^{t} \lambda_i(\tau)d\tau$ with $\lambda_i(t) = \lim_{\Delta t \to 0} \mathbb{P}(t \leq \tilde{T}_i \leq t + \Delta t|\tilde{T}_i > t, \mathbf{x}_i)$ being the hazard rate. From the cumulative hazard rate, the survival function can directly be derived [21, p. 16].

Some of the models that we consider assume discrete time. To discretize time, we bin time intervals into bins $r = 1, \ldots, R$, each having equal length ρ. Since the number of jobs having tenures longer than 5 years was sparse, we took $R = 5$. The time interval of period r is denoted by $u_r = [(1 - r)\rho, r\rho)$. To balance between sparsity within the bins (which happens for small ρ), and the precision of the estimate, we selected $\rho = 3$ months.

2) Benchmark models: All machine-learned survival models presented in this paper were benchmarked against three benchmark methods: 1) a Kaplan-Meier (*KM*) estimate [21, Ch. 4], 2) a Cox proportional hazard model with an elastic net penalty [22] (we named this *Cox Lasso*, since using a Lasso penalty produced the best results). The baseline hazard was estimated using the Brewlow estimator [23]. 3) A binary survival tree (*Surv. tree*), using the log-rank splitting rule [19].

3) Ensemble survival models: A common approach to improve the quality of predictions from weak learners is by using model ensembles. In this study, we considered two approaches: the Random Survival Forest (*RSF*) introduced by [19], and a Cox boosting approach (*GBM*) [24].

A Random Survival Forest [19] for the most part employs the same procedure as the original random forest algorithm by Breiman [25]. Though, since we wish to predict a survival function, there are two main differences. First, as with the binary survival tree, the log-rank splitting rule is used to recursively branch the observations in the tree. Second, for each leaf node, the cumulative hazard rate is estimated using the Nelson-Aalen estimator, based on the observations in the leaf node. An estimate of the cumulative hazard rate for some time t and covariate vector $\mathbf{x}$ is then obtained by computing the unweighted average over all cumulative hazard rates at time t, for leaf nodes subject to $\mathbf{x}$.

To employ boosting, we used the boosting procedure by Friedman [26]. As the method employs Cox's partial likelihood, the method does not provide an estimate of the baseline hazard. To find the baseline hazard, the same procedure as for the Cox model was applied. That is, we use the Breslow estimator to estimate the baseline hazard, though we now used the output from the boosting model instead of the linear link function.

4) Neural survival models:

a) Feedforward neural survival models: To model neural survival models, we used a similar approach as Gensheimer & Narasimhan [27]. This study models the neural survival model as a feedforward neural network, only adjusting the output layer to produce a survival curve. To rewrite the problem, let $\gamma_{i,r} = 1$ if $T_i \in u_r, \delta_i = 1$ (zero otherwise), and $v_{i,r} = 1$ if $T_i < (r - 1)\rho$ (zero otherwise). The output layer of the

 ISBN: 978-1-61208-891-4

neural network is modeled in two ways. In the flexible variant (*NN-Flex*), a simple feedforward neural network is used with one or multiple hidden layers, applying a sigmoid activation to each output $r \in \{1, \ldots, R\}$ to end up with estimates of the hazard rate.

The proportional hazard (*NN-PH*) variant uses the proportional hazard assumption. I.e., it assumes the hazard rate has the form $\lambda_i(t) = \lambda_0(t)\exp(\mathbf{x}_i^T\boldsymbol{\beta})$, $\boldsymbol{\beta} \in \mathbb{R}^P$ being a weight vector. Following [21, p. 43], when assuming intrinsically discrete time, the (now also discrete) estimated hazard rate $\hat{\lambda}_i(r)$ can be written as

$$\hat{\lambda}_i(r) = \frac{1}{1 + \exp(\alpha_i(r) + z_i)}, \tag{1}$$

with z_i and $\alpha(r)$ being outputs of different feedforward neural networks. Here, $\alpha_i(r)$ has as input the covariate vector $\mathbf{x}_i$, followed by one or multiple hidden layers. The variable z_i is obtained by a weighted average over the elements in the last hidden layer. Note that (1) is a sigmoid activation function, which simplifies the implementation in, for example, Keras.

