University College London

Department of Computer Science

Machine Learning MSc Project - COMPGI99

Career Path Advice using Machine Learning

Author: Edward Pease
Supervisor: John Shawe-Taylor

September, 2017

Acknowledgements

I would like to thank John Shawe-Taylor for his guidance throughout the project and the data science team at Adzuna (James, Renaud and Jamie) for all of their advice and support.

Abstract

Every job site aims to present their users with a selection of highly relevant jobs. Currently on most job sites, candidates input a desired job title, and perhaps various other parameters, into a search box to return some relevant jobs. The main problem with this approach is that the candidate might not know what they want to do next or might want to be aware of other available options. An obvious solution is to personalise any recommendations using the candidate's CV or LinkedIn profile. The current state-of-the-art paper in this area achieves strong results for two separate industries, finance and technology. Most job sites, however, cover many different industries. This work uses a comparable setup, a contextual LSTM, to train a single model which achieves similar results to the state-of-the-art but for a dataset broadly representative of the UK jobs market. Furthermore, the top job title recommendations for each candidate present a number of viable future career options. Both of these attributes demonstrate its suitability for use on a job site.

All code for this work can be found on `https://github.com/eddietheeagle1/career_path_recommendation`.

Contents

1 Introduction

1.1 Overview

The ultimate problem for any job site is presenting a candidate with their perfect job, or in practice, presenting the candidate with a selection of highly relevant jobs. Most job sites currently have a search box in which the user inputs a desired job title, location and perhaps a number of other parameters. This approach has the following problems:

- The job title could be broad and so result in an uninformative search - for example, job titles such as 'consultant', 'manager' and 'advisor'.
- A candidate with certain education, skills and experience might not want, or be able to, apply to all roles for even a very specific job title.
- Many candidates might not know what they want to do next. Anecdotally, this is a common situation.

As a result, searching for a new job can be a very painful and tedious process. The obvious way to address the above process is to personalise the recommendations using the candidate's CV or LinkedIn profile. This could filter out irrelevant roles and present possible job titles to the candidate which they had not considered previously.

1.2 Research Questions

There are a number of papers dealing with next job prediction and its closely related problem: next basket recommendation. Details can be found in chapter 2.

In the key paper, Li et al. (2016) [1] use a contextual LSTM to predict the next role given a candidate's previous roles, skills, education and location. The model is trained separately for two different industries - finance and technology - and is trained on data from over one millon LinkedIn profiles.

While Li et al. (2016) [1] achieve some very good results, their model needs to be trained separately for different industries and so it cannot deal with candidates who move industry. Many different models would need to be trained to achieve good coverage of an economy - the finance and technology industries in the UK employ 2.2m [2] and 1.64m [3] respectively, only 12% of the UK workforce [4]. These industries are also slightly unusual in that many of their jobs have very specific skills. For example, the most important skill for a java developer is 'java', a skill needed for few other professions. On the other hand, the most important skill for an estate agency sales negotiator is 'sales', a skill needed in many other roles.

Most job sites cover jobs in a number of different industries and so, any recommendation model used needs to work on many different industries as well as coping with candidates who move industry. Furthermore, the top suggested job titles for each candidate all need to be relevant if the model is to be used in production on a job site. Therefore, the research questions this work addresses are the following:

- Can a single model, trained on a broadly representative sample of the UK jobs market, achieve similar results to [1]?
- Are all of the top 10 job titles recommended by this model relevant to each candidate?

1.3 Project Overview

The data used was from a company called Adzuna ('the company'), a job website that operates in many countries including the UK. The company has developed a tool called 'ValueMyCV' which aims to provide candidates with hints on how to improve their CV and predict their next role and salary. They have provided over 1.6m CVs that are used in this work.

There are a number of difficulties with dealing with data as unstructured as the data in CVs. First, the CVs need to be parsed into a database-like format - the company use a third party to do this. One of the main issues is 'normalising' the raw text of a job title or a company name into distinct, non-overlapping categories. For example, the job titles ‘java programmer’ and ‘java developer’ are the same role and need to be recognised as such. Other difficulties arise when dealing with vague job titles such as ‘manager’ which can describe a wide variety of different roles depending on the company and industry. A further explanation of these challenges can be found in section 3.3.

The first step in developing a non-trivial model was to be able to represent various possible inputs to the model as vectors of numbers. Ideally, these representations should encode semantic similarity between entities. The company had already developed a 100-dimension embedding for skills using a word2vec approach, trained on over 10 million job descriptions. Most important was to develop a similar representation for job titles. Different approaches were tried. The approach which worked combined the skill embeddings with a profile that the company had produced of the skills required for each job. For each job title, skills were added together, weighted by their importance to the role, to produce the 100-dimensional job embedding. These embeddings were visualised which suggested that a good representation had been learnt. Further details on the job embeddings can be found in chapter 4. Details on education and company representations can be found in sections 7.3 and 3.3 respectively.

The metric used to evaluate all models was mean percentile rank (MPR), the same metric used in [1]. For each candidate in the test dataset, a model predicting the last job title on a CV ranks all possible job titles from the most probable to least probable. The percentile rank of the actual last job title is measured against this list. So, for example, if the actual job title falls 1% of the way down the ordered list of probable job titles, its percentile rank is 0.01. This metric is averaged across all CVs to produce the mean percentile rank (MPR). Therefore the lower the MPR, the better the result. More detail and a discussion around this metric can be found in section 5.1.

Using these embeddings and this metric, a baseline model was produced to provide a comparison to the final model. The two most recent job titles for each CV were selected - the most recent job title was the one to be predicted, the second most recent job title provided the features to the model. An alternative feature set was also tried whereby the skills on the CV were input as features to the model. Embeddings were used to represent the features in both cases and the Gaussian Naive Bayes algorithm was used to train the model. For a threshold of 5 (i.e. removing rare job titles which appeared under five times), the MPRs were 0.132 and 0.0583 for the previous job and skills features respectively. The fact that the skills features performed better than the previous job features was not surprising. Skills on a CV are usually included in one section and it is rarely possible to assign specific skills to specific jobs. As such, the skills on the CV contain skills learnt on the job that is being predicted. More details on the setup and training of the baseline model can be found in chapter 5.

The Gaussian Naive Bayes algorithm was used for the baseline model as it was the only algorithm attempted that ran in a reasonable amount of time. This was mainly because there were a large number of classes to predict, usually over 1000, and several classical algorithms struggle with this quantity of classes. A method known as Error Correcting Output Codes was attempted in order to make this multi-class problem less computationally demanding, such that algorithm other than the Naive Bayes could be used. However, this did not produce a better result than the baseline model - more details can be found in chapter 6.

The main model in this work (‘neural model’) was based on the model used in [1] (‘NEMO’). Like NEMO, the neural model is a contextual LSTM - the job titles in a career are modelled as a sequence using an LSTM but the input to the first cell is the context of a candidate, e.g. a candidate’s skills and education. There are a few subtle differences between NEMO and the neural model, aside from the fact that NEMO is trained separately for two different industries. NEMO predicts company and job title whereas the neural model just predicts job title. NEMO uses skills, education and location as context whereas the neural model just uses skills and education. The structure of the neural model can be found in Figure 8 and all the differences with NEMO are detailed in section 7.5.

The first iteration of the neural model used just skills as the context. Skills learnt on a role are the best way to describe a job as shown by the results from the baseline model. Again for a threshold of 5, which excluded very rare job titles from the dataset, the MPR achieved by the model was 0.0225 - significantly better than the 0.0583 achieved by the baseline model. Importantly for the first research question, this result is comparable with the results achieved from the NEMO model. The NEMO model achieves a MPR of 0.0182 and 0.0253 for the technology and finance industries respectively. The results also look similar to NEMO when the MPR is plotted against both number of roles and frequency with which the job title appears. A more detailed discussion of the results can be found in section 7.2 and a discussion on the extent of the comparability between NEMO and the neural model can be found in section 7.5.

Intuitively, education should make a difference for the first few roles in a career but become less and less important after that. The inclusion of education as additional context into the model should therefore increase the performance of the neural model. However according to the results, adding education to the model makes minimal difference and sometimes has a negative effect. This could be because of the weak education representation chosen, or the fact that skills adequately describe the context which education provides (a candidate who attended a top university is likely to have more valuable skills to match), or a combination of the two. The full results and discussion for the neural model with skills and education context can be found in section 7.3.

To understand why education was making no difference to the results, a model with just education as context was compared to a model with no context at all. The model with no context was simply an LSTM model with the first job input into the first cell rather than the context input into the first cell. Including just education as context produced much worse results than having just skills as context, producing MPRs of 0.0329 and 0.0225 respectively. However, the no context model produced an MPR of 0.0341 - the education context did make a slight difference. Therefore, it seems that the skills do include most of the context that education brings but undoubtedly, the representation for education could also be improved such that it makes more of a difference to the results. The results comparing these models are described and discussed in more detail in section 7.4.

While the results are good and broadly comparable to the NEMO model in [1], a candidate using this model on a website ultimately wants the top suggested jobs to be relevant, as per the second research question. Even if the 'correct' job is in the first percentile, if the top ten jobs, say, are not at all relevant to the user, then the model will appear to be performing badly. The neural model performs well in this scenario as well. Previous roles are remembered and the suggestions seem to take account of the 'step up' in responsibility that many candidates desire when they change roles. In the neural model with the skills context, the top results are reasonable - specific examples can be found in sections 7.2. These suggestions could also help to provide some inspiration to the candidate, by presenting options that they might not have thought about before.

There are, however, a few points of note for the neural model. As discussed for the baseline model, skills are being input to the model which have been learnt on the job that is being predicted - this artificially inflates the model performance. This is also the case for NEMO. This means that the neural model will be good at suggesting roles which have a skills overlap with the current role but less good at suggesting slightly more unusual jobs with less of a skills overlap. Secondly, there are a number of very vague job titles such as 'manager' and 'consultant'. As these job titles are popular, they frequently occur in the top ten recommendation job titles. However, recommending these as job titles to a candidate is meaningless unless some additional context, such a company, is provided. A more in-depth discussion concerning both of these points is included in 7.6.

A summary of the implications of this work can be found in the conclusion, chapter 8. A list of software used to implement the above is detailed in appendix A.

2 Literature Review

A variety of different problems in the jobs market have been tackled using a machine learning approach: Wang et al. (2013) [5] investigate the best time for a career switch, Kisaoglu (2014) [6] looks at predicting employee turnover and Li et al. (2017) [1] investigate predicting a next career move. In addition, there are a number of blog posts and other resources on the internet tackling questions such as estimating salary based on job descriptions and using machine learning to streamline the hiring process.

