Université de Montréal

Predictive Models for Career Progression

par

Zakaria Soliman

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des arts et des sciences
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Informatique

August 13, 2018

SOMMAIRE

Linkedin est le plus grand réseau social pour les professionnels où les utilisateurs du service partagent toute leur histoire professionnelle. Dans ce travail, nous explorons les méthodes par lesquelles nous pouvons modéliser la trajectoire de carrière d'un candidat donné et prédire les changements de carrière futurs. La première partie de cette thèse est une tentative de normaliser les données sur les titres d'emploi, car nous avons constaté que la façon dont les utilisateurs de la plate-forme de réseautage social professionnel décident d'y saisir leurs titres varie énormément. Ensuite, nous explorons divers modèles prédictifs inspirés des modèles de langage de forme, ainsi que des modèles neuronaux séquentiels.
Mots-clés: extraction de données, marquage de séquences, classification de séquences, réseaux de neurones, réseaux neuronaux récurrents, LSTM

SUMMARY

LinkedIn is the largest social network for professionals where users of the service share all of their professional history. In this work we explore methods by which we can model the career trajectory of a given candidate and predict future career moves. The first part of this thesis is an attempt to normalize the job titles data as we have found that there is a great deal of variation in how the users of the professional social networking platform decide to input their titles. Then we move on to exploring various predictive models inspired form language models as well as sequential neuronal models.
Keywords: data mining, naive bayes, sequence labeling, sequence classification, neural networks, recurrent neural networks, LSTM

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I'd like to thank Professor Philippe Langlais, my research advisor, for instilling some focus and discipline to my, at times, diffused way of working, and for guiding me through my first foray into computer science research. He enabled me to learn greatly from this experience and I am very grateful for this opportunity.

I'd also like to thank the amazingly skilled Fabrizio Gotti for his constant help throughout my work. He has been a source of inspiration whenever I met a hurdle.

Lastly, I'd like to thank Abbas Ghaddar for the endless discussions about the world of natural language processing we've had together. They have enabled me to learn a great amount about this highly interesting field in such a short time.

INTRODUCTION

This work was made possible in part with the help of Little Big Jobs, a human resources firm. They have provided the career data that allowed us to explore and attempt to learn new insights about professionals and their career paths. We believe this research will allow for more efficient and more personalized recruitment strategies and more desirable matching between talent and employers.

The growing popularity of social networks has given us the possibility to learn from data relating to various areas of a person's life. LinkedIn is the largest social network for professionals where users of the service share all of their professional history along with other relevant information such as their educational background and their skills set in the hopes of building an advantageous professional network. We have been provided a large collection of user profiles and we aim to extract some insights from that data and propose a solution for modeling a career trajectory. The main objective is to predict a set of plausible recommended positions that a professional can take as his next career move given his working history. Thus, the problem approached in this thesis, is a sequence modeling and prediction problem in which we try to learn the probability distribution of the next position or job title given a specific career history.

This thesis will be structured in a similar fashion as the way we approached the problem. In the next chapter, we begin by presenting the data set, explore its structure and look at the features we can use. Additionally, we review the statistical analysis that has been done in order to understand how to best use the provided data set. In chapter 2, we describe the models that were trained on the data along with the baselines that were compared to these models.

Chapter 1

SOCIAL MEDIA DATASET

1.1. Introduction

Building a predictive model to accurately describe career paths can present some challenging problems, related to the nature of the raw data, that we will describe in this chapter along with our proposed methods to solve them. The provided dataset was produced by scraping the data from the professional social networking platform LinkedIn. It is composed of a set of public and anonymous user profiles from the professional social networking site where we can find information mainly about their educational background, skill set, job history and their respective industry.

