Explainable Career Path Predictions using Neural Models

Roan Schellingerhout[1,2,*], Volodymyr Medentsiy[2] and Maarten Marx[3]

[1]Department of Advanced Computing Sciences, Paul-Henri Spaaklaan 1, 6229 EN, Maastricht, The Netherlands
[2]Randstad Groep Nederland, Diemermere 25, 1112 TC, Diemen, The Netherlands
[3]IRLab Informatics Institute, Science Park 904, 1098 XH, Amsterdam, The Netherlands

Abstract

Career path prediction aims to determine a potential employee's next job, based on the jobs they have had until now. While good performance on this task has been achieved in recent years, the models making career predictions often function as black boxes. By integrating components of explainable artificial intelligence (XAI), this paper aims to make these predictions explainable and understandable. To study the effects of explainability on performance, three non-explainable baselines were compared to three similar, but explainable, alternatives. Furthermore, user testing was performed with recruiters in order to determine the sensibility of the explanations generated by the models. Results show that the explainable alternatives perform on-par with their non-explainable counterparts. In addition, the explainable models were determined to provide understandable and useful explanations by recruiters.

Keywords

Career path prediction, Explainable AI, Sequence classification, Neural networks, User testing

1. Introduction

With the rise of the modern gig economy, it has become more difficult for job seekers to find stable positions of employment [1]. In addition, due to the average education level of the workforce having increased considerably in recent years, potential employees are faced with more opportunities than ever before [2]. This has made it significantly more difficult for job seekers, and employment agencies alike, to find positions that fit their needs. To assist in this complex and challenging task, many employment agencies have started to make use of computer-aided HR matchmaking (e.g., machine learning) to find suitable positions for individuals, and capable employees for companies [3]. This task is called *career path prediction*, which aims to predict a person's next position of employment, given their career up until this point.

Previous research on automated career path prediction tends to share a common flaw: a lack of explainability [4, 5, 6, 7]. While deep learning tends to deliver good performance, these models often function as a black box. Although good results that are difficult to interpret are acceptable in many use cases, choosing a new career is such an impactful event in a person's life that it is unrealistic to expect users to blindly trust the models. This is why explainability is such a crucial requirement for career path prediction models. Through the use of explainable artificial intelligence (XAI) [8], individuals with little knowledge of deep learning (e.g. recruiters or job seekers themselves) are able to interpret to what extent, and in what ways, each variable contributed to the final outcome of the model. By being able to concretely determine *why* a given position is ideal for a person, the recommendation becomes considerably more transparent, understandable, and thus more trustworthy.

In this paper, career path prediction is performed on a dataset provided by Randstad NV. As the world's largest employment agency [9], Randstad has an enormous dataset containing the careers of hundreds of thousands of individuals. Considering the large number of same-job switches within this dataset (57% of all career steps in the dataset consist of people working the same job, just at a different company), only candidates that actually made a job *switch* were considered. This was done in order to prevent the machine learning models from always predicting a candidate's previous job, as that would defeat the purpose of using such a model.

This paper attempts to answer the following research question: *To what degree can career path predictions done by deep learning models be made explainable?* This is done by means of the following sub-questions:

- **RQ1:** How well do state-of-the-art deep learning models perform career path prediction on Randstad's dataset?
- **RQ2:** How do different ways of making model predictions explainable impact performance?
- **RQ3:** Which explainable model is the most useful for recommending jobs to candidates?

This paper is structured as follows: first, an overview

RecSys in HR'22: The 2nd Workshop on Recommender Systems for Human Resources, in conjunction with the 16th ACM Conference on Recommender Systems, September 18–23, 2022, Seattle, USA.
*Corresponding author.
roan.schellingerhout@maastrichtuniversity.nl (R. Schellingerhout)
0000-0002-7388-309X (R. Schellingerhout)

of the current state of the art in terms of model performance and explainability is given. Then, Randstad's dataset is described in detail. Afterwards, the methods used to answer the research questions are explained. Subsequently, the research questions are answered, after which their answers are discussed.

2. Related Work

2.1. Career path predictions

The goal of career path prediction is to determine what position of employment is a logical next step given a job seeker's career [5]. Considering the number of different career opportunities and factors which have an influence on the career steps (e.g., previous job experiences, educational background, interests of a job seeker), the career prediction problem is incredibly difficult to model by hand.

In recent years a lot of progress has been achieved within the field of career path prediction. The first notable paper to use machine learning for career path prediction, was that by Liu et al. [5]. In this paper, Liu et al. scraped individuals' social media profiles to generate a dataset, after which they predict when an employee would be ready to move to a higher-paying position within their current field (e.g. moving from junior software developer to senior software developer). Meng et al. [4] then extended this task by not just considering within-field switches, but general job mobility. Their custom LSTM, the *hierarchical career-path-aware neural network* (HCPNN), was thus tasked to predict individuals' next employer, regardless of their current field of employment. The HCPNN has shown impressive results, outperforming every model that forewent it.

Similarly, He et al. [7] attempted to predict individuals' next job based on features they extracted from their resume. Unlike Meng et al., they made use of a convolutional neural network (CNN) for the predictions. With this CNN they tried to implement a multi-purpose model that could not only predict talents' next job position, but also their salary and the size of the company they would be working at. Out of those three tasks, their CNN proved to perform the best on career path predictions.

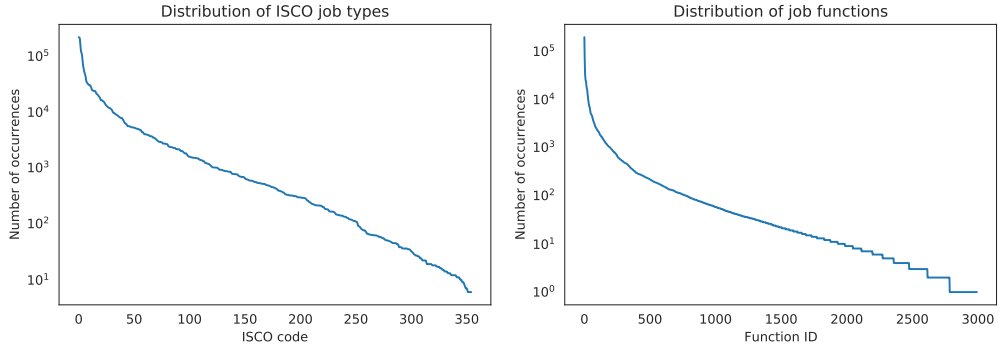
At their core, Meng et al.'s LSTM and He et al.'s CNN are simply feature extractors which feed their output into a dense layer. While both perform well on their own, it is common to combine these two architectures within the field of sequence classification [10, 11, 12]. Although such an architecture has not yet been used for career path predictions specifically, they have been shown to perform exceedingly well on other multivariate sequence classification problems [13, 14, 15]. Especially Livieris et al. [14] their CNN-LSTM has shown good results on another multivariate sequence classification task (gold price forecasting), outperforming every alternative architecture tested.

While the aforementioned models make up the current state of the art for career path predictions, they all share a common flaw: they function as black boxes. As a result, their outputs are hard to interpret for both recruiters and job seekers. Considering the impact a career change can have on an individual's life, this can make the models difficult to use in real-world scenarios.

2.2. Explainability in deep learning

Explainability and performance are often considered inverses of each other in the field of AI. A simple, easy to explain model is likely to perform mediocre at best, while a complex, difficult to explain model is more likely to perform well [8]. A common example of this inverse relationship can be seen in the difference between decision trees and random forests: random forests are based on decision trees, but with a higher degree of complexity, which strongly increases performance at the cost of explainability.