2.1 Overview of Next Basket Recommendation Papers

The problem of next job title prediction is at heart a recommendation problem: given a certain sequence and perhaps some additional context, it is possible to predict the next most likely values in the sequence? This problem has gained in prevalence recently because of the growth in online shopping and services. Recommending highly relevant products to each user for an e-commerce company is of huge value, especially when a large number of products are available.

Historically, there have been two different methods for next basket recommendation: Collaborative Filtering (CF) approaches, of which matrix factorisation is an example, and Markov Chain approaches (MC). CF is good at giving a relevant context for each user (e.g. whether the user likes music based on previous purchases) but not so good at modelling sequential features. MC is the reverse: good at modelling sequential features but less good at representing a context for each user [7].

Intuitively, an optimal algorithm would contain elements of both approaches, both learning a representation of each user but also allowing for a user's likes and interests to change over time. Various ways to do this have been suggested: personalised Markov chains by Rendle et al. (2010) [8], a hierachical model by Wang et al. (2015) [9] and, current state-of-the-art, a neural approach by Yu et al. (2016) [7].

2.2 Next Job Prediction Papers

However, the task of next job recommendation requires a different emphasis from next basket recommendation. First, a user can only have one job but can have several items in a basket. Secondly, it appears that the average number of transactions per customer for the datasets in [7] is higher than a reasonable number of jobs per person and so the length of the sequence to train on will typically be shorter. Lastly, for job recommendation, prior knowledge about the candidate's skills and education can be inserted in addition to the career history.

There have been a number of different feature-based approaches employed to tackling this problem.

Paparrizos et al. (2011) [10] predicted the next company worked at, given the current company. They had two types of features - features about the company (company name, industry, number of employees etc) and features about the individual (number of jobs, degree obtained etc). However, they only included candidates in their dataset who worked in very large companies.

Qu et al. (2016) [11] predicted the company size and position for a next job. They used features such as company size (small, medium or large) and position (internship, ordinary staff, senior staff, management) for the current role. However, they only predict the rank of the title rather than the title itself (e.g. they predict 'senior staff' rather than 'senior technical manager') and they only focus on the ICT jobs.

Both of the above papers artifically restrict the number of classes they consider: [10] by only focussing on very large companies and [11] by only focussing on the position rank rather than the position itself. In the jobs market, there are a few popular job titles but a large long tail of less frequently appearing jobs titles - the same is true for companies. Therefore, if a model is to be used on a job site, all job titles need to be included.

Li et al. (2017) [1] include the long tail of job titles and companies in a contextual LSTM model which they call 'NEMO'. The structure of NEMO is shown in Figure 1.

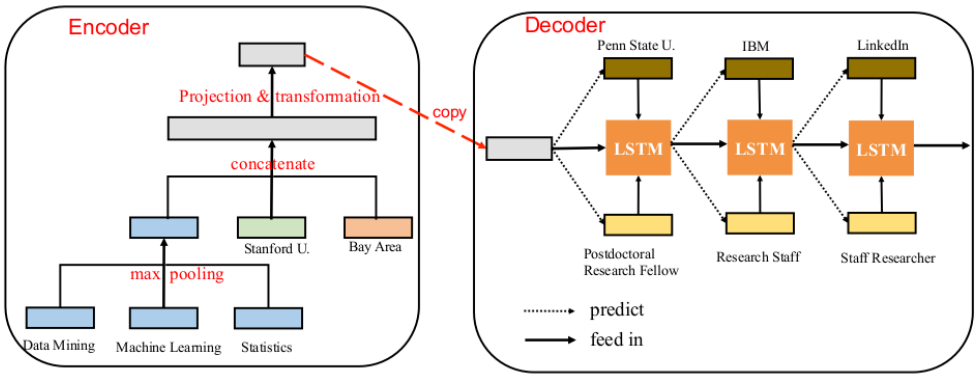

Figure 1: The structure of NEMO, taken from [1].

The 'Encoder' section of Figure 1 encodes the context for each candidate i.e. their skills, education and location. The authors consider these the most relevant contexts to predict a candidate's future career path. As users have different numbers of skills, the skills are max-pooled such that a consistent representation size can be fed into the model. The intuition behind this is that some skills are more important in a career move than others. The max-pooled skills are then concatenated with a representation of university and location and then fed through a non-linear layer such that the output is the same size as the job and company embedding.

The 'Decoder' section of Figure 1 uses an LSTM to learn long-term dependencies between the context and previous roles and companies. The context is input into the first cell of the LSTM and jobs/companies in a career path are input subsequently. The LSTM predicts both the company and the job title for the next role. Company and job title output from all the LSTM cells are fed through separate sampled softmax layers and a cross-entropy loss is used to train the model.

The NEMO model only predicts next roles for the finance and technology sectors, which as stated in chapter 1, only employ 12% of the UK workforce. These industries were probably chosen as roles in these sectors tend to have very specific skills and therefore make the next role easier to predict. This work predicts the next roles for a more representive dataset of the UK workforce. This brings additional challenges which were not as prevalent in [1] - it is more challenging to learn representations of job title and company (because there are far more possibilities) and undescriptive job titles such as ‘manager’ are far more prevalent. These challenges are detailed in section 3.3.

3 Background

3.1 Company Overview

Adzuna is a job aggregation site which has the tagline ‘every job, everywhere’. With traditional jobsites, users can only discover jobs which a recruiter or company has chosen to post on that specific site. Adzuna indexes every job from a number of different websites and this, in theory, means that a search for a specific job on its website will show all current advertised vacancies for that particular location. It earns revenue by charging per click for each job advert. Adzuna operates in a number of different countries including France, Germany, Canada, Australia and the USA.

It has developed a tool called ‘ValueMyCV’ for users of the site. At the time of writing, this tool:

1. Gives the user suggestions on improving their CV - for example, adding a summary, closing a work history gap, adding their education and advising on its length.
2. Advises the user what their next job should be and, as the name of the product would suggest, what they should be earning.

They have provided over 1.6m CVs that are used in this work.

3.2 Exploratory Data Analysis

Below are a number of different visualisations to give a sense of the type of CVs that are included in the dataset:

As can be seen in Figure 2, there are a small number of highly occurring job titles. Unspecific job titles such as ‘manager’ and ‘consultant’ cause particular issues, as explained in section 3.3.

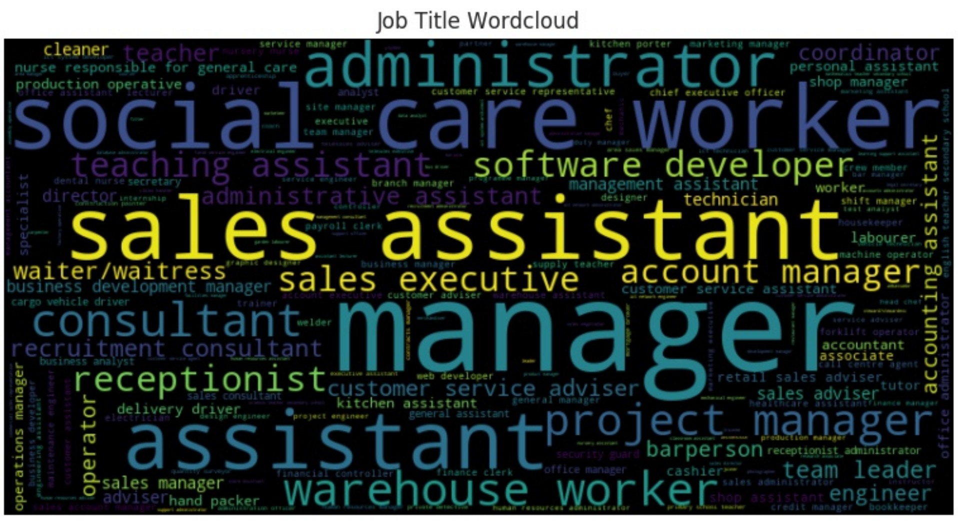

Figure 2: A wordcloud of the most frequent job titles for the most recent job. The larger the job title, the more frequently it occurs.

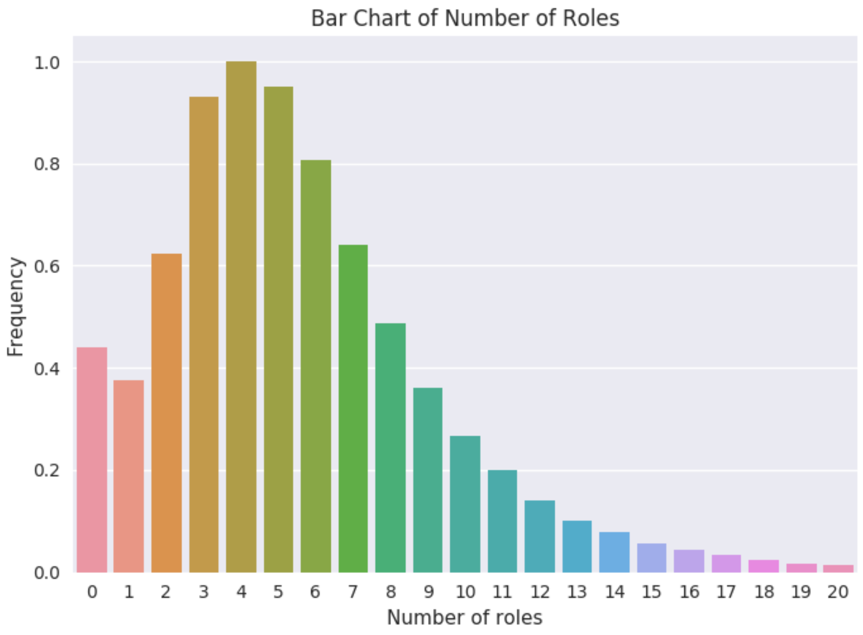

Figure 3: A bar chart of normalised frequency against the number of job titles for CVs in the dataset. There are CVs with more than 20 roles but these have not been shown for display reasons. It is also likely that anything much greater than this number is the result of a parsing error.