For both the NN-PH and NN-Flex approach, we find a loss function in the form of the binary cross-entropy

$$\mathcal{L} = \sum_{i=1}^{I}\sum_{r=1}^{R}[\gamma_{i,r}\log(\hat{\lambda}_i(r)) + (1-\gamma_{i,r})\log(1-\hat{\lambda}_i(r)v_{i,r})]. \tag{2}$$

Note that when $r\rho > T_i$ and $\delta_i = 0$, we have $y_{i,r} = 0$ and $\hat{\lambda}_i(r) = 0$. Therefore, these predictions do not contribute to the log-likelihood.

b) Recurrent neural networks: In addition to the two feedforward models, we also considered recurrent neural networks. Here, each output corresponds with one of the time periods $r = 1, \ldots, R$. We considered a standard recurrent neural network with either a Gated Recurrent Unit (*NN-GRU*) [28] or a Long Short-Term Memory (*NN-LSTM*) unit [29]. As we do not include time-varying covariates, only at $r = 1$ an input vector is inserted, whereas at the other time periods a vector containing only zeros is fed to the network. Also here we multiply (element-wise) the output vector $(\hat{\lambda}_i(1), \ldots, \hat{\lambda}_i(R))$ by the censoring vector $(v_{i,1}, \ldots, v_{i,R})$ to exclude observations after censoring.

D. Model evaluation

To evaluate the survival models, we used the Brier score to assess the accuracy of the survival curve [30], and the C-index [31] to assess the accuracy in correctly predicting the order of turnover. Since the number of observations at some time points was quite sparse, we decided not to use IPCW weights [32].

Since the C-index assesses the correct order of job churn at the start of both jobs, and it considers any job pair, even those unrelated, we also considered a somewhat altered C-index. This altered C-index, which we will refer to as the Integrated Conditional Concordance Index (ICCI), has two adjustments compared to the C-index. First, instead of assessing the correct order of survival estimates at some time t, it assesses the correct order of the expected remaining survival times, conditioned on survival up until times t_i, t_j for jobs i and j respectively. Second, we only sample over pairs in the same (*function_group*, *transitionlustrum*) bin. Since the number of observations in each bin differs, we use three types of sampling: 1) stratified sampling, 2) sampling the same number of observations from each bin, 3) sampling random pairs, ignoring the bins. As the name ICCI suggest, we integrate the conditional concordance indices over time.

TABLE I. HYPERPARAMETER GRID SEARCH

Model	Hyperparameters	Best param.
Cox-PH with elasticnet penalty	$\alpha \in \{0$ (Ridge), 0.5, 1 (Lasso)$\}$	$\alpha^* = 1$ (Lasso),
	penalty weight as in [33]	penalty$^* = 0.0107$
Survival tree	term. node size $= 6$	NA
Random survival forest	trees $\in \{100, 500, 1000\}$,	trees$^* = 500$,
	term. node size $= 6$,	
	depth $\in \{6, 12\}$,	depth$^* = 12$
	random split points $= 5$,	
	tree feature sample size $= \sqrt{P}$	
Cox boosting	trees $\in \{1000, 2500, 5000\}$,	trees$^* = 2,500$,
	shrinkage $\in \{0.001, 0.05, 0.01, 0.1\}$	shrinkage$^* = 0.01$
	depth $\in \{3, 6\}$	depth$^* = 3$
Neural survival non-sequential	hidden units $\in \{64, 128\}$,	hidden units Flex $^* = 128$
	hidden layers $= 2$,	hidden units PH $^* = 64$
	epochs $= 100$,	learning rate Flex$^* = 0.01$
	batch size $= 65,536$,	learning rate PH$^* = 0.001$
	learning rate $\in \{0.001, 0.01, 0.1\}$	
Neural survival sequential	time periods $= 21$,	hidden layers GRU$^* = 16$
	hidden layers $\in \{1, 4, 16\}$,	hidden layers LSTM$^* = 16$
	epochs $= 100$,	learning rate GRU$^* = 0.001$
	batch size $= 9,292$,	learning rate LSTM$^* = 0.001$
	learning rate $\in \{0.001, 0.01, 0.1\}$,	
	drop out $= 0.1$	

IV. RESULTS

A. Overall performance

Table I gives an overview of the grid search we applied to the validation set to find appropriate values for the models' hyperparameters. The obtained best parameter values are given in the last column of Table I. Our dataset contains some variables that could introduce unwanted discrimination in terms of *gender* and *age*. Instead of removing these attributes upfront, we included them while training the model, but imputed them by their overall average value (in case of categorical values, we imputed after dummification) during validation. However, it should be noted that this procedure was only partially effective, due to the many missing values for both *year_of_birthlustrum* and *gender*.