Tab. 1. I. Structure of a user profile

Field	**Sub-fields**	**Description**
_id		Unique identifier for a user
experiences		Array, where the first element is the most recent position held.
	job title	Name of the current position held by the user represented as a string.
	missions	Job description as entered by the user (not always available)
	place	The location of the company (not always available)
	companyName	The name of the company
countryCode		User's country
skills	name	Array containing objects that describe the skills name

1.1.1. Structure of the data

To provide an understanding of the dataset that was used throughout this work, we show how the raw data is structured in table 1. I. The fields that we are most interested in are the experiences array, where we find a list of the professional history of every user we have in our dataset, and the skills array, which contains the corresponding user's skill set which they input on the social media site. For brevity's sake, we only show the data fields that are relevant to us. Thus, we do not show the fields reserved for metadata. A full example of an actual user profile in JSON format is shown in appendix A, but to better understand how the data is structured, we show an example of the candidate's job history (table 1. II). The data fields that are of interest to us are the job title and the skills. For our purposes, we treat every user profile as a *sequence* of job experiences along with a paired list of skills.

TAB. 1. II. Example of a sequence of job experiences of a user. See the full JSON structure in annex A

Experience #	0	1	2	3
Job Title	Senior Vice President Consumer Marketing	VP Mobility Solutions	Director, Continuous Improvement	Associate
Company Name	TELUS	TELUS	Barrick Gold Corporation	McKinsey & Company
Mission	Support TELUS' continued growth in Consumer markets nationally across both wireless and wireline products and services	Support TELUS Consumer segment wireless overall business and marketing strategy, product marketing and pricing.	Led global Operations Continuous Improvement team aimed at driving hundreds of millions of dollars of operational improvement. Pioneer in importing Lean and Six Sigma philosophies and tools into the Gold Mining sector.	N/A
Place	toronto, ontario	N/A	N/A	N/A
Start Date	2014-05	2008-12	2000	1993
End Date	N/A	N/A	2003	2003

1.1.2. Initial Preprocessing of the Data

As is often the case when working with real world data, some initial cleaning must be done:

1. We remove all user profiles were the job title was not properly filled out (i.e. filled with `-`, `-`, `--`, `.`, `_`, and other sequences of non alpha numeric characters)
2. Setting all the textual data into lower case
3. Manually converting the most common abbreviations to their full length alternative. A list of these substitutions is shown in table 1. III. Before performing these substitutions, we start by removing periods (.) from the job title strings. This simplifies the regular expression rules that need to be applied to the set of strings. For example, we would substitute `c.e.o.` to `ceo`, `r.n.` (registered nurse) to `rn`, `sr.` to `sr`, and so on.

Tab. 1. III. Some Examples of String substitutions for common abbreviations and alternate titles

Target String(s)	**Substitution**
sr/jr	senior/junior
ceo	chief executive officer
cfo	chief financial office
cto	chief technology officer
cmo	chief marketing officer
coo	chief operations officer
vp/svp/evp	vice president/senior vice president/executive vice president
co-founder/cofounder	co founder

A complete list of the substitutions that have been applied to the set of strings along with the regular expressions are shown in appendix B.

1.2. Statistical Analysis of Data

After these initial steps, we inspected the dataset to understand how we may approach the problem and to gain some insights on how to build our models. A quick summary of some preliminary statistics about the data is shown in table 1. IV. In this section, we want to develop our intuition about the dataset and to get an understanding of how the data is distributed.

Looking at table 1. IV, the first problem we encounter is that we have a bit less than twice the number of unique job titles than we have user profiles, in other words, we have more job titles than we have user profiles. This suggests that we need a practical solution to

allow our models to learn something from the sequences of job experiences. However, from the table 1. IV, we notice that we have far less unique job titles used as the last position held by the candidates in our dataset. Thus, a good baseline is a predictor using only the last seen position in the sequence of jobs as the prediction target. Some other important metrics we should be aware of are that across all user profiles that were gathered for this dataset, we have at least 3 job experiences.

TAB. 1. IV. Some statistics describing the data

Total number of user profiles	2 789 111
Total number of unique job titles	3 859 835
Total number of unique job titles used as last job	927 209
Avg. length of job title strings (# of words)	4.55
Length of longest string (# of words)	42
Average length of job history	5.18 (positions held)
Shortest job history	3 (positions held)
Longest job history	140 (positions held)

1.2.1. Length of job history

Given that the main goal of this work is to model career paths, we are interested in understanding how the number of positions (held by the candidates throughout their career) is distributed. A bar plot is shown in figure 1.1 which illustrates the distribution of the lengths of the job experience sequences in our dataset. We immediately notice that there are several outliers, but for most of the user profiles, the average number of positions is 5.177 as shown in table 1. IV.