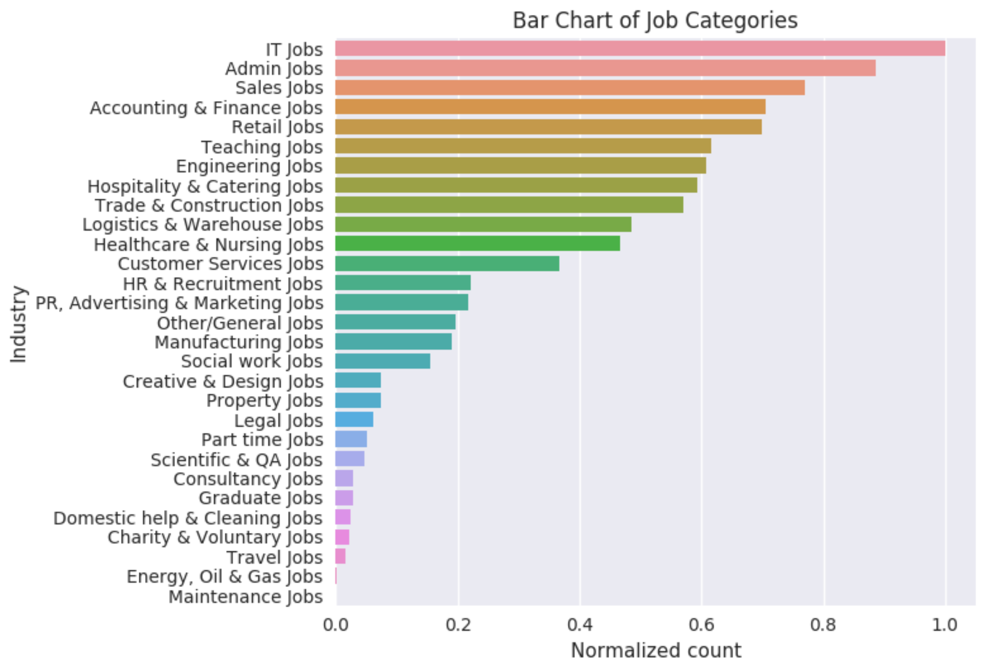

Figure 4: A bar chart of the industry-standard job category for the most recent job. The category is determined by job function rather than company function e.g. a lawyer at an IT company would work in the 'Legal Jobs' category.

Figure 3 shows that the modal number of roles is 4 and that the distribution of number of roles per CV is skewed towards low numbers of roles. CVs with few roles could present problems in predicting the next role as any model would have less information with which to predict subsequent job titles. Intuitively, one would think that fewer roles would make education history and skills more important.

As shown in Figure 4, jobs in the CVs are spread over many different industries. The categorisation of different jobs appear sub-optimal as the categories are not mutually exclusive - for example, a graduate nurse would be in both the 'Graduate Jobs' and 'Healthcare + Nursing Jobs' categories. However, these are the categories that are used by the industry.

Further exploratory data analysis can be found in Appendix B.

3.3 Challenges

CVs and the data on them are very unstructured and so there are a number of pre-processing steps required before any meaningful analysis can be done. These are outlined below - each one is a potential source of error:

The CVs were provided in a JSON format after they had been 'parsed' i.e. converted into a database-like format. The company use a third-party to do this. This parsing is not perfect but on the whole, it works well. As a further pre-processing step, duplicate entries were removed by discarding all but the most recent CV for each email address. The columns provided in the JSON were:

address line, country code, education history, email address, employment history, normalised employment history, filename, first name, language of CV, languages spoken, last name, municipality, number of referees, number of web addresses, objective present, phone number, postal code, region, revision date, skills, summary present, total months management experience, total months work experience.

There are 1 million unique job titles in the dataset but many of those job titles describe the same role. For example, 'bartender', 'bar man', 'bar staff', 'staff bar' and 'barstaff' all describe the same role, that of a bartender. As the job title is a free text field on a CV, the number of possible variations, including misspellings, of the same job title is very large which in turn makes it difficult to perform robust analysis. The company has developed a job title normalisation process which uses an ontology and various systematic approaches. The reduces the number of possible job titles down to 5641 genuinely different roles.

There are similar issues with company names, except in this case, the number of possible companies is far greater than the number of job titles. For example, working for 'Barclays PLC', 'Barclays Bank', 'Barclays Ltd', 'Barclays Capital', 'BarCap', 'Barclays UK' means working for the same organisation. Further complications arise when companies use a brand name that is not the same as their registered companies name. The company has not done much work on this area and as such, any analysis of the 'company' field is not included in this work.

Similar issues occur when normalising university name (and qualification type) but luckily the number of different universities is small. The company has developed a simple normalisation process which cleans the university name and maps it to its Shanghai University Ranking, a measure of the university quality. More details of this representation of university name can be found in section 7.3.

One of the most difficult issues to deal with is that of vague job titles. Many job titles are specific and give a good idea of role's responsibilities - 'Key Stage 2 supply teacher' for example. A job title such as 'manager' however, could refer to almost any industry and skill set. As this job title is so generic, it appears frequently in the dataset, as shown by Figure 2. Additional context, such as company, would make these vague job titles more meaningful but, as discussed above, this was not possible for this work.

Intuitively, the best way to describe a role is through the skill set acquired on the job. However, skills cannot easily be assigned to roles on a CV. A typical CV has a history of work experience, perhaps with a short description for each role and a separate skills section. This makes it hard to associate skills with specific roles. Further implications of this are discussed in section 7.6.

4 Embeddings

To be able to answer the research questions, it was essential to be able to represent jobs and skills as vectors of numbers. A simple representation is a one-hot embedding where the skill/job in question is represented by a one and all other skills/jobs are represented by a zero. However, as there are 6826 unique skills and 5641 unique normalised jobs, this representation is very sparse. Furthermore, intuitively, semantically similar skills (e.g. 'Java' and 'Python') should be close together in this 'skills space' and whereas semantically dissimilar skills (e.g. 'Python' and 'Organic Farming') should be far apart. This is not necessarily the case for a one-hot encoding.

The company had already learnt a good vector 100-dimensional representation of skills using a word2vec approach [15], trained on 10 million job descriptions. In addition, the company had an ontology of all the skills that compromised each (normalised) job title, complete with the TF-IDF score [16] for each skill and job title.

4.1 Method

The first approach attempted was a skipgram model ([15]) on the skills profiles for each (normalised) job title. The idea was that similar jobs require similar skills and so the skipgram model could learn a job representation based on this. However this was not successful as the algorithm struggled to relate similar skills. A java and python developer, for example, might not have many overlapping skills as they specalise in different languages but the domain they are working in is very similar.

The second approach was to learn the job embeddings using the resources that the company had already calculated, in particular using the skill embeddings to encode the similarity between sets of skills. Therefore, the embedding for each job was calculated by adding the embedding for each constituent skill. These skills were not weighted equally though as for most jobs, some skills are much more important than others. For example, the skill 'account management' for an audit accounts manager is more relevant than the skill ‘administration’ and so should have a larger effect on the embedding. The TF-IDF weighting [16] was used to weight each skills appropriately for each job title. In this way, the more important skills for a job title assume a high weighting and therefore contribute more towards the embedding.

4.2 Results and Discussion

The validity of the embedding was checked using the t-distributed stochastic neighbour embedding algorithm (t-SNE) [17], a dimensionality reduction technique often used for visualising high dimensional spaces. Figure 5 is a t-SNE plot for teaching and software development jobs - only two sets of job titles are shown for clarity.

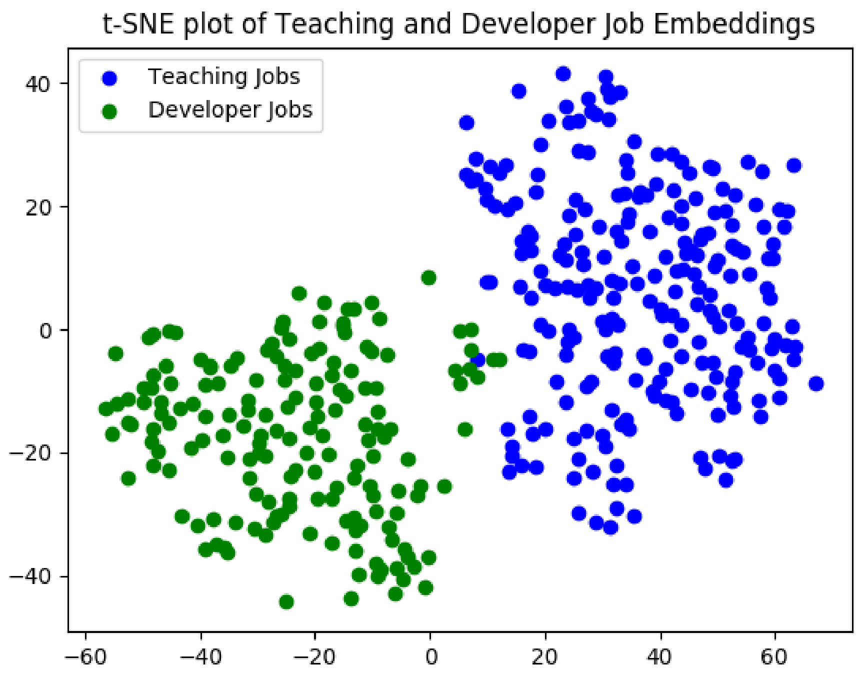

Figure 5: A two-dimensional t-SNE plot of all (normalised) teaching and software development job titles. The axes on this graph have no physical meaning.

As can be seen, the different categories of job are in distinct clusters which implies that the embeddings have learnt a useful representation of jobs. When visualising a greater number of jobs including jobs in other sectors, t-SNE shows similarly good results.

There is no obvious way to evaluate the quality of embeddings - a number of different approaches are detailed by Schnabel et al. (2015) [12]. The t-SNE visualisation provides some reassurance that a useful representation has been learnt but the ultimate test will be the performance of subsequent algorithms that use these embeddings.

5 Baseline Model

5.1 Metric

The metric chosen to evaluate the models in this work is of crucial importance. Accuracy is a metric often used to evaluate classification models in machine learning, especially when the number of classes is small. In this case, however, there are 5641 possible classes. The accuracy metric though would not distinguish between the correct job title being ranked as the second most likely job title and the 1000th most likely job title - they would both be marked as incorrect. There are similar problems for precision, recall and F1 metrics when calculated on just the most likely job title. A user on a job site is typically presented with a selection of roles instead rather than just a single role and so being amongst the top-ranked jobs is important.

Mean percentile rank (MPR) is a metric which does take account of the 'rank' of the correct job title. For each data point, the model ranks all the job titles in order of likelihood, with the top position being the most likely job title and the bottom position being the least likely job title. The actual job title is located in the list and its percentile is calculated. So if the actual job title appears 30% of the way down the list, it is in the 30th percentile and so has a percentile rank of 0.3. The MPR is calculated by carrying out this procedure for all the data points and taking the mean of the percentile ranks. A model with an MPR score of 0.5 is the same as randomly ordering the job titles, a model with an MPR score lower than 0.5 is better than random and a model with an MPR score higher than 0.5 is worse than random. Therefore, the lower the MPR score, the better the model.