However, with the increasing interest in explainable AI, more and more solutions have been brought up that can make even the most complex deep learning models explainable to a degree [16]. Most commonly, this explainability takes the shape of visualizations of the networks' behaviour. Saliency maps and attention distributions are capable of visualizing the importance of different variables, usually through some type of colour scheme indicating higher or lower feature importance. Initially, Springenberg et al. [17] used guided backpropagation to visualize the features learned by convolutional layers. Extending past guided backpropagation, Selvaraju et al. [18] created Grad-CAM, which could not only visualize *general* learned features, but also determine which features were important for a *specific* predicted class. Since these post-hoc interpretability techniques merely look at the behaviour of the model, they do not alter their performance. However, it is often necessary to make alterations to the models' architecture to allow good explanations to be generated (e.g., they only work on convolutional layers, and preferably only on the *final* convolutional layer of a model) [18, 17]. As a result, such techniques either do not change performance at all, or decrease it slightly. In contrast, while both aforementioned methods were created for computer vision, Vaswani et al. [19] proposed 'attention mechanisms' for natural language processing. These attention mechanisms cause the models to predict the importance of each feature per time step (or the importance of a given time step in general) which can then be visualized. As a result, Vaswani et al. made it possible for different model architectures to become explainable, while simultaneously *improving* their performance.



(a) Distribution of ISCO job types

(b) Distribution of job functions

Figure 1: The distributions of ISCO job types and job functions (**N = 1664565**). Both use a logarithmic scale for the y-axis.

2.3. Explainability in sequence classification

Sequence classification brings an additional factor into the mix: the temporal dimension. Simply visualizing which features garner the most attention thus becomes insufficient in this scenario. While a given variable might be highly important to the network initially, it could become less relevant as time progresses. Thus, to make explainable sequence classifications, not only should there be an explanation of which variables contributed the most to the final prediction, but also at what moment their values were most decisive [20]. Nonetheless, saliency maps are still useful in this scenario, as a multivariate sequence can be treated as a 2-dimensional image of shape ($Number\ of\ features \times sequence\ length$). However, these saliency maps do not necessarily reach the level of finesse required to generate understandable explanations for sequence. As a result, saliency maps are often combined with attention mechanisms. By combining saliency maps with attention distributions, it is possible to improve the quality of the explanations [21].

3. Description of the Data

The data on which the models were trained, configured, and tested, was provided by Randstad NV (Randstad). Due to the nature of Randstad's operations, they have an exhaustive data lake consisting of temporal employee-related data.

3.1. Overview of the datasets

Randstad's dataset consists of over two million jobs relating to more than 500 thousand individuals. These jobs span over multiple decades, going back as far as the early twentieth century. Although Randstad is a multinational company, the used dataset only contains data pertaining to candidates living in the Netherlands. For each job, the dataset includes a number of relevant features, such as the company for which the person worked, the period within which they worked, ISCO[1] classifications of the job, and the specific function that was performed. While job function and ISCO type both represent job positions, the former is more granular as it takes over 3000 unique values, while the latter takes a mere 355.

Additionally, Randstad stores structured and unstructured profile-specific data, which can be used to describe the profile of a candidate. The structured data includes:

- education history, which includes education level, completion status, the start and (if applicable) end date;
- skills (e.g. 'programming: Python', 'operating a forklift', 'Microsoft Word', etc.);
- languages;
- driving licenses;
- location.

The unstructured data is represented by curriculum vitaes (CVs), which are user-generated documents.

3.2. Data imbalance

There is a huge imbalance in work experience and education levels of candidates present in the data. The imbalance in work experience occurs in job positions, which are represented by ISCO job types and job functions (see Figure 1a and 1b respectively), and the number of positions candidates have had (see Figure 2). We addressed the skew in the number of jobs a candidate had by limiting the job history to the 25 most recent jobs. The imbalance in education levels (see Figure 3) is less

[1] https://www.ilo.org/public/english/bureau/stat/isco/isco08/

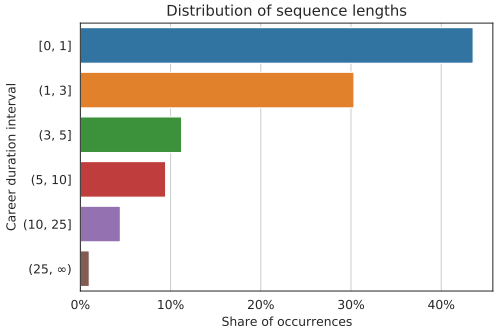


Figure 2: Distribution of different bins of total number of jobs held by each candidate (**N = 472647**). Individuals in the [0, 1] bin were removed from the dataset. For the full distribution, see Figure 6.

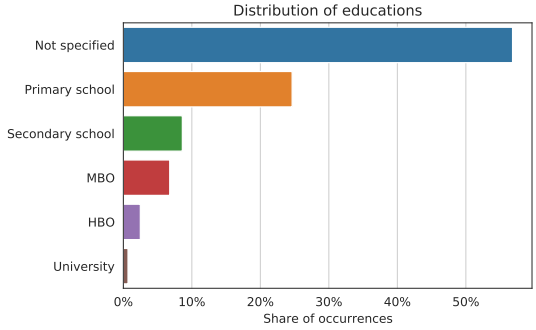


Figure 3: Distribution of highest education level obtained by candidates (**N = 1664565**). In the Dutch education system, MBO refers to intermediate vocational training, and HBO refers to university of applied sciences.

impactful, as the education level of candidates is merely a predictor, unlike the ISCO job types and job functions, both of which could be used as the actual labels to be predicted. To construct the final dataset we

- limited the job history of candidates to the 25 most recent jobs;
- dropped candidates with fewer than two jobs in the dataset, due to the inability to convert their careers to a sequence;
- balanced class labels distribution through weighted sampling during training.

This resulted in our final dataset consisting of the careers of 113724 candidates, each being limited to the 25 most recent jobs they had. For each job, the (normalized[2]) time spent working there, the ISCO function level of the job, the highest education enjoyed up until then, the company for which the candidate worked, the specific job function ID, the ISCO job type, and the most recent CV were stored. Additionally, the zip code, obtained certificates, mastered languages, skills, and driving licenses of candidates were stored as static variables, since they rarely changed in between jobs.

4. Methodology

In order to make career path predictions, candidates' profiles were turned into sequences which could be fed into different (deep learning) models. For each candidate we used the last 25 jobs along with profile-specific features as input for the models, after which the models would predict their next job in the form of its ISCO job type. Candidate profiles that consisted of fewer than 25 jobs were zero-padded to prevent mismatched sequence lengths. This section outlines how candidates' careers were converted into sequences, as well as how those sequences were fed into different models.

Lastly, an overview of the models used is given. The used models can be split into three separate categories: non-neural baselines, non-explainable neural models[3], and explainable neural models. 80% of the data was used as a training set, 10% of the data was used as a validation set, on which the optimal hyperparameters were determined, and the last 10% of the data was used as a test set to evaluate model performance on unseen data. We used weighted sampling during training to address the imbalance within the class labels distribution.

4.1. Data prepocessing

Due to the availability of temporal data, candidates' career paths were turned into sequences. For these sequences, each job held by a candidate was considered to be one time step. The order of the time steps was determined by the date at which the candidate started the position. As a result, every career was turned into a sequence, in which each time step was a candidate's current job, combined with their location and the skills, certificates, languages, and education they had achieved at the time of starting the position. To also include candidates' curriculum vitaes (CVs) at each time step, the most recent CV uploaded by a candidate at each time step was converted to numerical features using averaged Word2Vec [23] embeddings and combined with the other features.