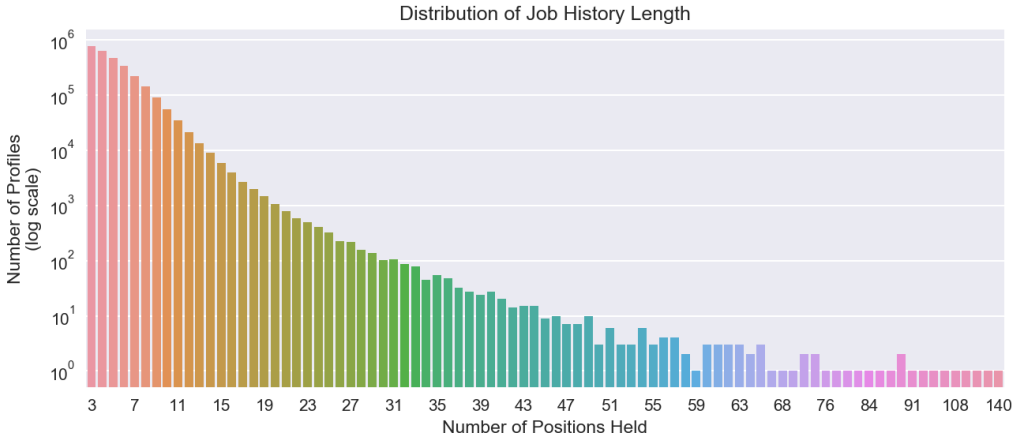


FIG. 1.1. Number of jobs held by candidates throughout their career

1.2.2. Distribution of Unique Job Title Strings and their Counts

As discussed at the beginning of section 1.2, our first major problem is that there are too many unique job title strings compared to the number of job experience sequences (profiles). Plotting the distribution of all the unique job title strings is difficult and not very readable so, in the x axis of figure 1.2, we used numerical ids instead of the actual job titles. We also show the distribution of the unique job title string **occurrences** in figure 1.3.