The literature uses a few different metrics for evaluation. Papers on the next basket recommendation problem ([9],[7] - referred to in the chapter 2) tend to use metrics such as F1 score and Normalised Discounted Cumulative Gain on the top K items. However, Li et al (2015) [1] use the MPR metric and this will provide a good basis for comparison for the results in this work.

5.2 Method

Two basic approaches were used for the baseline model to predict the most recent job title:

- Using the previous job title as features
- Using the skills from the CV as features

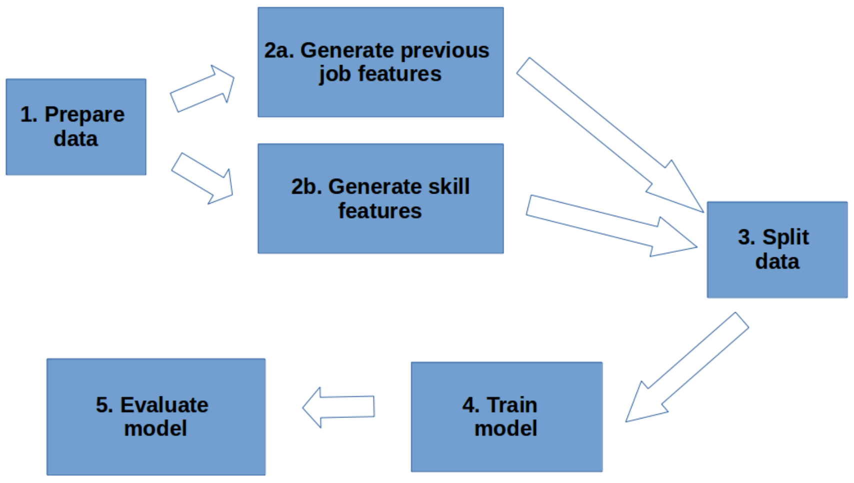
1. Prepare data
2a. Generate previous job features
2b. Generate skill features
3. Split data
4. Train model
5. Evaluate model

Figure 6: A flowchart detailing the main steps of the baseline method.

Figure 6 shows the workflow for the model:

1. **Prepare data**: 45% of the CVs did not contain a normalised job title in either the most recent job or the previous job, or did not contain two roles for more - these were removed from the dataset.

2. **Generate features**: Two types of feature representation were tried. The bag-of-skills representation used a one-hot encoding of skills i.e.

1 if the skill was present and 0 otherwise. As the number of unique skills was 6826 and the mean number of skills per candidate was 64, this feature set was sparse. The embedding representation used either the skill or job embeddings - the number of dimensions was therefore the size of the embedding. The bag-of-skills representation did not have any predictive power in every configuration attempted. Hence, both methods below only use an embedding representation.

   (a) *Previous job features*: The embedding of the previous job was used as the features. So, for example, if the user's previous job was a 'sales assistant', the features would be the job embedding for that role.

   (b) *CV skill features*: An embedding of the skills on the candidate's CV was used as features. All the skills on the CV were given an equal weighting.

3. **Split data**: The data needed to be split into a train and a test set so that performed could be evaluated. Splitting the CV data into a reasonable train and test set is non-trivial. The ideal split would result in the graphs of the CVs shown in section 3.2 looking similar for both the train and test set i.e. same distribution of job titles, job categories, and number of roles. After some experimentation, a stratified split on the most recent job title was performed - the graphs shown in section 3.2 were very similar for the train and test sets. A threshold was also implemented such that rare job titles that appeared under a certain frequency were removed. An 80/20 split was used for the train/test set respectively.

4. **Train model**: Different algorithms were tried on the train set, each of which had differing levels of performance and run times. The main obstacle was the number of different classes to predict. Although the exact number of the classes depended on the size of the dataset and the threshold, the number of classes was usually in excess of 1000. Many algorithms struggled to train on so many classes in a reasonable run time. Other difficulties included the size of the dataset and the high dimensionality of the bag-of-skills representation. The Gaussian

Naive Bayes algorithm was selected as it was the only algorithm that could be run in a reasonable time.

5. **Evaluate model**: The algorithm was evaluated on the test set. The MPR metric was used to score the model (see section 5.1 for details).

5.3 Results and Discussion

The results for this baseline model are shown below. As shown in Figure 7, using skills from the CV as features resulted in a higher performance than using the previous job title. It is also noteworthy that the threshold, which determines the number of classes to predict, has a large effect on the performance. In general, MPR increased with an increasing threshold (i.e. less classes to predict), the exception being the generic skills with a threshold of 1.

The strong performance of the embeddings (much better than random performance) is further proof, in addition to Figure 5, that the embeddings have learnt a useful representation of job titles and skills. The Naive Bayes algorithm assumes that all features are independent of each other and the results suggest that this assumption is reasonable for the embeddings. The bag-of-skills approach is likely to have failed because of its high dimensionality but perhaps also because the Naive Bayes assumption is not as valid.

It is not surprising that the features comprising of CV skills are better than features from the previous job title. As described in the section 3.3, CVs typically have separate sections for skills and job history, making it difficult to associate specific skills with specific roles. As a result, skills that relate to the job that is being predicted are likely to be appearing in the feature set. This is therefore flattering the performance of the algorithm.

In addition, vague job titles such as ‘manager’ and ‘advisor’ are likely to be roles for which the CV skill features significantly outperform the previous job features. The job title of ‘manager’ appears very frequently in the dataset and could refer to almost any sector and skill set. It has 4826 out of 6826 skills in the skills profiles with skills as diverse as ‘construction’ and

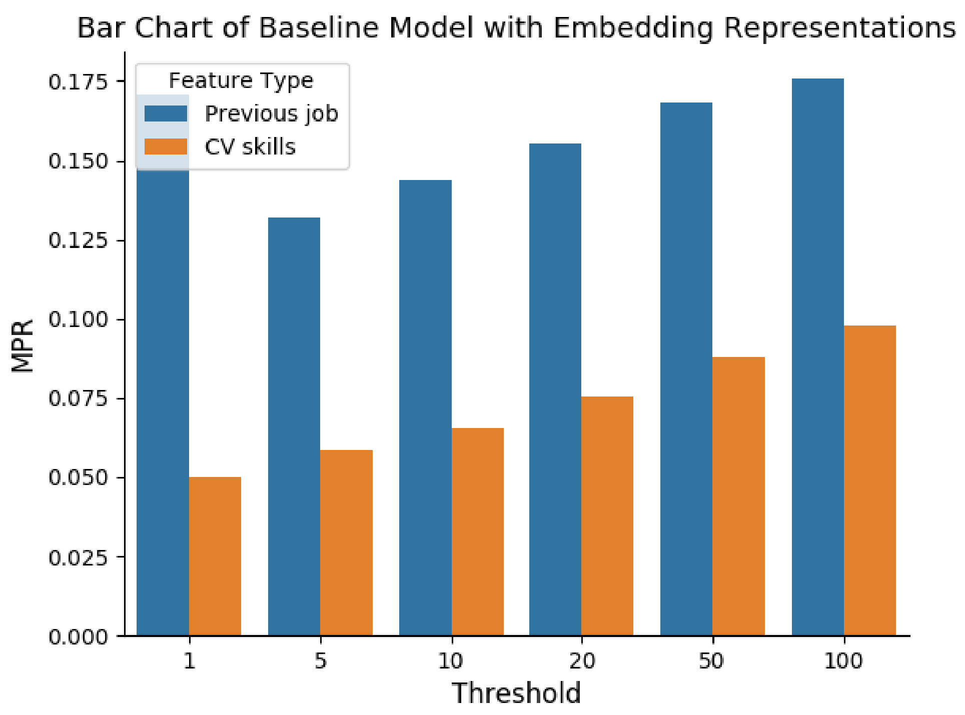

Figure 7: Results from the baseline model with both CV skills and previous job embeddings as features. The lower the MPR, the better. The threshold is the frequency at which a job title has to appear for inclusion in the dataset i.e. a high threshold means fewer classes to predict.

'fashion'. Clearly, this is not a good representation of any type of manager. This problem can be partially addressed through additional context such as skills, education or company.

There is a further discussion of both these points in section 7.6.

6 Error Correcting Output Codes Framework

For the Baseline model in chapter 5, the Naive Bayes algorithm had to be used as it was one of the few algorithm that had a reasonable run-time. This was mainly due to the large number of classes, which could be in excess of 3000 depending on the threshold which was set. Algorithms such Support Vector Machines (SVM), Random Forests and Logistic Regression do not scale well with the number of classes and quickly become intractable. If a SVM, for example, is trained on a dataset with 3000 classes, it has to learn 3000 different binary classifiers to obtain an overall result. If the number of classifiers to be learnt could be reduced, then algorithms with a better performance than the Naive Bayes could become tractable.

6.1 Method

One framework is called error correcting output codes (ECOC). In this framework, rather than a binary classifier being learnt for each class, a much smaller number of binary classifiers are learnt. The output of these classifiers are related to a class by a coding matrix which has the dimensions number of classes by number of classifiers. Therefore, each row of ones and zeros of this coding matrix defines a class. More detail on the ECOC framework can be found in [13].

An obvious question is how the coding matrix is calculated. According to [13], a good coding matrix has the following properties:

- Involves a small number of classifiers such that the representation of each class is efficient.
- Ensures that the classes are sufficiently separated - separation between two binary strings is usually calculated by Hamming distance.
- Results in accurate binary classifiers.

To learn accurate binary classifiers, similar jobs need to be grouped together for each column of the coding matrix. If zeros and ones were defined randomly in the coding matrix for each column, then the Hamming distance between two similar jobs, say ‘Spanish teacher’ and ‘French teacher’, would be similar to the Hamming distance between two dissimilar jobs. Ideally the Hamming distance between similar jobs would be smaller than it between less similar jobs.

An embedding has already been learnt to encode similarity between jobs and this embedding was used to fill out the coding matrix using a technique called Locality Sensitive Hashing (LSH) [14]. Random hyperplanes were drawn through the embedding space for each column of the coding matrix. For each hyperplane, jobs on one side of the plane were assigned 0 and those on the other side were assigned 1. The hyperplane was rejected if it did not result in roughly equally sized classes. This method ensured that the Hamming distance between similar jobs was small while the Hamming distance between dissimilar jobs was large.

6.2 Results and Discussion

Some results for the ECOC framework are shown in Table 1. Both results use a Support Vector Machine (SVM), use a threshold of 1 and were trained on only 1/20th of the dataset, for speed reasons. Other algorithms were used and performance was very similar to the SVM.

Number of Classifiers	Training time (mins)	MPR (3sf)
15	22	0.203
30	43	0.173

Table 1: Table containing the ECOC results.