Candidates' career paths were turned into sequences

[2]Normalization was done through Z-transformation in order to maintain a common scale for all features.

[3]The neural models were created in PyTorch and trained on an NVIDIA tesla K80 GPU [22].

$\mathbf{x}$ as follows:

$$\mathbf{x} = [\mathbf{x^{(1)}}, ..., \mathbf{x^{(T)}}], \tag{1}$$

$$with\ \mathbf{x^{(t)}} = [\mathbf{x^{(t)}_{job}}; \mathbf{x^{(t)}_{structured}}; \mathbf{x^{(t)}_{CV}}] \tag{2}$$

where the order of timesteps t is determined by the date at which the candidate started the job. Every element $\mathbf{x^{(t)}}$ of the sequence $\mathbf{x}$ consists of a feature vector $\mathbf{x^{(t)}_{job}}$, which represents candidate's current job at a timestep t, feature vector $\mathbf{x^{(t)}_{structured}}$, which represents their location, skills, certificates, languages, and education they had achieved at the time of starting the position, and feature vector $\mathbf{x^{(t)}_{CV}}$, which represents the most recent CV uploaded by a candidate at each time step (embedded using Word2Vec).

4.2. Baselines and Models

Considering the fact that careers do not necessarily follow a logical trend, they can be rather difficult to model properly. For example, a person might have a job for a while not because they want to, but because they are forced to do so in order to support themselves. A person going from a position as a software engineer to a store clerk does not constitute a logical progression, but can obviously occur in the real world whenever someone gets laid off and needs to work a temporary job while they search for new alternatives. This makes career path prediction a notoriously difficult problem for deep learning models [4]. To evaluate the added value of using such models, and to allow for better contextualization, baselines were set with three non-deep learning (but coincidentally highly explainable) models. The first one is a simple majority class baseline, which always predicts the most common job in the dataset. The second baseline is the majority *switch*, which always predicts the most common job following the current job of the candidate. The last simple baseline is more sophisticated: k-nearest neighbors based on the dynamic time warping distance between candidates that had the same previous job (KNN-DTW). This baseline uses dynamic time warping [24] to determine which candidates have had the most similar careers, and then uses k-nearest neighbors to make a prediction based on these similarities. For each candidate, all candidates that had the same previous job were compared in terms of DTW distance (i.e., the numerical distance between the sequences); the k-nearest neighbors algorithm was then used to find the closest neighbors for each candidate, based on whom a prediction would be made.

4.2.1. RQ1 - State of the art

To study the impact of explainability mechanisms on model performance, three state-of-the-art models, each with a unique architecture (Section 2.1), were trained and tested on Randstad's dataset. The performance of these models will function as a non-explainable baseline, with which the performance of the explainable alternatives can be compared. The following models were used:

LSTM : The LSTM -based model used in this paper is based on the HCPNN by Meng et al. [4]. While the original HCPNN combines candidate-specific data with company-specific data, its modular architecture allows for the removal of some of the model's components. As a result, the HCPNN was implemented using only candidate-specific features. This results in a model that takes embedded position features, feeds them into an LSTM, runs the LSTM's output through an attention layer, and combines that output with a candidate's embedded static features, after which a fully-connected layer makes a prediction.

CNN : The CNN -based model used in this paper is that of He et al. [7]. This architecture feeds the input data into a 2D convolutional layer, followed by a pooling layer. The output is then flattened and ran through a drop-out layer. Lastly, a fully-connected layer is used to do the final prediction.

CNN-LSTM : The CNN-LSTM -based model used in this paper is based on the model created by Livieris et al. [14]. It uses two sequential 2D convolutional layers, followed by a pooling layer. The pooled features then get fed into an LSTM, after which a fully-connected layer is responsible for the final predictions of the model.

To evaluate performance, accuracy @ k ($k \in \{1, 5, 10\}$) was used, which shows how often the correct answer was within the top k predictions given by the model [25]. Considering the fact that candidates could not be interested in a specific job type (e.g. no open vacancies, not interesting enough, it pays too little), it is expected of recruiters that they can provide multiple recommendations for the candidate, allowing them to choose and consider multiple options. As a result, the models provide multiple predictions, which can be evaluated using accuracy @ k.

4.2.2. RQ2 - Explainable models

Although the explainable models' architectures differ slightly from the aforementioned state-of-the-art models to allow for improved explainability, they are largely identical.

Explainable LSTM : The explainable LSTM -based model (eLSTM) used in this paper is based on the *spatiotemporal attention LSTM* (STA-LSTM) by Ding et al. [26]. This architecture starts off by

determining spatial attention; it runs each individual time step through a linear layer, after which the Hadamard product between the linear layer's output and the features per time step is taken to determine the importance of each feature at each time step. The output hereof is then fed into an LSTM, after which the temporal attention is calculated. This is done by flattening the output of the LSTM and running it through another linear layer. This calculates a normalized importance of each time step, based on that step's hidden values. The dot product between the linear layer's output and the LSTM's hidden output is then calculated, which is fed into a fully-connected layer to make the final predictions.

Explainable CNN : The explainable CNN -based model (eCNN) used in this paper is based on the *explainable convolutional neural network for multivariate time series classification* (XCM) by Fauvel et al. [27]. It makes use of two stages which run in parallel. The first stage (top) uses a 2D convolutional layer with kernel size ($window\,size \times 1$) that generates $F1$ feature maps. A (1×1) 2D convolutional layer is then used to summarize those $F1$ feature maps into a single feature map. The other stage (bottom), running independently, uses a 1D convolutional layer with kernel size ($window\,size \times N\,features$) and also generates $F1$ feature maps, which are summarized by a (1×1) 1D convolutional layer. The two feature maps generated by the two stages are then concatenated in the feature-dimension, after which a 1D convolutional layer with kernel size ($windowsize \times (Nfeatures+1)$) generates $F2$ feature maps. These feature maps are then ran through a pooling layer, which is also responsible for the predictions. $F1$, $F2$, and *window size* are three separate hyperparameters for this model.

Explainable CNN-LSTM : The explainable CNN-LSTM -based model (eCNN-LSTM) used in this paper is based on that of Schockaert et al. [21]. This model runs the input data through a 2D convolutional layer with kernel size ($sequence\,length \times 1$), whose output gets concatenated to the original sequential data. This combined output gets fed into an LSTM. All but the last hidden state of the LSTM get passed through a temporal attention mechanism. This temporal attention mechanism runs each hidden state through a fully-connected layer which attributes it a given amount of attention. These attention values are then normalized, after which the dot product of the attention vector and the hidden states is calculated to create a *context vector*. This context vector is then concatenated to the last hidden state of the LSTM, and fed into fully-connected layer, which makes the final prediction.

4.2.3. RQ3 - Real-world utility

To measure the adequacy of the explanations generated by the models, user testing was performed. Potential users of the models (e.g. Randstad's recruiters), were tasked to determine which variables were most relevant for a prediction made by the system. Six recruiters were split into three groups based on their recruiting expertise (finance, customer support, health care), and shown three separate predictions within that industry (one per model). For each predictions, they were tasked to distribute 100 'relevance points' over all of the features used by the models (previous jobs, education, skills, etc.), after which their distribution was compared to that of the models. In order to determine the sensibility of the models' explanations, the Pearson's correlation, root mean squared error (RMSE), and mean absolute error (MAE) of each models' distributions compared to the recruiters' distributions were calculated. Furthermore, the recruiters were presented with the explanations generated by each model, and tasked to judge each part of the explanations (spatial/feature attention, temporal attention, and spatiotemporal attention), as well as the general usefulness of the explanations for finding a suitable position for a candidate. By averaging the scores given by the recruiters, the real-world utility of each explanation was determined.