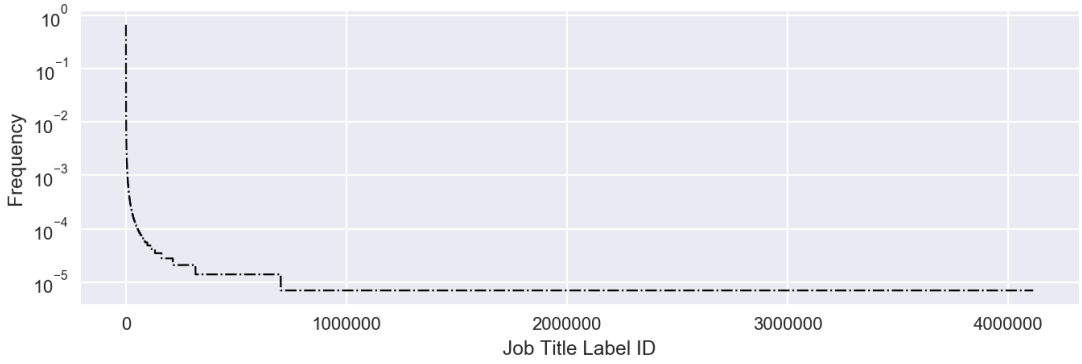


FIG. 1.2. Distribution of the job title strings in our dataset

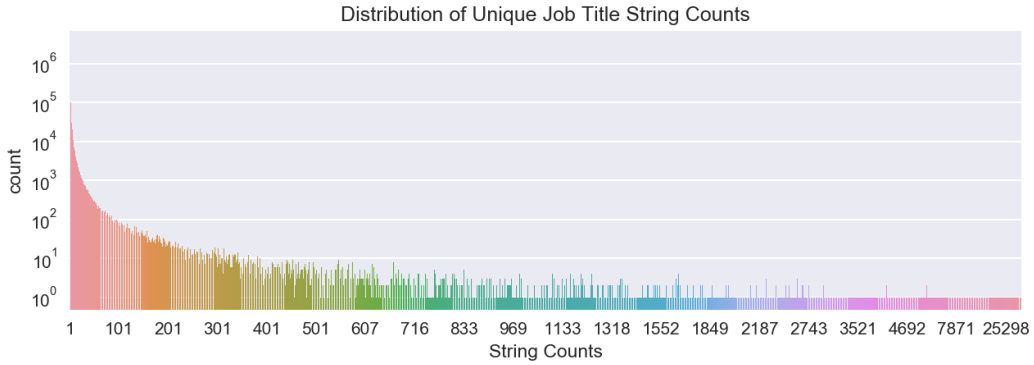


FIG. 1.3. Distribution of the occurrences of job title strings in our dataset

Figure 1.3 does a good job at showing the amount of variation we encounter when dealing with user provided job titles. We can see that most of the unique strings appear very rarely. In fact, 98.17% of all the job title strings appear less than 10 times in our entire dataset. The disparity between the average number of tokens and the maximum number of tokens found in job titles suggests that this might be the cause of the high diversity in our set of job titles. Looking at figure 1.4, we gain a bit of insight on what the root problem is. It seems that in the case of rarely seen job titles, users provided more specificity to their title thus increasing the number of words found in the string and making it less likely to coincide with the job title of another user.

FIG. 1.4. Random sampling of rarely seen job title strings. Colored text is the reduced job title we would like to have.

```
co instructor and teaching assistant, executive programs & undergraduate programs
specialist in organization and standardization of labor
college professor in human biology
accountant, sales and marketing department
senior manager, project management methodology & governance
hardware quality assurance engineer, mswam test engineering
sales associate/visual merchandising assistant
journalist and travel writer
director, strategic innovations & programs, digital & channels marketing
senior marketing analyst, demand planning
vice president for marketing department of university art group
project manager, key accounts - human health therapeutics
information technology manager - operations & architecture
national airfreight manager route development south america/perishable products
```

For some, like `specialist in organization and standardization of labor`, we can't reduce it or map it to a similar or an equivalent job title because it is so specific. However, if we look at `co instructor and teaching assistant, executive programs & undergraduate programs` we might want to reduce it to `teaching assistant` which appears 50 903 times. Likewise, we would reduce titles like `accountant, sales and marketing department` to `accountant` which appears 25 081 times.

Figure 1.5 confirms our intuition. We clearly see on the scatter plot that as we increase the length of the job title strings (in number of words/tokens) we find that the rarity of the job titles increases.

1.3. Approaches to Normalize the Dataset

In this section we present two different strategies to attempt to normalize the dataset with respect to the job title strings. First, to quickly understand how to model the problem and how these models would learn, a simple heuristic was used to filter out "bad" user profiles using a more or less rigorous criterion. Then, we dive deeper in the dataset to try and find a strategy that would allow us to keep a bigger subset of user profiles for training purposes.

1.3.1. Simple Heuristic

In an effort to minimize the ratio of number of unique job title strings to number of user profiles, we looked at what happens when we discard the least common job titles based on