The time taken to train the algorithm is significant - minutes rather than seconds. The reason for using the ECOC framework in the first place was to improve training time but this does not appear to be the case - just a subset of the dataset takes a significant amount of time to train. Part of the reason for this could be that this ECOC implementation did not fully utilise all the cores of the CPU.

The performance of the ECOC framework is also significantly worse than the baseline model. The MPR of 0.173 compares to an MPR of 0.0501 for the equivalent result for the baseline model. The results do improve with an increasing number of classifiers (as expected) but this comes at the cost of additional training time.

For both of these reasons, the ECOC framework was not explored further.

7 Neural Model

Even if the ECOC framework had produced good results, it would still have been fundamentally limited by its structure. The skills provide a context about a particular individual but do not take into account the candidate's career path to date.

7.1 Introduction

An ideal model would include the following:

- *Context.* Skills learnt during a role are the best way to describe that role. As discussed in section 3.3, job titles can often be quite vague. Skills acquired throughout a career and contained on a CV give a very good indication of possible future roles. For example, someone who has the 'accounting' skill is more likely to be an accountant than a data scientist. Education, both university and subject studied, can also provide useful context. Someone educated at UCL is more likely to become a banker than someone who did not go to university, for example, and someone who studied computer science is more likely to go into a technology role. Intuitively, you would expect education to become less and less important as the number of roles increases. Finally, location could be important as different areas of the country, especially outside London and the commuter belt, could have an industry focus. For example, someone working in Port Talbot in Wales is likely to work in the steel industry given that Tata Steel is by far the largest employer in the area. You would assume in general that all this context would be most useful towards the start of a career when the number of roles is few.

- *Career Path.* If a candidate has had a career change, they are unlikely to want to return to their previous career, even though they have the right skills for it. Therefore it is likely that the current role is going to determine the next role to a large extent. Often, a job move involves staying in the same industry but with a 'step up' in responsibility. A new job also likely involves elements of previous roles and so ideally the whole career path to date would be taken into account, with a greater weighting on the more recent roles. Company worked at can also provide useful context for each role, particularly for more vague job titles such as 'manager'.

The company had done a lot of work on representing skills and some work on representing university attended but not on subject studied, location or company. Working with CVs necessarily meant working with free text and so the time involved to generate reasonable representations can be substantial. This work, therefore, only considers skills and education as context.

7.2 Skills Context

7.2.1 Method

A neural approach was used to achieve this to incorporate the aims above - see Figure 8 for the detail of the model structure. The first iteration of the model just uses the skills as context - this is likely to be the most important part of the context. The model works as follows:

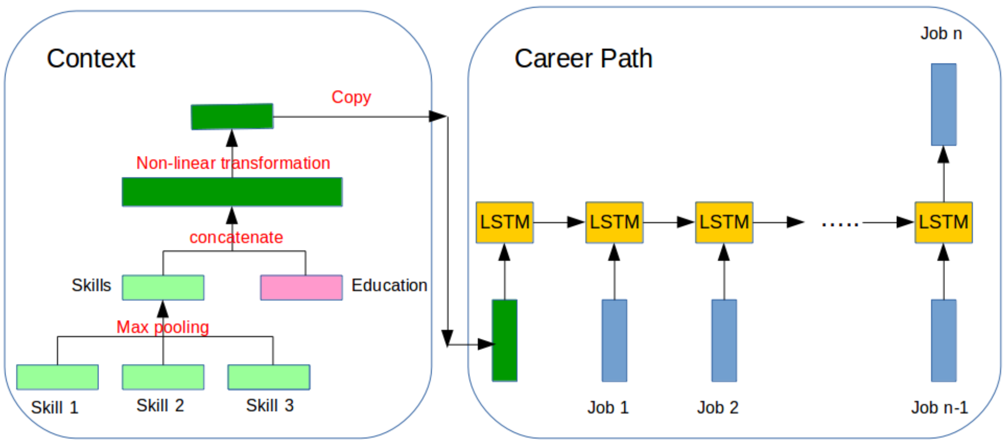

Figure 8: Structure of the neural model - a contextual LSTM. Only 3 skills are shown for clarity but the number of skills for each candidate is typically much greater.

- *Context.* The embeddings of skills on the CV are taken and a max pooling operation is performed dimension-wise such that all skills are represented by a single 100-dimension vector. If other contexts were being used, they would be concatenated with the skills here. This vector is fed though a non-linear layer with a tanh activation function to produce a 100-dimensional context. This context must be the same dimensions as the job embeddings.

- *Career Path.* This context is then fed into the first cell of a particular type of recurrent neural network called a Long Short-Term Memory (LSTM). The LSTM was first introduced by Hochreiter et al. (1997) [18] and is frequently used in modelling sequences as it can remember values over both short and long durations. This model needs to be able to remember the most important skills and jobs over the whole sequence to suggest the next job and so the LSTM is an obvious choice. After the context is put into the first cell of the LSTM, the first job (represented by the previously calculating embeddings) is fed into the second cell, the third job into the fourth cell and so on. When the penultimate job title is entered into the LSTM, the network outputs the probabilities for all possible last job titles via a softmax function. This is compared to the actual last job title and a cross entropy loss is calculated. This loss is then minimised during the training process.

This model is known as a contextual LSTM - the recurrence in the model captures the sequential nature of a career given some initial context and the LSTM is chosen to learn long term dependencies between different roles and skills.

The train/test split on the dataset was performed in same way as for the baseline mode - an 80/20 stratified split on the last job title. A threshold was again applied to filter out job titles which appears under a certain frequency in the dataset. The data was also pre-processed such that any candidate with either no jobs or a missing job title at any point in their career history was removed from the dataset. This reduced the dataset from 1.6m to 630k CVs, a significant reduction.

After much experimentation, the following network configuration was used.

50 hidden units were used in the LSTM - this was found to perform better than a higher number of hidden units or a stacked LSTM. The weight initialisation of both the linear layers and the LSTM was important: a Xavier initialisation [19] produced the best results. The Adam optimiser [20] was used with a learning rate of 0.001. The network trained with batch sizes of 1000 on an AWS ‘p2.xlarge’ GPU instance and took about 15 minutes to converge. After training, the network was evaluated by the metric described in section 5.1, mean percentile rank (MPR).

7.2.2 Results and Discussion

The results for the neural model, with just skills as context, are shown in Table 2 and Figures 9 and 10.

Threshold	MPR (3sf)
1	0.0196
5	0.0225
10	0.0256
20	0.0293
50	0.0353
100	0.0401

Table 2: Results of the neural model with just skills as context for different thresholds. The lower the MPR, the better.

Most importantly, these results are significantly better than the baseline of the Naive Bayes with CV skills as features. Taking into account previous job titles, in addition to the skills context, and modelling the career history as a sequence has made a big difference.

As with the baseline model, the threshold makes a big difference to model performance with the MPR increasing as the threshold increases. The frequency of job titles is a long tail with a few job titles appearing very frequently and a lot of job titles appearing infrequently. As the threshold increases, the number of classes rapidly decreases. As the number of classes decrease, the absolute position of the correct job title rises but because the number of classes decrease faster, the MPR decreases. In this case, the

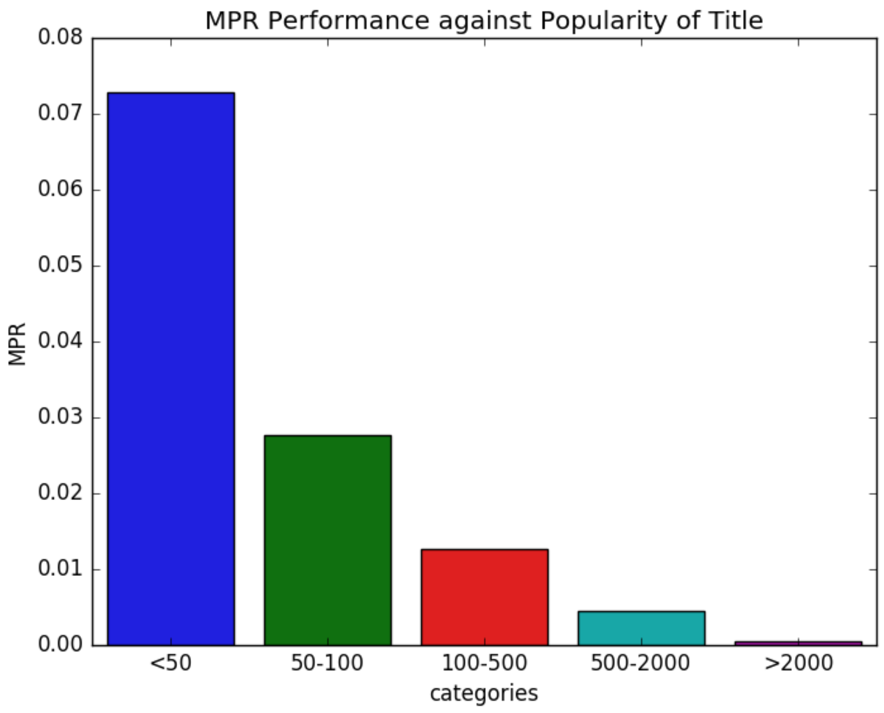

Figure 9: Results for neural model of performance against job title frequency for a threshold of 5. The mean MPR is shown for each category. These bins were chosen to align with the bins in paper [1]. The lower the MPR, the better.

MPR metric could be misleading. In the use case for this work, a candidate using this model to predict their next job would most likely want to see their top ten, say, most likely next job titles. So a model with a higher threshold and worse MPR could be a better model to use if the candidate is looking for a more common job. On the other hand, a user might want to see some slightly unusual but relevant job titles to provide some inspiration for their next career move. A threshold of 5 was chosen for subsequent neural models, an explanation of this can be found in section 7.6.