5. Results

5.1. RQ1 - State of the art

To better convey the performance gained by using deep learning models, the score of each model will be directly compared to that of the best-performing baseline. Of the three simple baselines, the majority switch baseline performed the best, reaching 19.1% accuracy @ 1, 46.6% accuracy @ 5, and 61.3% accuracy @ 10. KNN-DTW performed worse initially, but converged to the majority switch baseline as the number of neighbors (K) approached infinity. With low values of K, e.g. 5, it failed to break even 10% accuracy @ 1. However, using a higher value for K, e.g. 100, greatly improved this score, reaching 18.1% accuracy @ 1, 46.4% accuracy @ 5, and 58.1% accuracy @ 10, showing a sub-linear performance gain as K increased. The majority class baseline performed significantly worse, only reaching 10.5% accuracy @ 1, 36.8% accuracy @ 5, and 49.1% accuracy @ 10. As a result, the performance of the deep learning models was compared against the scores achieved by the majority switch baseline.

Model	Accuracy @ 1 ⇑	Accuracy @ 5 ⇑	Accuracy @ 10 ⇑
	Baseline		
Majority switch	19.1% ± 0.7%	46.6% ± 0.9%	61.3% ± 0.9%
	Non-explainable models		
LSTM	21.9% ± 0.8%	49.3% ± 0.9%	62.9% ± 0.9%
CNN	20.8% ± 0.7%	50.8% ± 0.9%	63.7% ± 0.9%
CNN-LSTM	**26.4% ± 0.6%**	**56.5% ± 0.7%**	**68.6% ± 0.6%**
	Explainable models		
eLSTM	22.2% ± 0.8%	47.6% ± 0.9%	60.8% ± 0.9%
eCNN	20.1% ± 0.7%	47.7% ± 0.9%	61.5% ± 0.9%
eCNN-LSTM	**26.0% ± 0.8%**	**55.7% ± 0.9%**	**67.5% ± 0.9%**

Table 1
Test set performance of each model at different values of k (**N = 11372**). Different values of k indicate how often the correct answer was within the top k predictions given by the model. Green text indicates scores *higher* than the majority switch baseline, while red text indicates scores *lower* than the majority switch baseline.

While similar architectures were used for the explainable and non-explainable models, different hyperparameter configurations led to different performance for each architecture. The results shown in Table 1 only indicate the performance given by the best hyperparameter configuration found for each model. For a full overview of hyperparameter configurations and their related performance see Appendix B.

5.2. RQ2 - Explainable models

Out of all the models, the CNN-LSTMs performed the best. Unlike what was hypothesized, the explainable models were not inferior to their non-explainable counterparts (Table 1). In fact, the eLSTM provides a higher accuracy than the non-explainable LSTM by a slight margin, although this difference falls within the confidence intervals of the scores, and is therefore not significant ($p > .05$). The explainable CNN took a slight (but statistically significant) hit in performance in exchange for the increase in explainability, especially suffering at higher values of k.

5.3. RQ3 - Real-world utility

Each explainable model is able to generate three separate explanations for a prediction: (i) the weight of each feature, (ii) the weight of each time step, and (iii) a time step/feature interaction map (spatiotemporal attention). The way in which these explanations are generated differs per model, but the final visualizations are the same, regardless of the method used to generate them (Figure 10, 11, and 12 in Appendix E).

In order to verify the integrity of these explanations, user research was done with Randstad's recruiters. After providing the recruiters with the predictions made by the model, they were asked to estimate which variables were most important. The averaged estimates made by the recruiters and models can be seen in Figure 4 (for the comparison per model see Appendix C). The results indicate that the models' explanations were positively correlated with those made by the recruiters (Table 2. For the eCNN-LSTM, this correlation was moderate, while for the eCNN and eLSTM, it was quite weak. In general, the models considered more 'job-specific' features such as the previous functions, companies, ISCO job types, and ISCO job levels to be highly important, while the recruiters leaned more towards 'general' features such as education and skills.

To measure the sensibility of each model's explanations, three metrics were calculated for each of them:

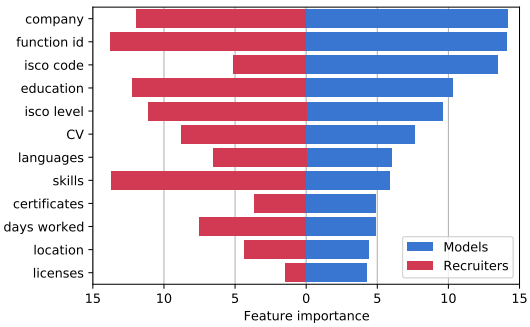


Figure 4: Average distribution of feature importance of the three explainable models compared to that of Randstad's recruiters (**N = 18**).

	Pearson's r ⇑	RMSE ⇓	MAE ⇓
eLSTM	0.142	4.661	4.094
eCNN-LSTM	0.436	6.014	4.847
eCNN	0.152	5.594	4.518

Table 2
The Pearson correlation, RMSE, and MAE of each model compared to the scores given by the recruiters ($\mathbf{N = 6}$). For each feature, both the models and the recruiters gave a score; the scores are calculated based on those two scores.

RMSE, MAE, and Pearson correlation. This was done by calculating the difference between the average score that recruiters gave to each feature and the attention put towards that feature by the models (RMSE and MAE), as well as the correlation between the models' values and the recruiters' values (Pearson correlation). The results can be seen in Table 2.

Additionally, the recruiters were asked how sensible they found the models' explanations, as well as how useful they considered the models (including their explanations) for helping candidates find a new job. The averaged scores for each model is shown in Table 3.

In general, the recruiters showed a preference for the feature explanations, and to a lesser extent the spatiotemporal explanations. The temporal explanations were considered the least sensible, failing to reach a sufficient grade (i.e., above a 5.5/10 on average). While the eCNN was judged to deliver the worst explanations, receiving barely a 5/10 on average, the eCNN-LSTM's and eLSTM's explanations were considered sufficient by the recruiters. Out of these two, the eCNN-LSTM was determined to provide the best explanations, scoring the highest average rating in each category. Regardless of the insufficient grades reached by some explanations/models, all three models were considered generally useful for recommending a job to a candidate.

6. Discussion and conclusion

6.1. Interpretation of the results

6.1.1. State of the art performance

Although career path prediction is a notoriously difficult problem in deep learning, the state-of-the-art models used on Randstad's dataset ended up performing commendably. All three models ended up achieving significantly ($p < .05$) higher scores than the majority switch baseline, which already performed well. However, this improvement is relatively small for the CNN and LSTM. This marginal increase over the baseline is largely in line with the results found in previous research. Meng et al. [4] found that the HCPNN outperformed non-neural baselines by about 20% on their dataset; improving from 6.0% to 7.3% accuracy @ 1. Although this is a larger improvement than that of the HCPNN compared to the majority switch baseline presented in this paper (14.6% increase in accuracy @ 1), this result can still be considered a confirmation of Meng et al. their findings. The smaller relative improvement could in part be caused by the fact that Randstad's dataset includes data that has been manually input by candidates themselves. This data, as opposed to that input by Randstad's recruiters, has not been verified, and could therefore include errors, a substantial amount of missing values, etc. While these data points could have been removed from the dataset to improve performance, a conscious decision was made not to. Removing all data entered by candidates themselves would get rid of more than half the dataset, in exchange for a relatively minor improvement in performance (in the neighborhood of 5-10%, absolute). Additionally, in real-world use, providing candidates with the ability to enter their own career into Randstad's system and instantly being able to receive job recommendations is very valuable.