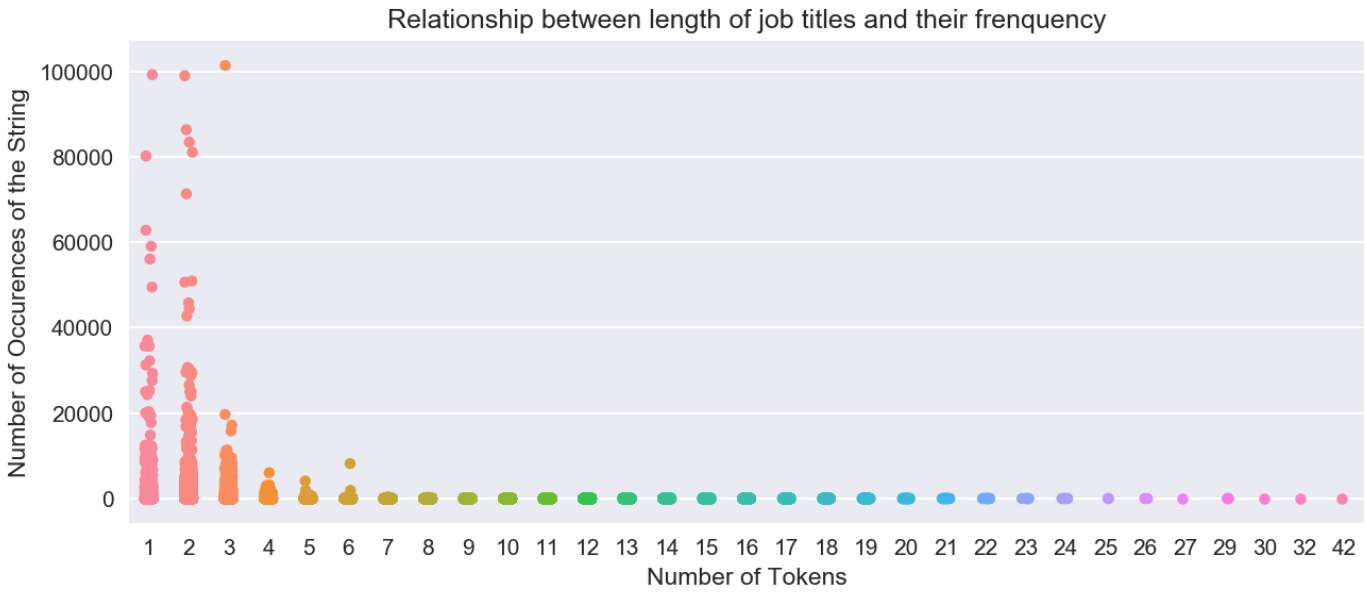


FIG. 1.5. Categorical scatter plot of the relationship between the number of words composing a title string and the number of occurrences of that string

some lower bound on the frequency of the string in the user profiles. For example, we may look at how many user profiles solely use the 100 most common job titles in their job history, we then look at how many users exclusively use the 101 most common job titles, and we measure gain from considering the top 100 to the top 101 job titles. We plotted this relation in figure 1.6. To understand what we mean by gain we define the list of job titles ordered by their frequency in the dataset as $\mathbf{j} = [j_1, \ldots, j_n]$ and we denote the set of user profiles that **only** use the k most common job titles in their profile as $N(\mathbf{j}_{1:k})$. We can then define the gain as:

$$\delta(k, k+1) = \frac{N(\mathbf{j}_{1:k+1}) - N(\mathbf{j}_{1:k})}{N(\mathbf{j}_{1:k})}$$

Looking at the plot, we see that after the 550 most common job titles, the gain becomes negligible compared to the amount of job titles that are under considerations ($N(\mathbf{j}_{1:k})$). So, a preliminary dataset is produced by keeping a list of the top 550 most common job title strings and discarding all the user profiles that have used at least one job title, in their job history, that is not in this list. Table 1. V shows the number of user profiles and the job titles we keep. We call the resulting dataset `550-titles`.

TAB. 1. V. Summary of the `550-titles` dataset

Total number of user profiles	120 371
Total number of unique job title strings	551