Both Figures 9 and 10 make intuitive sense. Figure 9 shows the model performance improving as a job title is seen more frequently - the neural model learns better with more examples. Figure 10 broadly shows the model performance improving with increasing experience. This is because the greater the number of roles, the more information the model has to learn from. The volatility of the graph for number of roles greater than 10

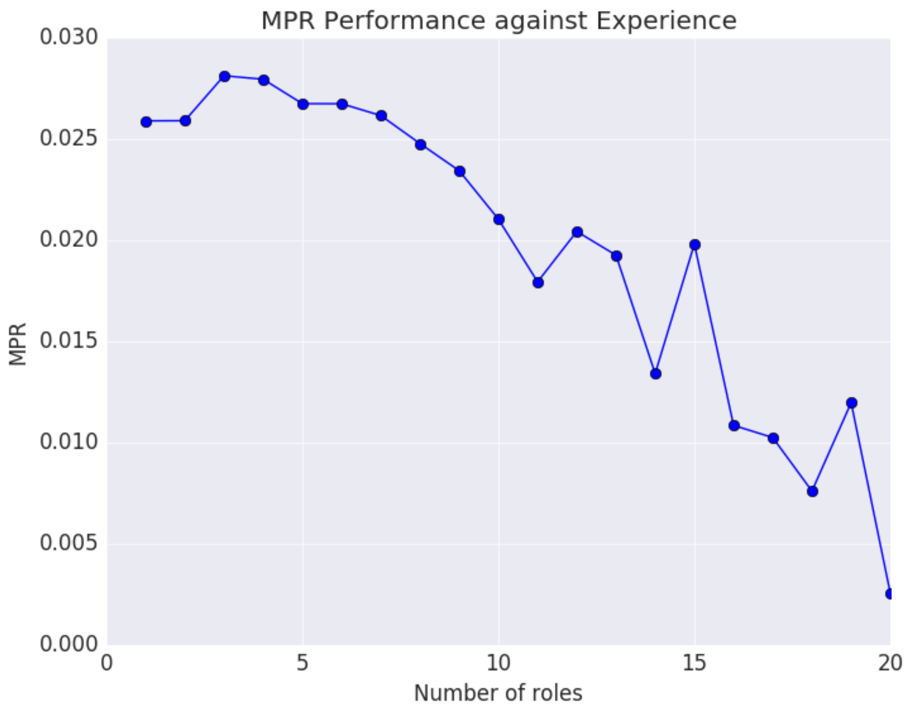

Figure 10: Results for neural model of performance against experience for a threshold of 5. The mean MPR is shown for each category. The lower the MPR, the better.

reflects the fact that there are many more CVs in the dataset with less than 10 roles, as demonstrated by Figure 3. So, a small number of unusual career moves for CVs with greater than 10 roles could make a large difference.

While the aggregate results are good, a candidate using the model ultimately wants all the jobs titles in the top few results to be relevant. Even if the correct job title is in the top 1% of all job titles, the model will be seen to be performing poorly if the top 10 results, say, are not relevant at all. Relevant job titles could also provide some inspiration for the user.

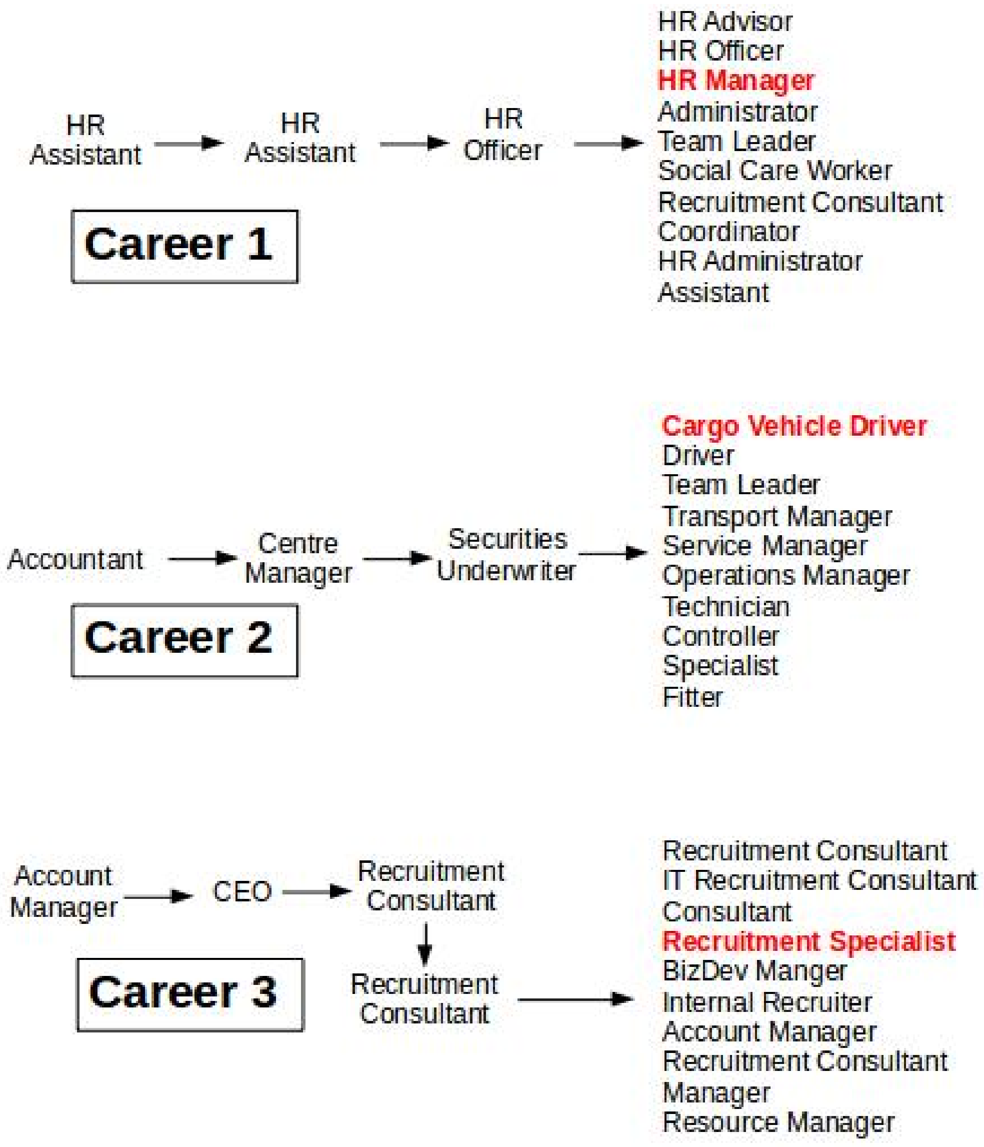

Figure 11: 3 sample career paths. The list on the right hand side of each career path is the most probable jobs according to the model with the highest probability first. The 'correct' job title is highlighted in red.

Career 1 in Figure 11 shows a candidate following a classic HR career path and is an example of the model automatically incorporating the idea of a 'step up' in responsibility when changing jobs. Both 'HR Advisor' and 'HR Manager' are steps up from the previous role of 'HR Officer' so it is encouraging that these job titles are right at the top of the list.

The candidate shown has in Career 2 in Figure 11 has, at first glance, an odd career path with roles that do not logically follow on from each other. This example highlights the importance of skills to the model and the efficacy of the LSTM in remembering the skills that were input at the start of sequence. All the suggestions made by the model do not have any obvious relation to the previous role but the skills have enabled the model to predict the 'correct' jobs title as the most probable.

Career 3 in Figure 11 is in some respects the most interesting of the 3 careers. The candidate seems to have started off as an 'account manager', tried to start their own business and subsequently pursued a career in recruitment. The top few recommendations relate to recruitment as would be expected. Interestingly, the model has not recommended the role of 'CEO', judging it not correct in the context, but has suggested the candidate's first role of 'account manager' as a possible option.

These sample career paths were not atypical and demonstrate that the top suggestions of the model look reasonable.

7.3 Skills and Education Context

The neural model with just skills as context works well as shown above. Intuitively, education should make a difference too. As stated at the start of this chapter, someone educated at UCL is more likely to be a banker than someone who didn't go to university.

7.3.1 Method

Specifically, building on some existing work that the company had done, university name was input into the model along with skills, as shown in Figure 8. Two types of representation for education were tried:

1. *1st Education Representation.* After some basic pre-processing, the university name was mapped to Shanghai Ranking, a ranking of the top 500 universities in the world. This ranking was used to come up with a one-hot seven dimensional representation as described below.

This representation had been used by the company previously with some success.

(i) 1 if university was in the top 30, 0 otherwise

(ii) 1 if university was ranked 31-150, 0 otherwise

(iii) 1 if university was ranked 151-500, 0 otherwise

(iv) 1 for an unranked university, 0 otherwise

(v) 1 if the individual did not go to university, 0 otherwise

(vi) 1 if the individual had a MBA, 0 otherwise

(vii) 1 if the individual had a PhD, 0 otherwise

2. *2nd Education Representation.* After applying the same basic pre-processing, a 504-dimensional vector was created. The first 500 positions were a one-hot representation of all the universities contained in the Shanghai Ranking, the last 4 positions were identical to the last 4 positions shown above. This representation was sparse but could capture dependencies between specific universities and job titles.

Two models, one for each education representation above, were trained in a very similar way to the skills context neural model.

7.3.2 Results and Discussion

The results are shown in Table 3. The results show that adding in education barely makes a difference to the model.

Context of Neural Model	MPR (3sf)
Skills	0.0225
Skills + 1st Education Representation	0.0225
Skills + 2nd Education Representation	0.0229

Table 3: Table comparing different context combinations for the neural model

They also suggest that the first representation is slightly better than the second representation for education. The second representation is very sparse - only 150 out of 500 ranked universities appear more than 10 times in the whole dataset. About 30% of the dataset has gone to university and only around 5% of the dataset has gone to a top 500 university. The education context attempts to capture the education quality which would intuitively give an indication as to the subsequent job titles. Given the sparsity of university, it makes sense that a denser representation would be better.

However, the representations described have a similar performance which implies that skills are a much more important context than university name. Intuitively though, education is an important determinant in the first couple of jobs in a career and becomes less and less important subsequently. The above result highlight the importance of a good representation. A one-hot representation cannot encode the semantic similarity between entities, universities in this case, and so is a much weaker representation than a good embedding. In addition, subject studied is likely to be just as important as university attended and perhaps the ideal representation of education would be an embedding of both subject and university.

7.4 Education Context

As shown in the results above, education does not contribute anything when combined with the skills and inserted as the context to the neural model. There are two possible reasons for this. The context included in education could already be provided by the skills. For example, a candidate from a good university will, on average, have learnt more valuable skills which will then be reflected in their future jobs. Alternatively, the representation itself might not be good enough to contribute to the model. An interesting question then is which of these explanations is true i.e. is a neural model with just education as context better than a neural model with no context at all?

7.4.1 Method

The setup for the two models was as follows:

- *Education Context.* The model is structured as in Figure 8 except it is only education that is fed into the non-linear transform - skills are not used at all. Both education representations were used again.
- *No Context.* As there is no context, the context part of Figure 8 is not used at all. The left most cell of the LSTM is also discarded and so the first job is fed into the first cell.

In addition to the pre-processing done previously for the neural model, all individuals with only one job or less were excluded from this dataset. The model with no context required CVs with at least two jobs on them and this amended dataset was also fed into the education context model to enable a like-for-like comparison. These models were trained in a similar way to the other neural models.