As opposed to the CNN and LSTM, the CNN-LSTM showed a major improvement over the baseline. This is in accordance with the results found by Livieris et al. [14], who showed that their CNN-LSTM significantly outperformed a bare LSTM baseline. Considering the fact that both the convolutional layers and LSTM layers are used as feature extractors, this result is expected. By combining the two layer types, the model is able to learn more abstract representations of the data, allowing it to generalize better [28, 29, 30].

6.1.2. Explainability's impact on performance

Though it was initially expected that the inclusion of explainability mechanisms would impact model performance to a degree [8], the experiments have shown that this is not the case. While for Grad-CAM (CNN) this result might seem obvious, considering this technique does not alter the model, but merely looks at the model's gradients, this is still surprising. Despite the fact that the technique itself is not intrusive, the model's architecture still needed to be altered in order to create sensible explanations (e.g. the eCNN's parallel design), as shown by Fauvel et al. [27]. Regardless of this architectural change, however, the explainable model still performed on-par with its counterpart. Similarly, the explainable CNN-LSTM, which uses not only guided backpropagation, but also an attention mechanism, showed roughly equal performance to the non-explainable CNN-LSTM. For the LSTM, the addition of explainability even improved the model's performance (in terms of accuracy @ 1), although this improvement was not statistically significant. Thus, the experiments show that explainability mechanisms can be used in deep learning models

	Feature explanation	Temporal explanation	Spatiotemporal explanation	General usability
eLSTM	6.4 (SD=2.30)	5.4 (SD=2.30)	5.4 (SD=1.14)	6.0 (SD=0.71)
eCNN	5.2 (SD=1.79)	4.6 (SD=2.70)	5.4 (SD=2.07)	6.4 (SD=1.14)
eCNN-LSTM	6.6 (SD=2.51)	5.4 (SD=1.67)	6.4 (SD=2.41)	6.8 (SD=1.10)

Table 3
The average rating of each type of explanation for each model, as well as their general usability score, as determined by Randstad's recruiters ($\mathbf{N = 5}$). 1-10 scale.

for career path prediction without hindering the models' predictive powers. For the most part, this is in line with the results of previous research on the topic [17, 18]. However, the fact that the attention mechanisms used in the eCNN-LSTM and eLSTM did not improve model accuracy in a statistically significant manner is in stride with the results found by Schockaert et al. [21] and Ding et al. [26]. This is likely caused by the differences between their datasets and the one provided by Randstad. For example, the majority of candidates in Randstad's dataset only had one job on record. In such a scenario, temporal attention adds no value, as all attention will be directed towards that single time step.

6.1.3. Real-word utility

User testing showed that recruiters consider the explainable models usable in a real-world scenario. Although they were quite critical, giving mostly sufficient (but not outstanding) grades, they determined that each model type would at least be helpful to a degree in finding a job for a candidate. The individual explanation types tended to score lower than the models as a whole, indicating that the current implementation of the models' explanations (i.e. the visualizations in Appendix E) might require some tuning or extra clarification in order to be used efficiently by recruiters. Regardless, the recruiters did indicate that they considered the current implementation useful as is. Considering the environment for user testing is quite bare-bones (Appendix D), this is a positive indication for the actual usability of the models' explanations. Thus, to to allow further capitalization of the explanations, a more user-friendly interface (e.g. interactive explanations, clear textual descriptions of the data) could be used. In doing so, the models might also become usable by candidates themselves. Considering the inference time of the models (less than a second), candidates could enter their careers into Randstad's system, and instantly be provided a list of job recommendations, accompanied by explanations. However, more research will need to be done to determine if this is preferable for candidates over having recruiters interpret the models' predictions.

6.2. Potential biases

While the models performed commendably, and the explanations were determined to be satisfactory, it is important to consider the impact of biases in the training data on the predictions. Although protected features, such as gender, race, and age were removed from the dataset, correlation between such features and input features may still have caused discrimination [31]. For example, while age was not explicitly present in the data, the models could still roughly determine a candidate's age based on their total number of days worked across all jobs (a person with a few hundred total days worked is likely to be in their twenties, while someone with over ten thousand days worked is probably nearing retirement). The models' ability to 'retrieve' such protected features may have negatively affected the recommendations for specific candidates. Future research could look into the extent to which this occurs, as well as methods to alleviate this effect.

6.3. Limitations and expansion

Due to the lack of a publicly available dataset, determining state-of-the-art performance is complicated for career path prediction. Even within Randstad's own dataset, performance could be increased by simply filtering out data entered by candidates. To advance the field of career path prediction, future research should focus on creating a general dataset that can be used to directly compare model performance within the field (in the same vein as ImageNet for image classification[4] and TREC for text retrieval[5]). This benchmarking dataset should consist of relatively clean, GDPR compliant, exhaustive career data of a large variety of candidates. Using this dataset, future research will be able to better gauge the performance of different architectures used for career path prediction (e.g. LSTMs, CNNs, temporal graphs) and draw direct comparisons between models. Thus, having a clear and definite state of the art will most certainly advance the field as a whole.

Another limitation posed in this paper, is the lack of hardware resources. The NVIDIA Tesla K80 used to train

[4] https://www.image-net.org/
[5] https://trec.nist.gov/data.html

the models fell short when training the CNN-based models. Because of the low CUDA core count of 2496, and the limited 12 gigabytes of VRAM, the convolutional models had to be limited in terms of kernel size, output channels, embedding sizes, epochs, and batch sizes to decrease VRAM usage and keep training time reasonable. Consequently, not all possible hyperparameter configurations could be tested, possibly leaving better model configurations unexplored.

Furthermore, the small sample size used for user testing is an important limitation to acknowledge. Because the participating recruiters were on payroll, it was difficult to get their managers' approval, as well as to schedule a moment to perform the tests. Subsequently, the results gathered by the user testing are subject to high variance and are therefore difficult to use as conclusive evidence. Increasing the sample size by also performing user testing on candidates themselves would have helped solve this issue and might have provided additional insights. Also, improving the clarity of the UI used for user testing and the models' explanations could have led to lower variance, making the results more conclusive.

Additionally, while only including career switches in the training data strongly improved the models' usability, it also hinders individuals who are looking for new work within their current field from receiving recommendations. To account for such candidates, future work could expand upon the current pipeline by including a recommendation on whether a candidate should stay within their current field, or pursue a position with a different function. For individuals who get recommended to stay within their profession, the models could, for example, be altered to recommend a next employer within the field.

6.4. Conclusion

In the span of this paper, it was shown that career path predictions made by deep learning models can be made explainable to a high degree. While different types of explanations made by the models can differ in terms of how understandable they are to humans, all of them turned out to be useful for recruiters nonetheless. Due to the fact that these explainability mechanisms do not lead to a decrease in performance, they form a good addition to existing career path prediction models. This goes especially for CNN-LSTMs, as those perform the best as explainable and non-explainable models, while also providing the best explanations according to recruiters.

References

[1] P. Parigi, X. Ma, The gig economy, XRDS: Crossroads, The ACM Magazine for Students 23 (2016) 38–41.

[2] M. Hanson, F. Checked, Educational attainment statistics [2022]: Levels by demographic, 2021. URL: https://educationdata.org/education-attainment-statistics.

[3] T. Zimmermann, L. Kotschenreuther, K. Schmidt, Data-driven hr-r\'esum\'e analysis based on natural language processing and machine learning, arXiv preprint arXiv:1606.05611 (2016).