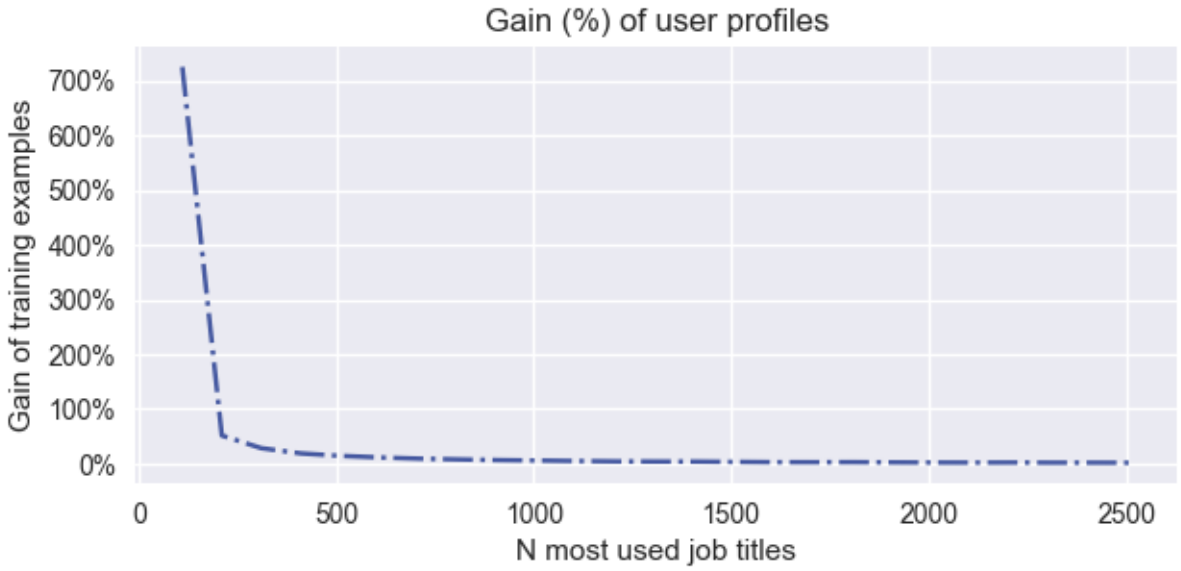


FIG. 1.6. Plot describing how many user profiles can be used by only considering the users that exclusively use the N most common job title strings (x-axis)

1.3.2. Removing Excess from Job Title Strings

Looking at figure 1.4 leads us to think that another heuristic we could use to reduce the number of unique job titles is to remove the portion of the string that comes after the first punctuation (like a comma) or any other non alphanumeric character we encounter. It seems that the candidates tend to naturally place the most important information of their job title at the beginning of the string and continue on adding more specificity. In order to extract the most relevant information out of a longer job title we split the label for each for each job title at the punctuations and conjunctions and we select the substring that occurs more frequently in the set of job titles as a whole. For example, if we have the job title `co instructor and teaching assistant, executive programs & undergraduate programs` we would split it into the following substring:

$$\left[\texttt{co instructor}, \texttt{teaching assistant}, \texttt{executive programs}, \texttt{undergraduate programs}\right]$$

After doing a look-up in a frequency table, we choose `teaching assistant` as the new label because it is the most common among this set of strings.

We can use the same heuristic to find the most relevant job titles strings that we might be interested in that we show in section 1.3.1. Doing so gives us the graph shown in figure 1.7. We can choose to only consider the 7000 most common job title strings thus leaving us with 837 910 user profiles that we can split into training set and a test set to evaluate our models. We will call this dataset `7000-titles`.