7.4.2 Results and Discussion

The results are shown in Table 4 and a comparison graph is shown in Figure 12.

Context of Neural Model	MPR (3sf)
1st Education Representation	0.0329
2nd Education Representation	0.0332
No Context	0.0341

Table 4: Table comparing the education context with no context for the neural model.

The results show that adding education as context does make some difference to the performance of the neural model. As before, the first education representation is more effective than the second representation. As Figure 12 shows, this difference arises from the first few roles in a career.

Education only making a difference in the first few roles in a career does make intuitive sense. Anecdotally, previous experience in similar roles count for more and more the greater the number of roles a candidate has had. The bigger picture though is that although education does make a

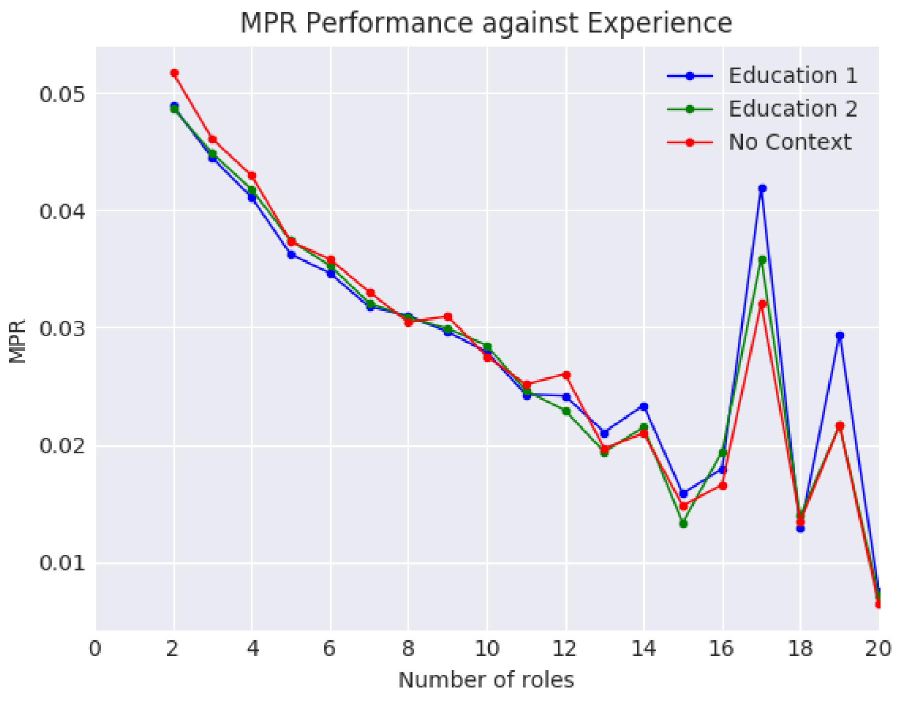

Figure 12: Comparison of results from the neural model with different contexts. A threshold of 5 is used for all models. The mean MPR is shown for each category.

small difference, including skills as context makes a significantly larger difference.

So, the benefit of including education is masked by the skills when they are included together as context but it is also clear that the representation of education does matter too. No doubt, using an embedding rather than a one-hot representation would make a further difference as discussed in section 7.3.

7.5 Comparison to NEMO Model

The results shown for the neural model are good and substantially better than the baseline model. The research question though is whether the results obtained by NEMO in [1] can be achieved using a single model and a dataset broadly representative sample of the UK jobs market. A comparison to [1] is therefore necessary.

The results from NEMO are compared to the neural model with just skills as context (as this model gave the best results) and with a threshold of 5. At a threshold of 5, the MPR for the neural model is 0.0225 - better than the ‘Finance’ dataset but worse than the ‘Computer’ dataset from NEMO which had job title MPRs of 0.0253 and 0.0182 respectively. It would seem that the research question can be answered in the affirmative: similar results can be achieved with a single model and a much broader dataset.

However, as is highlighted by Table 2, the MPR metric depends very much on various attributes and parameters of the dataset provided to the model - this makes comparison between the neural model and NEMO non-trivial. Furthermore, direct comparison between datasets is difficult as the authors of [1] do not provide the exact quantities for their dataset, only orders of magnitude, citing privacy concerns. The obvious difference is that the NEMO dataset only focuses on two industries whereas this dataset compromises of job titles from many different industries. Aside from that, a rough, high-level comparison between the datasets is contained in Table 5.

Attribute	NEMO	Neural Model
CVs	>1M	640K
Training positions	>10M	1.9M
Job Titles	>10K	5.5K

Table 5: Differences between the datasets for NEMO and the neural model.

The number of CVs or profiles is larger in the NEMO dataset but as this dataset is used to train two separate models, it is possible that both the neural model and NEMO are fed datasets of similar sizes.

NEMO has a much larger number of training positions than this neural model. It seems likely that the number of positions per CV is higher in the NEMO dataset than the Adzuna dataset but it is hard to be sure because of the approximate number provided.

The number of unique job titles in NEMO is less than for the Adzuna dataset and perhaps significantly less. Although this might seem counter-intuitive given that NEMO only focuses on two industries, it depends on the amount of pre-processing done to 'normalise' the raw text job titles. Adzuna is focussed on all the UK job market and so subtle differences between different technology and finance jobs in the LinkedIn dataset will probably be normalised to the same job title in the Adzuna dataset. In particular, Adzuna removes all seniority references and deals with them separately. The LinkedIn dataset has the job titles 'Technical Project Manager' and 'Senior Technical Project Manager' whereas Adzuna dataset only includes the normalised role of 'Technical Project Manager'.

The threshold chosen made a huge difference to the MPR as shown by Table 2. Therefore, a question that arises is what threshold should be picked to ensure the results from NEMO and this neural model are as comparable as possible. The NEMO paper does not include job titles in the dataset that have an occurrence of less than 10. All the results after Figure **??** have been reported with a threshold of 5. This is not an exact science but this seemed appropriate given that the number of training positions is significantly less in the Adzuna dataset.

The neural model is very similar to NEMO but with some important differences. The most important difference is that NEMO is trained separately for different industries whereas the neural model is just trained once on a dataset containing many industries. Diagrams of the structures of NEMO and the neural model can be found in Figures 1 and 7 respectively. The main differences are:

- *Context.* The context includes university name and location in addition to the skills.
- *Career Path.* Both the job title and the company is predicted rather than just the job title. The loss is also calculated across all job titles rather than just the last job title i.e. the model is a many-to-many

contextual LSTM rather than a many-to-one contextual LSTM.

Most interesting, though, is to compare the results between NEMO and the neural model. The same graphs as Figures 9 and 10 are plotted in [1] and so a comparison can be made. The same caveats about comparing the datasets apply as before.

Figure 9 and Figure 4 in [1] have exactly the same shape as expected - the more frequent the job title the better the performance of both models. The scale is different though; for example, in the '<50' bucket, NEMO performs at an MPR of around 0.27 whereas the neural model scores 0.072. The disparity could be explained by the larger size of the NEMO dataset - if the dataset of the neural model is smaller then proportionally the less frequent examples have been seen more often. A larger part of the explanation though is probably the embeddings. A next job would very rarely be completed unrelated to previous roles and so high quality job and skill embeddings would have a large impact in predicting job titles that occur infrequently. For this reason, it appears as though the job and skill representations for this model are superior to those used for the NEMO model.

Figure 10 and Figure 5 in [1] have a similar shape too. The more experienced an individual, the easier it becomes to predict their next role, as the models have more information from which to make a prediction.

7.6 Summary

The results for the neural model are significantly better than the baseline model and similar to the results in the NEMO model. Direct comparison, though, is hard because of the differences in the datasets. Furthermore, the indication is that the model would work well in a production environment. All the top job titles suggested by the model look reasonable, and even if the correct job title is not in the top 10, say, it could well provide some inspiration for a candidate unsure as to what do next.

However, there are a couple of important points to note about the neural model as it stands.

Firstly, as mentioned in the discussion around the baseline model in chapter 5, the evaluation score is misleading. In the test set, skills that have been learnt on the job that is being predicted are included as an input to the model. This was unavoidable as a CV typically has just one skills section and so it is not possible to assign skills learnt to specific roles. This is also the case for the NEMO model.

An interesting question is whether this is just a problem for the evaluation or a problem for candidates who will be using the model as well. With the current setup, the neural model is certainly better at recommending job titles which have a skills overlap with previous jobs. A typical job move involves either moving to a similar job at another company or taking on a role with slightly more responsibility. In both these scenarios, the model would work well as there is a large skills overlap. The model would be less good for a slightly unusual move, for which the skills overlap is much less although it should be picked up if a particular 'unusual' move happens frequently. It is perhaps unreasonable to expect the model to capture unusual moves that occur infrequently.

Secondly, as discussed previously, there are a number of vague job titles that appear frequently such as 'manager', 'consultant' and 'advisor'. As these appear often, they are frequently recommended in the top 10 most likely future job titles. Recommending these as potential job titles though is not helpful if no additional context is given to the output. For example, being recommended 'manager at PwC' is far more useful than just being recommended the job title of 'manager'. In the short term, these vague titles could just be removed in the output that is shown to the candidate but in the longer term, it would be better to incorporate company in the output as well (like in [1], the NEMO paper).

The results overall highlight the importance of a good representation for the inputs to the model. Skills had a very good representation and as such made a big difference to the performance. Education had a much weaker representation and made less difference than might have been expected. It seems as though the best way to improve this model going forward would be to improve or include context representations such as university name, subject studied and location, and also to add company to the predicted output. It would also be interesting to add an attention mechanism to the model to add interpretability to the results.

8 Conclusion

The first research question that this work attempts to answer is whether the results obtained by NEMO in [1] can be achieved using a single model and a dataset representation broadly representative of the UK jobs market. The answer is a qualified ‘yes’. The results obtained by the neural model are broadly comparable to those achieved by the NEMO, with a few caveats. Furthermore, in answer to the second research question, the top job title recommendations made by the neural model are all reasonable and so the model could be used on a jobs website.

The main strengths of this model are its ability to recommend jobs in any industry for any career path and to provide a selection of highly relevant job titles which can present options to the candidate which they might not have considered before. Its main limitation is that it fails to provide sufficient context for vague job titles like ‘manager’.

Suggestions for future work to improve this model are:

- Improving the representation for university name by training an embedding
- Learning representations for subject studied at university and location for additional context.
- Predicting company name as well as job title, in particular to provide more context for vague job titles.
- Adding an attention mechanism to the neural model to add interpretability to the results.

References

[1] Liangyue Li, How Jing, Hanghang Tong, Jaewon Yang, Qi He, Bee-Chung Chen. *NEMO: Next Career Move Prediction with Contextual Embedding.* `http://team-net-work.org/pdfs/LiJTYHC_WWW17.pdf`.