[4] Q. Meng, H. Zhu, K. Xiao, L. Zhang, H. Xiong, A hierarchical career-path-aware neural network for job mobility prediction, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 14–24.

[5] Y. Liu, L. Zhang, L. Nie, Y. Yan, D. Rosenblum, Fortune teller: predicting your career path, in: Proceedings of the AAAI conference on artificial intelligence, volume 30, 2016, pp. 1–16.

[6] M. Kokkodis, P. G. Ipeirotis, Demand-aware career path recommendations: A reinforcement learning approach, Management Science 67 (2021) 4362–4383.

[7] M. He, D. Shen, Y. Zhu, R. He, T. Wang, Z. Zhang, Career trajectory prediction based on cnn, in: 2019 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), IEEE, 2019, pp. 22–26.

[8] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, G.-Z. Yang, Xai—explainable artificial intelligence, Science Robotics 4 (2019) eaay7120.

[9] S. R. Department, Staffing industry: Leading companies worldwide, 2022. URL: https://www.statista.com/statistics/257876/staffing-companies-worldwide-by-revenue/.

[10] A. Vidal, W. Kristjanpoller, Gold volatility prediction using a cnn-lstm approach, Expert Systems with Applications 157 (2020) 113481.

[11] W. Lu, J. Li, Y. Li, A. Sun, J. Wang, A cnn-lstm-based model to forecast stock prices, Complexity 2020 (2020).

[12] R. Rick, L. Berton, Energy forecasting model based on cnn-lstm-ae for many time series with unequal lengths, Engineering Applications of Artificial Intelligence 113 (2022) 104998.

[13] T.-Y. Kim, S.-B. Cho, Predicting residential energy consumption using cnn-lstm neural networks, Energy 182 (2019) 72–81.

[14] I. E. Livieris, E. Pintelas, P. Pintelas, A cnn–lstm model for gold price time-series forecasting, Neural computing and applications 32 (2020) 17351–17360.

[15] H. Xie, L. Zhang, C. P. Lim, Evolving cnn-lstm models for time series prediction using enhanced grey wolf optimizer, IEEE Access 8 (2020) 161519–161541.

[16] J. Choo, S. Liu, Visual analytics for explainable deep learning, IEEE computer graphics and applications

38 (2018) 84–92.

[17] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: The all convolutional net, arXiv preprint arXiv:1412.6806 (2014).

[18] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 618–626.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

[20] T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, N. Díaz-Rodríguez, Explainable artificial intelligence (xai) on timeseries data: A survey, arXiv preprint arXiv:2104.00950 (2021).

[21] C. Schockaert, R. Leperlier, A. Moawad, Attention mechanism for multivariate time series recurrent model interpretability applied to the ironmaking industry, arXiv preprint arXiv:2007.12617 (2020).

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Advances in neural information processing systems 32 (2019).

[23] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).

[24] D. J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series., in: KDD workshop, volume 10, Seattle, WA, USA:, 1994, pp. 359–370.

[25] D. M. Powers, Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation, arXiv preprint arXiv:2010.16061 (2020).

[26] Y. Ding, Y. Zhu, J. Feng, P. Zhang, Z. Cheng, Interpretable spatio-temporal attention lstm model for flood forecasting, Neurocomputing 403 (2020) 348–359.

[27] K. Fauvel, T. Lin, V. Masson, É. Fromont, A. Termier, Xcm: An explainable convolutional neural network for multivariate time series classification, Mathematics 9 (2021) 3137.

[28] R. Eldan, O. Shamir, The power of depth for feedforward neural networks, in: Conference on learning theory, PMLR, 2016, pp. 907–940.

[29] A. Subasi, Chapter 5 - other classification examples, in: A. Subasi (Ed.), Practical Machine Learning for Data Analysis Using Python, Academic Press, 2020, pp. 323–390. URL: https://www.sciencedirect.com/science/article/pii/B9780128213797000059. doi:https://doi.org/10.1016/B978-0-12-821379-7.00005-9.

[30] Y. Chen, K. Zhong, J. Zhang, Q. Sun, X. Zhao, Lstm networks for mobile human activity recognition, in: 2016 International conference on artificial intelligence: technologies and applications, Atlantis Press, 2016, pp. 50–53.

[31] F. Träuble, E. Creager, N. Kilbertus, F. Locatello, A. Dittadi, A. Goyal, B. Schölkopf, S. Bauer, On disentangled representations learned from correlated data, in: International Conference on Machine Learning, PMLR, 2021, pp. 10401–10412.

[32] T. Contributors, Torch.sparse¶, 2022. URL: https://pytorch.org/docs/stable/sparse.html.

[33] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

7. Appendix

All code used in the experiments can be found on https://github.com/Roan-Schellingerhout/MSc_thesis.

A. Encoding and indexing

With over 100 thousand careers, each spanning 25 time steps, and over 1000 features per time step (embedding values for skills, certificates, previous jobs, previous companies, addresses, and spoken languages, as well as 300 w2v dimensions per CV), feeding the data into deep learning models as is, turned out to be infeasible. Making use of sparse vectors to lower memory usage also was impossible, due to the incompatibility between CUDA and sparse vectors/matrices [32]. However, considering the large amount of duplicate data (a candidate's skills/certificates/CVs do not change at every time step, and can therefore often be repeated), use was made of indices in order to lower memory usage, at the cost of a slight time complexity increase. For each candidate, a location within each index was created that contained their unique attributes, and the time steps from which those attributes became the most recent ones. By then retrieving the relevant attributes for each candidate in a batch during training, the required memory usage was lowered drastically.

B. Hyperparameters

All hyperparameter tuning results can be found on GitHub. For each configuration, the models were ran for 3 epochs. Based on the results after those 3 epochs, the best performing configuration was ran for 20 epochs to find the optimal number of epochs. Not every intended hyperparameter configuration could be tested due to hardware/time constraints. For example, the CNN-based models needed to be limited to small kernels and output channels to prevent running out of VRAM. Additionally, the eCNN was only trained for a total of 3 epochs, due to time constraints (as each epoch took nearly 8 hours).

All models were optimized using the Adam optimizer [33] (learning rate = $1 * 10^{-3}$) with cross-entropy loss. The hyperparameters used for the results of the non-explainable models in Table 1 were the following:

LSTM : The HCPNN used a batch size of 512 and reached optimal performance after 18 epochs. It used a single LSTM layer with hidden size 1000.

CNN : The CNN used a batch size of 128 and reached optimal performance after 11 epochs. The 2D convolutional layer consisted of a (5×5) kernel, with (1×1) padding and stride, and generated 64 feature maps. The 3D max-pooling used a $(64 \times 1 \times 1)$ kernel with $(1 \times 1 \times 1)$ stride.

CNN-LSTM : The CNN-LSTM used a batch size of 128 and reached optimal performance after 20 epochs. The first 2D convolutional layer used a (1×1) kernel, with a (1×1) stride and half padding, and generated 32 feature maps. The second 2D convolutional layer made use of the same kernel size, stride, and padding, but generated 64 feature maps. The following 3D average-pooling layer used a $(64 \times 1 \times 1)$ kernel and a $(1 \times 1 \times 1)$ stride. Lastly, the model used a single LSTM layer with hidden size 1000.