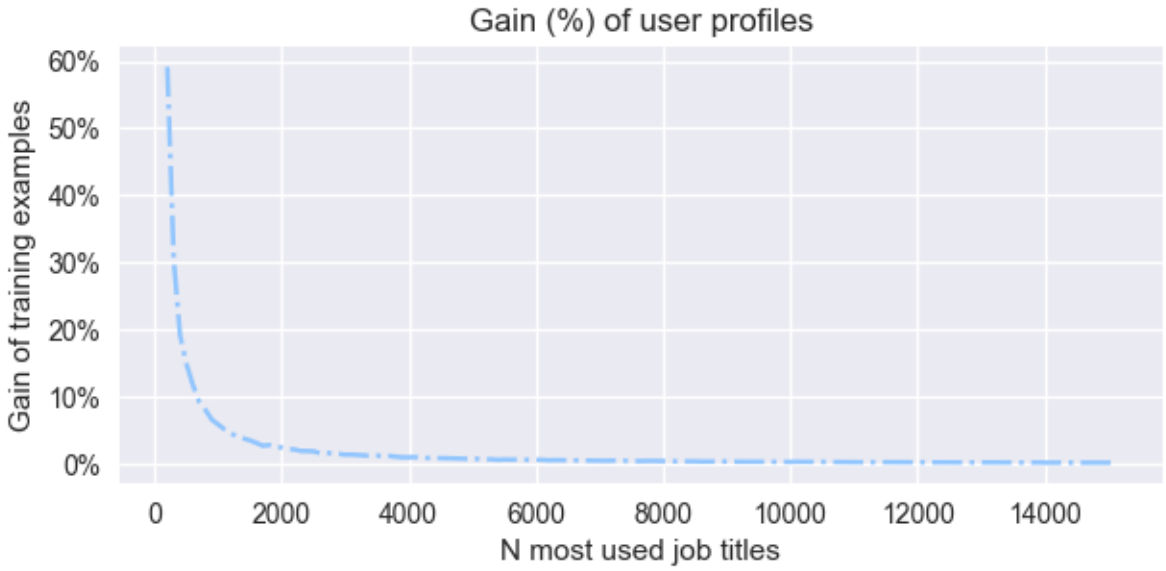


FIG. 1.7. Plot describing how many user profiles are taken into account by exclusively considering the ones that have the N most common job titles (on the x axis)

1.3.3. Classification Conflicts Between Sub-sequences

Upon further inspection of the data, done in an effort to understand the poor performance of our initial set of experiments, we notice that often, sequences in our dataset do not have the same outcomes. In other words, we can find user profiles that share the same exact job history but do not end up in the same position. This could lead to confusion during the learning phase of the models when estimating the optimal parameters. As an example, for the job title sequence `accountant -> assistant controller` we have 7 possible outcomes:

1. `president`
2. `manager`
3. `general manager`
4. `accountant`
5. `partner`
6. `controller`
7. `staff accountant`

For a given sequence of job titles, $j_1, j_2, \ldots, j_l$, we define the job history as $j_{1:l-1}$. So we say the job history is the given job title sequence with the last job title removed. The last job title is the one we want our models to predict (the prediction target). Then, to better understand the classification conflicts, we built a hash table where we use the job histories as keys and accumulate a list of the prediction targets (the last job title in each sequence). Doing this we learned that we have 47 149 unique job histories for the `550-titles` dataset. Recall that we have 96 486 training examples in this dataset. For the `7000-titles`, we have

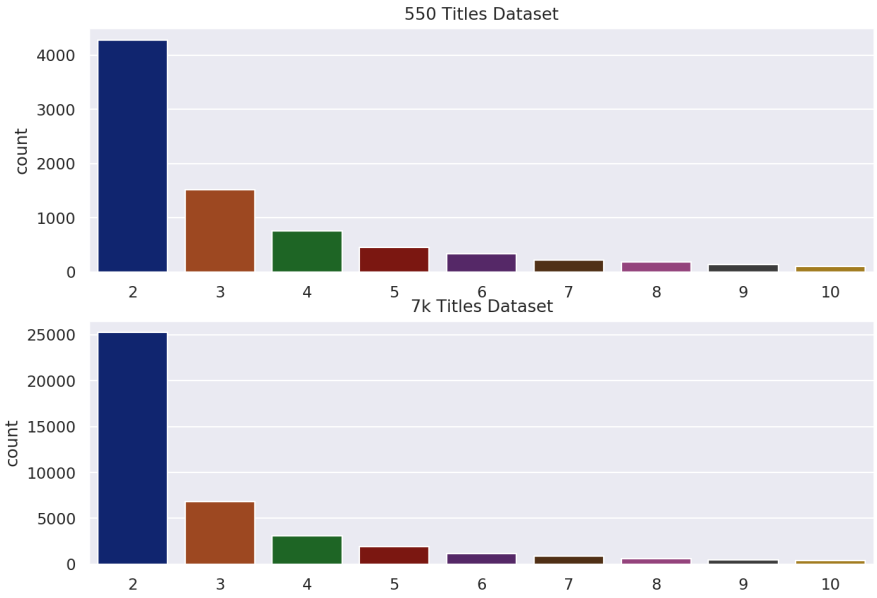


FIG. 1.8. Distribution of samples with respect to the branching factor when following a sequence of jobs in chronological order