Figure 1 shows the resulting Brier and C-index on the test set, whereas Table II shows the integrated and normalized scores. Since the Kaplan-Meier estimate is the model with the least complexity (i.e., it does not take into account any covariates), the results of the KM model are emphasized in Figure 1. Gradient boosted trees and a random survival forest produce the best results, with a slight preference for GBM. Interestingly, the neural models and the Cox model barely outperform the Kaplan-Meier estimate both in terms of the Brier score and C-index. The single survival tree shows a trade-off between the Brier score and C-index. For $t < 3$ it shows reasonable performance in terms of the C-index, but the results are poor in terms of the Brier score.

The good performance of random survival forest and GBM seems to diminish when we include conditional survival times, as shown in Table II. Although in absolute terms the ICCI for

GBM and random survival forest are somewhat comparable to their C-index, the values are also closer to the results of a KM-estimator. Furthermore, taking different kinds of samples only had a marginal impact on the ICCI. Hence, when predicting the correct order, the advantage of using more complex models seems to diminish.

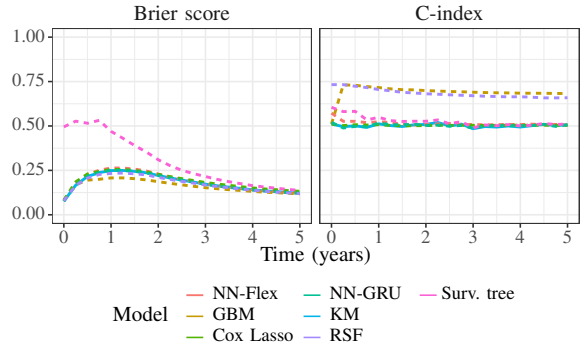

Figure 1. Brier score and C-index over time.

TABLE II. INTEGRATED BRIER SCORE, C-INDEX AND ICCI

Model	Integrated Brier	Integrated C-index	ICCI stratified	ICCI eq. per group	ICCI random
GBM	**0.16**	**0.69**	**0.66**	**0.66**	**0.67**
RSF	0.17	0.68	0.65	0.65	0.66
NN-Flex	0.19	0.51	0.64	0.64	0.63
NN-PH	0.18	0.51	0.64	0.65	0.63
NN-LSTM	0.18	0.51	0.63	0.64	0.64
NN-GRU	0.18	0.50	0.63	0.62	0.63
KM	0.18	0.50	0.64	0.64	0.64
Cox Lasso	0.19	0.50	0.64	0.65	0.63
Surv. tree	0.30	0.53	0.56	0.55	0.54

B. Performance on sub-datasets

Next, we split the results per *function_group* in order to study differences in predictive ability for different job types. As GBM had the best overall score (Table II), we used this model for further inference. The results over the five largest job types in the dataset are shown in Figure 2. As we may have expected from the Brier scores in Figure 1, which are somewhat similar to those of a Kaplan-Meier estimate, the fitted survival curves for the different job types are also rather similar.

We also considered the effect of excluding certain attributes from the model. To do so, we construct four sub-datasets: 1) a dataset in which age and gender were not imputed, 2) a dataset without the *word2vec* word embedding, 3) a dataset in which we exclude attributes derived from the job classifier (i.e., excluding *function_class*, *function_group*, *sector*, *expdaysfunctiongroup*, and *orderperfunctiongroup*), and 4) a dataset including only features related to time-dependent variables (i.e., including *transitionlustrum*, *order*, *expdays*, *month_of_startdate*, *orderperfunctiongroup*, and *expdaysfunctiongroup*). A comparison between the performance of GBM on these datasets and the full dataset is shown in Figure 3. Especially inclusion of the time variables caused a substantial improvement in both the Brier sore and C-index. Inclusion/exclusion of other types of attributes had a negligible effect.