The optimal hyperparameters found for the explainable models are as follows:

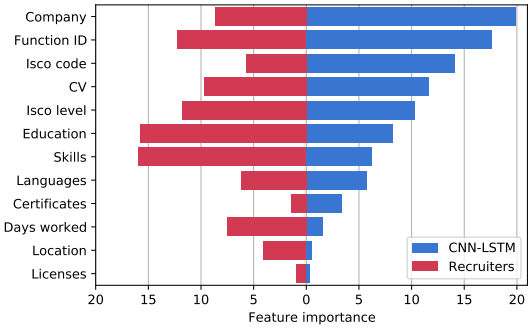
eLSTM : The explainable LSTM used a batch size of 128 and reached optimal performance after 5 epochs. It used a single LSTM layer with hidden size 1000.

eCNN : The explainable CNN used a batch size of 128 and reached optimal performance after 2 epochs. The top part used a 2D convolutional layer with a (5×1) kernel (thus, $window\ size = 5$), a (1×1) stride, half padding, and generated 8 feature maps (thus, $F1 = 8$). For the bottom part, the 1D convolutional layer used a $(5 \times N\ features)$ kernel, a (1×1) stride, half padding, and also generated 8 feature maps. The final 1D convolutional layer used a kernel size of $(5 \times (N\ features + 1))$, a (1×1) stride, half padding, and generated 32 feature maps (thus, $F2 = 32$). These 32 feature maps were then ran through an 3D average-pooling layer with kernel size $(32 \times 1 \times 1)$ and a $(1 \times 1 \times 1)$ stride.

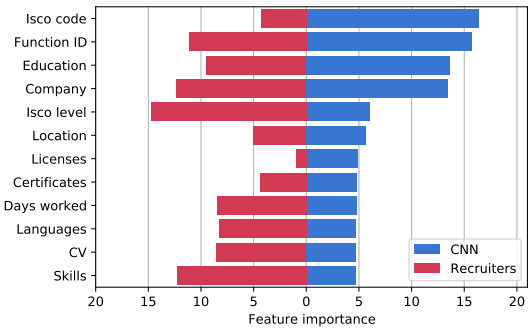
eCNN-LSTM : The explainable CNN-LSTM used a batch size of 2048 and reached optimal performance after 15 epochs. Its 2D convolutional layer used a kernel of size $(sequence\ length \times 1)$ and half padding, and was followed by a single LSTM with hidden size 1000.

C. Recruiter vs. model distributions

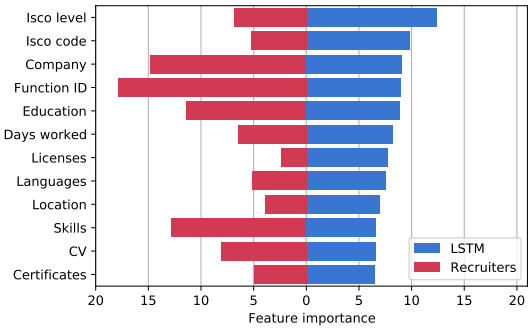
The distributions of feature importance on which Table 2 is based can be seen in Figures 5a, 5b, and 5c. Each model distribution is based on the average feature importance determined by the models across the three categories (finance, health care, and customer support). For the recruiter distribution, the average is taken over the three industries, as well as all recruiters within those industries (as a result, $N = 6$ for all recruiter distributions).



(a) Distribution of feature importance of the CNN-LSTM.



(b) Distribution of feature importance of the CNN.



(c) Distribution of feature importance of the LSTM.

Figure 5: Distribution of feature importance of the different models compared to that of Randstad's recruiters. $\mathbf{N = 6}$, averaged over three categories.

D. User testing

User testing was conducted using a web environment accessible by the recruiters. The web app was hosted using Amazon ec2 in combination with Docker, and built using Flask, JQuery, Jinja, and AJAX. The recruiters were tasked to enter their e-mail address (to allow follow-up questions if needed) and select their expertise (finance, health care, customer support). Afterwards, they were redirected to the first example relevant to their expertise. On this page, the recruiters were shown the data related to the candidate in question (Figure 7), the prediction made by the model, as well as one slider for each feature which they could adjust (Figure 8). In total, 6 recruiters participated in the experiment (although one of them only submitted their slider ratings, and not their model judgements).

By adjusting the sliders for each feature, they could distribute the 'relevance points' and thereby indicate which features they considered most important for the given prediction. After submitting their relevance distribution, the recruiters were redirected to a page that again showed the data of the user, the prediction made by the model, this time accompanied by the model's explanation, and the four questions regarding the sensibility of the explanations and the usability of the model (Figure 9). Once the recruiters gave a rating to each explanation type, the recruiters would be shown the second example and repeat the steps. Upon having completed the third example, they were informed they were done, after which their results were retrieved from the ec2 server and processed using Python.

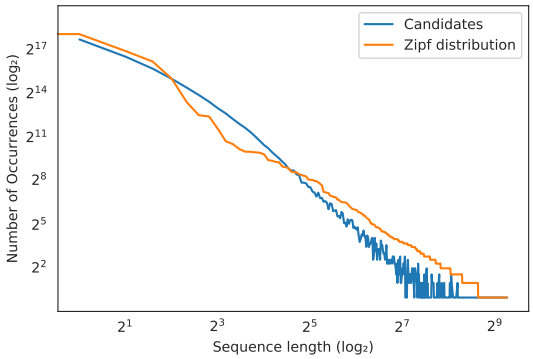


Figure 6: The full distribution of the job sequence lengths (number of jobs held per candidate). The longest single sequence consisted of 613 jobs. Both axis are in log_2. Distributed according to $Zipf(\alpha = 1.5, n = 613)$.

E. Explanation examples

The explanations provided by the three different models for the same candidate can be found in Figures 10, 11, and 12. The correct label for this candidate was *Survey and market research interviewer.*

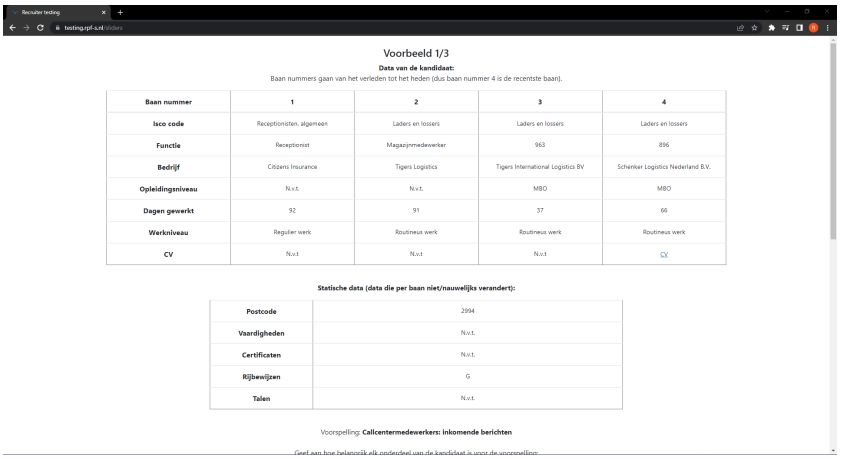


Figure 7: The interface for observing the users' data. Time series data and static data are shown separately in the two different tables to improve clarity. Below the tables, the label predicted by the model is displayed in bold.