[2] TheCityUK. *Key Facts about UK-Based Financial and Related Professional Services.* `https://www.thecityuk.com/assets/2017/Reports-PDF/UK-Key-Facts-2017-updated.pdf`.

[3] Tech City UK. *Tech Nation 2017: At the forefront of global digital innovation.* `http://technation.techcityuk.com/`.

[4] Office for National Statistics. *UK labour market: August 2017.* `https://www.ons.gov.uk/employmentandlabourmarket/peopleinwork/employmentandemployeetypes/bulletins/uklabourmarket/latest`.

[5] Jian Wang, Yi Zhang, Christian Posse, Anmol Bhasin. *Is is time for a career switch?*. Proceedings of the 22nd international conference on World Wide Web, pp 1377-1388.

[6] Zehra Ozge Kisaoglu. *Employee Turnover Prediction using Machine Learning.* `http://etd.lib.metu.edu.tr/upload/12617686/index.pdf`.

[7] Feng Yu, Quang Liu, Shu Wu, Liang Wang, Tieniu Tan. *A Dynamic Recurrent Model for Next Basket Recommendation.* SIGIR16.

[8] Steffen Rendle, Christoph Freudenthaler, Lars Schmidt-Thieme. *Factorizing Personalized Markov Chains for Next-Basket Recommendation.* WWW 2010.

[9] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, Xueqi Cheng. *Learning Hierarchical Representation Model for Next Basket Recommendation.* SIGIR'15.

[10] Ioannis Paparrizos, B. Barla Cambazoglu, Aristides Gionis. *Machine Learned Job Recommendation.* Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011.

[11] Hongyang Qu, Zhiwen Yu, Zhiyong Yu, Huang Xu, Bin Guo, Xing Xie. *What is My Next Job: Predicting the Company Size and Position in Career Changes.* 2016 IEEE TrustCom-BigDataSE-ISPA.

[12] Tobias Schnabel, Igor Labutov, David Mimno, Thorsten Joachims. *Evaluation methods for unsupervised word embeddings.* Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 298307.

[13] Xiao Zhang, Lin Liang, Heung-Yeung Shum. *Spectral Error Correcting Output Codes for Efficient Multiclass Recognition.* 2009 IEEE 12th International Conference on Computer Vision (ICCV).

[14] Alexandr Andoni, Piotr Indyk. *Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions.* Commun. ACM, 51(1):117122, 2008.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space.* arXiv:1301.3781.

[16] Juan Ramos. *Using TF-IDF to Determine Word Relevance in Document Queries.* `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.1424&rep=rep1&type=pdf`.

[17] Laurens van der Maaten, Geoffrey Hinton. *Visualizing Data using t-SNE.* Journal of Machine Learning Research 9 (2008) 2579-2605.

[18] Sepp Hochreiter, Jurgen Schmidhuber. *Long Short-Term Memory.* Neural Computation 9(8): 1735-1780, 1997.

[19] Xavier Glorot, Yoshua Bengio. *Understanding the difficulty of training deep feedforward neural networks.* 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010.

[20] Diederik P.Kingma and Jimmy Lei Ba. *Adam: A Method for Stochastic Optimization.* ICLR 2015.

Appendices

A Code Implementation Details

Python 3.5 was used the implement the methods described in Chapter 3. In addition to the basic Python language, the following open source python libraries were used:

Markdown 2.2.0
Pillow 4.1.1
PyYAML 3.12
Werkzeug 0.12.2
awscli 1.11.104
backports.weakref 1.0rc1
bleach 1.5.0
botocore 1.5.67
certifi 2017.4.17
chardet 3.0.4
colorama 0.3.7
cycler 0.10.0
docutils 0.13.1
fuzzywuzzy 0.15.0
gmplot 1.1.1
h5py 2.7.0
html5lib 0.9999999
idna 2.5
jmespath 0.9.3
matplotlib 2.0.2
numexpr 2.6.2
numpy 1.13.0
olefile 0.44
pandas 0.20.2
pip 9.0.1
protobuf 3.3.0

pyasn1 0.2.3
pyparsing 2.2.0
python-Levenshtein 0.12.0
python-dateutil 2.6.0
pytz 2017.2
requests 2.18.1
rsa 3.4.2
s3transfer 0.1.10
scikit-learn 0.18.2
scipy 0.19.0
seaborn 0.7.1
setuptools 36.0.1
six 1.10.0
sklearn 0.0
tables 3.4.2
tensorflow 1.2.0
urllib3 1.21.1
wheel 0.29.0
wordcloud 1.3.1

B Additional Exploratory Data Analysis

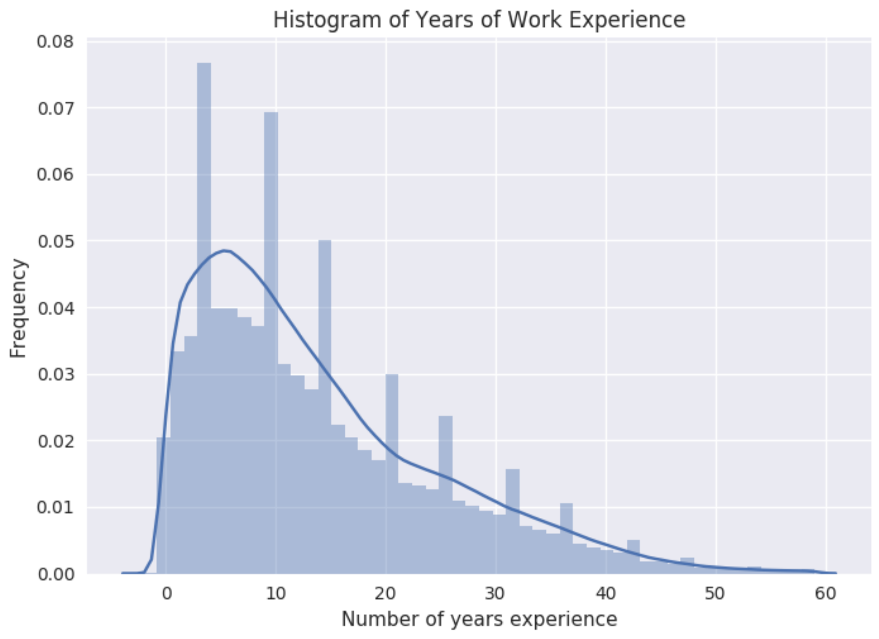

Figure 13: A histogram of the number of years of work experience. The number of years is truncated at 60 for display reasons - it is also likely that any number over this is an error in the CV parsing. The spikes in the graph occur as people tend to put a whole number of years against their time in a role.

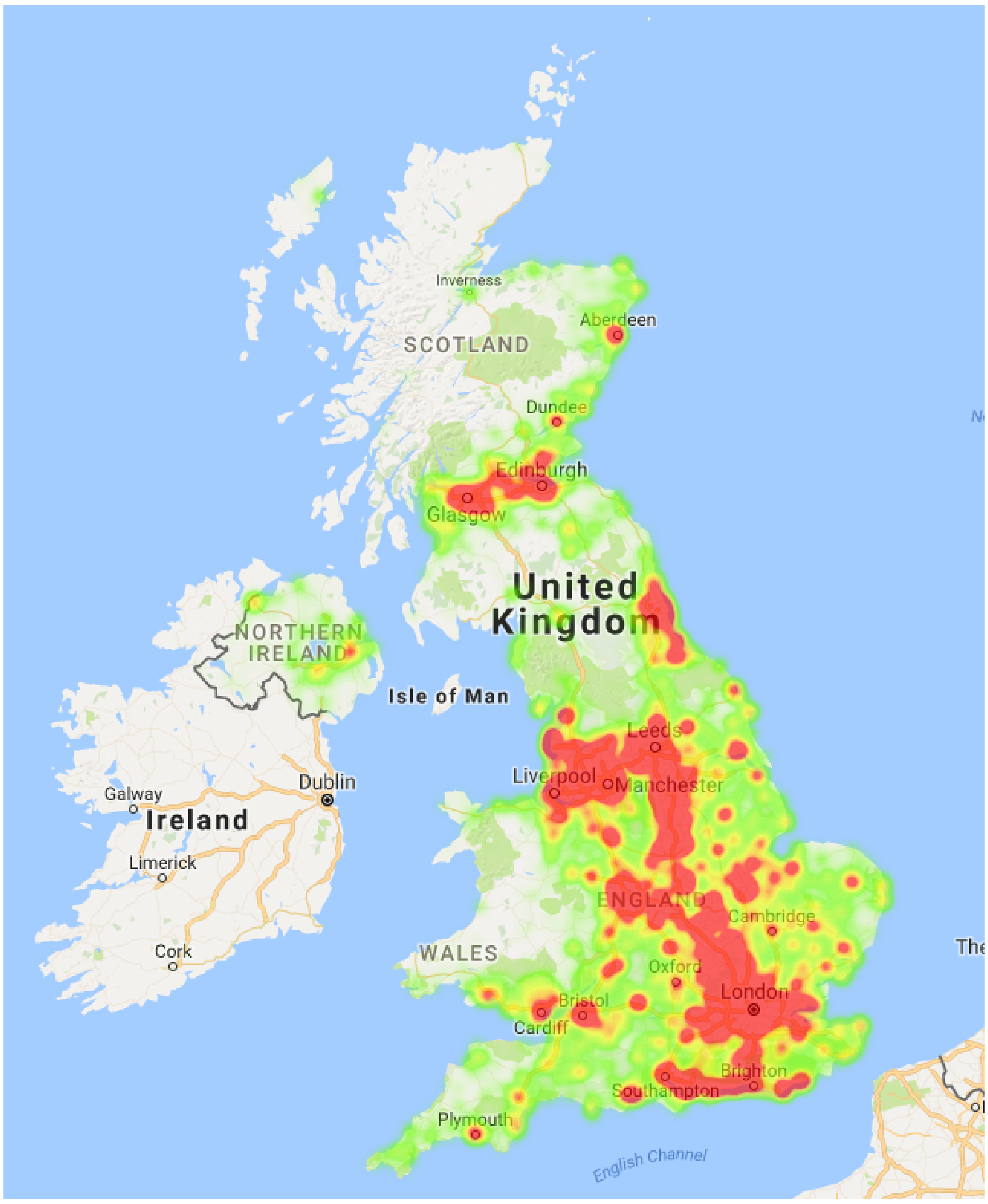

Figure 14: A heatmap of candidates' location. The redder the area the higher the candidate density. The map shows that the dataset achieves a good geographical spread around the UK, clustered around cities as would be expected.