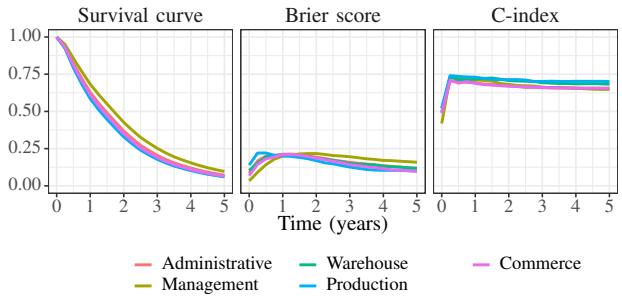

Figure 2. Results over the 5 most common job types.

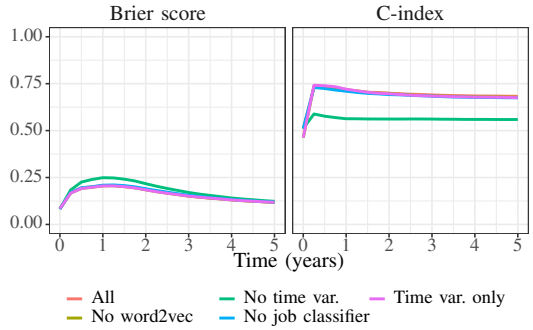

Figure 3. Scores on other datasets.

V. Conclusion and Future Work

From our comparison of machine-learned survival models, we find that especially tree-based ensembles, such as a random survival forest and Gradient Boosting Machines, work well to predict job tenure from unstructured résumés. They outperformed benchmark models in terms of the Brier score and C-index. These benchmarks included a Kaplan-Meier estimator and Cox regression, but also more complex models such as neural survival models and recurrent neural networks. Especially the importance of time-related variables in these models is interesting. Job - vacancy matching is often done using semantic overlap, e.g., comparing skill overlap between the vacancy and job. Our results suggest that including time-related variables in these matching algorithms may improve their performance.

Although tree-based ensembles outperformed benchmark models, still the prediction problem remains difficult. The difference with benchmark models are relatively small, and if one takes into account conditional survival times, and compares more similar job pairs, the error scores between the tree-based ensembles and benchmark models become more similar.

This limited performance may be explained in several ways. First, it should be acknowledged that predicting job

 ISBN: 978-1-61208-891-4

tenure from résumés is a difficult prediction problem. Previous work (using mostly Cox-PH models [10]) finds only weak correlations between predictors and tenure, (r^2 between 0.33 and 0.37) [3, p. 261]. Second, as illustrated in this study, résumés come with considerable fuzziness. Turnover itself may be indicated at different levels of granularity. Missing data is a considerable problem, as the résumé parser has to deal with a variety of formats. Also, job seekers may have different definitions of a job. E.g., one might define two positions at the same employer as one job, whereas another will consider these as two jobs. Naturally, such fuzziness complicates interpreting models derived from résumés.

Given these results, we are in particular interested in two directions for further work. Given the fuzziness of résumé data, an interesting direction would be to study whether survival analysis on résumés could benefit from models trained in different contexts, i.e., transfer learning. One could think of applying language models trained on larger corpora. But also training survival models on corporate turnover data, for which we expect to have more precise measurements, would be an interesting direction.

A second direction is with regards to the practical implications of predicting tenure from résumés for candidate recommendation. It would be interesting to consider how these models compare with semantic matching methods, using more application-directed error scores, such as NDCG. From a practical perspective, further research could also consider whether the model benefits from asking the user for additional data when uploading one's résumé.

VI. Acknowledgements

We would like to thank Ton Sluiter and USG People for their collaboration and guidance during the course of this work.

References

[1] F. Suvankulov, "Job search on the internet, e-recruitment, and labor market outcomes," Ph.D. dissertation, Pardee RAND Graduate School, 2010.

[2] M. N. Freire and L. N. de Castro, "e-recruitment recommender systems: a systematic review," *Knowledge and Information Systems*, pp. 1–20, 2020.

[3] W. F. Cascio, *Applied psychology in human resource management*. Prentice-Hall, 1998.