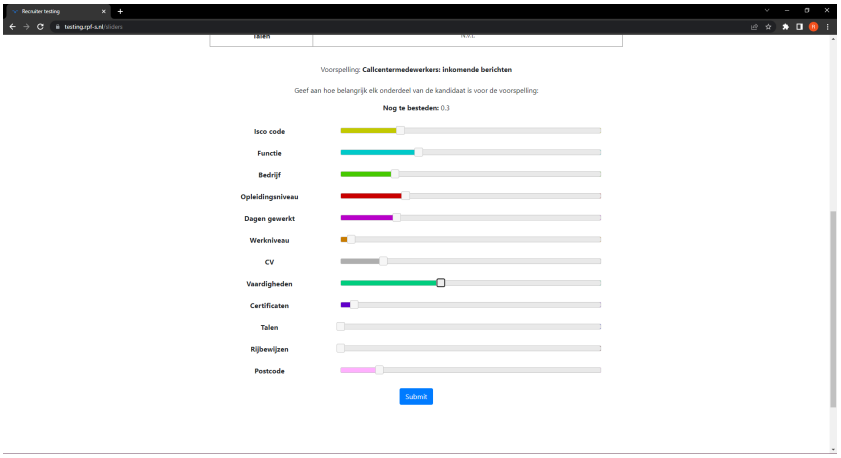


Figure 8: The sliders used to determine feature importance. At the top, the total amount of 'relevance points' left to spent is displayed in bold. Once this number reaches 0, the sliders can no longer be increased, unless another is decreased.

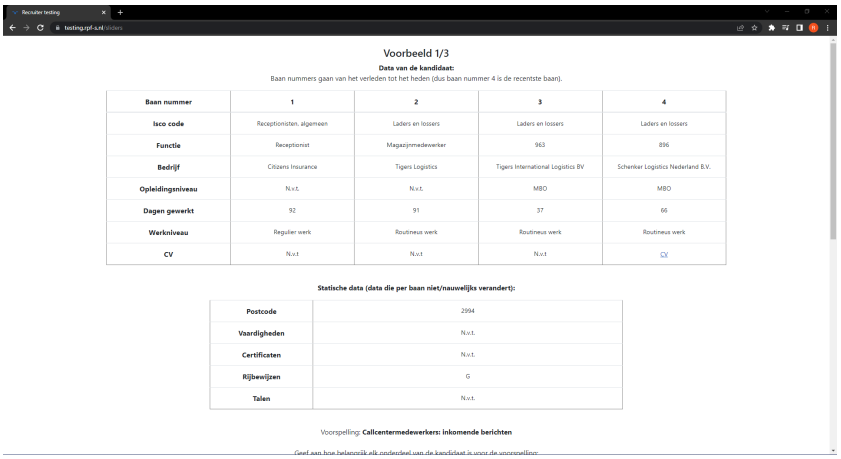


Figure 9: The interface for judging the models' explanations. By scrolling up, the prediction made by the model, as well as the users' data can be observed (as in Figure 7). A brief explanation on how to interpret the explanations is also provided.

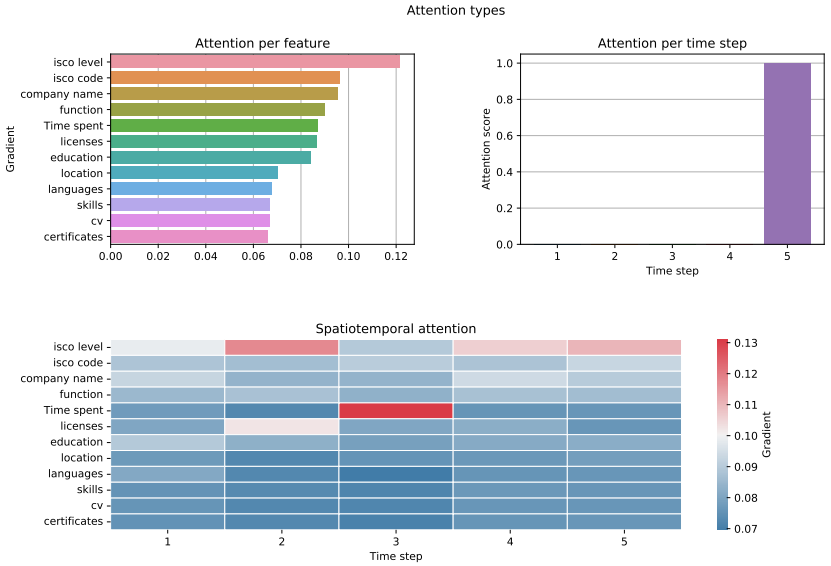


Figure 10: Explanations provided by the explainable LSTM. Top left: attention per feature. Top right: attention per time step. Bottom: Feature/time step interaction (spatiotemporal attention).

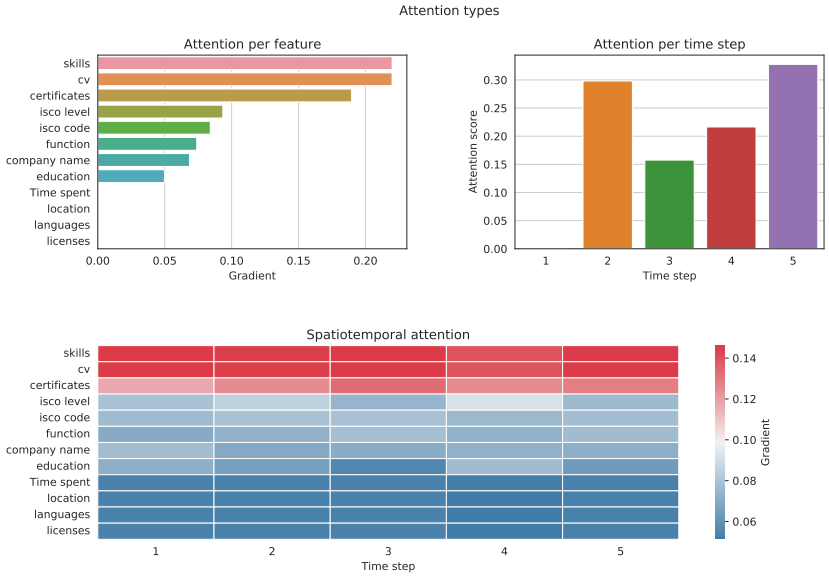


Figure 11: Explanations provided by the explainable CNN. Top left: gradient weight per feature. Top right: gradient weight per time step. Bottom: Feature/time step interaction (grad-CAM)

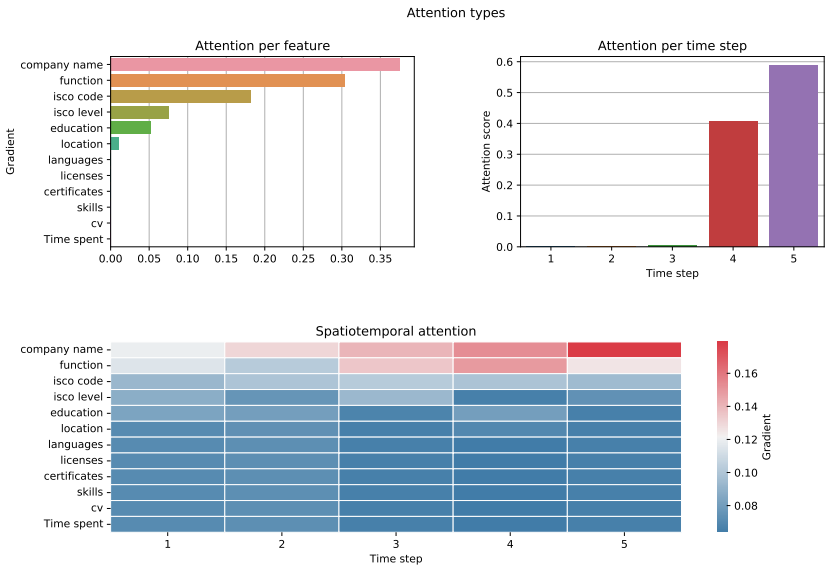


Figure 12: Explanations provided by the explainable CNN-LSTM. Top left: gradient weight per feature. Top right: attention per time step. Bottom: Feature/time step interaction (guided backpropagation)