[4] K. Yu, G. Guan, and M. Zhou, "Resume information extraction with cascaded hybrid model," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005, pp. 499–506.

[5] S. K. Kopparapu, "Automatic extraction of usable information from unstructured resumes to aid search," in *2010 IEEE International Conference on Progress in Informatics and Computing*, vol. 1. IEEE, 2010, pp. 99–103.

[6] I. Paparrizos, B. B. Cambazoglu, and A. Gionis, "Machine learned job recommendation," in *Proceedings of the fifth ACM conference on Recommender systems*. ACM, 2011, pp. 325–328.

[7] M. Jiang, Y. Fang, H. Xie, J. Chong, and M. Meng, "User click prediction for personalized job recommendation," *World Wide Web*, vol. 22, no. 1, pp. 325–345, 2019.

[8] H. Li, Y. Ge, H. Zhu, H. Xiong, and H. Zhao, "Prospecting the career development of talents: A survival analysis perspective," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 917–925.

[9] L. Li, H. Jing, H. Tong, J. Yang, Q. He, and B.-C. Chen, "NEMO: Next career move prediction with contextual embedding," in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 505–513.

[10] P. W. Hom, T. W. Lee, J. D. Shaw, and J. P. Hausknecht, "One hundred years of employee turnover theory and research." *Journal of Applied Psychology*, vol. 102, no. 3, pp. 530–545, 2017.

[11] S. Strohmeier and F. Piazza, "Domain driven data mining in human resource management: A review of current research," *Expert Systems with Applications*, vol. 40, no. 7, pp. 2410–2420, 2013.

[12] J. Wang, Y. Zhang, C. Posse, and A. Bhasin, "Is it time for a career switch?" in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 1377–1388.

[13] Gus, *Website Gus*, 2017, https://www.gus.nl, retrieved: September 2021.

[14] Textkernel, *Website Textkernel*, 2017, https://www.textkernel.com/, retrieved: September 2021.

[15] M. Bouchet-Valat, "Package 'SnowballC'," R package version 0.6.0, https://cran.r-project.org/web/packages/SnowballC/index.html, 2019, retrieved: September 2021.

[16] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015, retrieved: September 2021.

[17] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation*, 2016, pp. 265–283.

[18] T. Tieleman and G. Hinton, "Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, 2012.

[19] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer, "Random survival forests," *The Annals of Applied Statistics*, pp. 841–860, 2008.

[20] H. Ishwaran and U. B. Kogalur, "Random Forests for Survival, Regression, and Classification (RF-SRC)," 2018, R package version 2.6.0, https://cran.r-project.org/web/packages/randomForestSRC/index.html, retrieved: September 2021.

[21] S. P. Jenkins, *Survival analysis*, 2005, unpublished, https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.176.7572&rep=rep1&type=pdf, retrieved: September 9, 2021.

[22] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 39, no. 5, pp. 1–13, 2011.

[23] D. Lin, "On the Breslow estimator," *Lifetime data analysis*, vol. 13, no. 4, pp. 471–480, 2007.

[24] B. Greenwell, B. Boehmke, and J. Cunningham, "Package 'gbm'," 2019, R package version 2.1.5, https://github.com/gbm-developers/gbm, retrieved: September 2021.

[25] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[26] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[27] M. F. Gensheimer and B. Narasimhan, "A scalable discrete-time survival model for neural networks," *PeerJ*, vol. 7:e6257, 2019.

[28] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] U. B. Mogensen, H. Ishwaran, and T. A. Gerds, "Evaluating random forests for survival analysis using prediction error curves," *Journal of Statistical Software*, vol. 50, no. 11, pp. 1–23, 2012.

[31] P. Wang, Y. Li, and C. K. Reddy, "Machine learning for survival analysis: A survey," *arXiv preprint arXiv:1708.04649*, 2017.

[32] M. Wolbers, P. Blanche, M. T. Koller, J. C. Witteman, and T. A. Gerds, "Concordance for prognostic models with competing risks," *Biostatistics*, vol. 15, no. 3, pp. 526–539, 2014.

[33] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for Cox's proportional hazards model via coordinate descent," *Journal of Statistical Software*, vol. 39, no. 5, pp. 1–13, 2011.

 ISBN: 978-1-61208-891-4