497 025 unique job histories for a training set of 670 328 examples. Figure 1.8 shows the distribution over the branching factors of each job history. By branching factor we mean the number of different prediction targets for a given job history. So the figure shows the number of job histories that have branching factor b for every b. For the `550-titles` dataset, we have 8 847 job histories that share at least 2 different prediction targets and 43 251 for the `7000-titles` dataset.

Chapter 2

THEORETICAL BACKGROUND REVIEW

2.1. Introduction

This chapter will serve as a survey of the model families and techniques that have been used when implementing our experiments. We start by a brief introduction to the different methods in which we can represent text data. Later, we dive into the models by presenting the naive Bayes model to then move on to the sequential models that have been explored in this work.

2.2. Representation of Textual Data

2.2.1. Vector Space Model

The vector space model, introduced by [Salton, Wong, and Yang, 1975], allows us to represent text data as vectors. The idea was that word frequencies in a text could give us some semantic information about the text. These vectors are constructed such that the indices of the coordinates are associated to a unit such as a word or an N-gram. Thus, every document is represented as a point in a vector space where each dimension will have a non null value if the corresponding word or an N-gram is present in the document. These vector can then be used as feature vectors by a classifier for example.

2.2.1.1. *Bag-of-Words*

One of the most commonly used method to generate such feature vectors is by using the bag-of-words model where we represent each document as a *multiset* of words that is, in turn, represented as a binary or a co-occurrence vector by associating each component of the vector to a single word in our set of words. The basis for this approach in information retrieval tasks is that the word frequencies within a document might give us some information about what the document is about or the relevance of the document to a given query [Salton et al., 1975].
For example, if we look at these randomly selected examples from our LinkedIn data set, we

would start by first building a vocabulary based on the words that are observed in these texts.

1. `"analyst, information technology consultant, project manager, program manager, director, director, director"`
2. `"educator, project coordinator, associate project manager, project manager"`
3. `"sales associate, sales associate, financial services associate"`

Thus, these 3 documents would yield the following set of words:
$\mathcal{V} = \{$ `analyst`, `information`, `technology`, `consultant`, `project`, `manager`, `program`, `director`, `educator`, `coordinator`, `educator`, `associate`, `sales`, `financial`, `services` $\}$

We can then assign an index to each word:

$$
\begin{aligned}
0 &\rightarrow \texttt{analyst} \\
1 &\rightarrow \texttt{information} \\
2 &\rightarrow \texttt{technology} \\
3 &\rightarrow \texttt{consultant} \\
4 &\rightarrow \texttt{project} \\
5 &\rightarrow \texttt{manager} \\
6 &\rightarrow \texttt{program} \\
7 &\rightarrow \texttt{director} \\
8 &\rightarrow \texttt{educator} \\
9 &\rightarrow \texttt{coordinator} \\
10 &\rightarrow \texttt{educator} \\
11 &\rightarrow \texttt{associate} \\
12 &\rightarrow \texttt{sales} \\
13 &\rightarrow \texttt{financial} \\
14 &\rightarrow \texttt{services}
\end{aligned}
$$

Consequently, our constructed vocabulary induces a 14 dimensional vector space. We can then construct our vector representation of the corresponding document by either choosing to do so by using binary vector, where each component of the vector can either take a $0-1$ value indicating the occurrence of the associated word, or we could opt for a co-occurrence

vector, where the value of each dimension of the vector indicates the number of times the associated word occurs in the given document.

So, if we would like to represent the document #1 from the examples shown above by a vector $\mathbf{v}_1$, the non zero values would be:

$$\mathbf{v}_1[0] = 1, \mathbf{v}_1[1] = 1, \mathbf{v}_1[2] = 1, \mathbf{v}_1[3] = 1, \mathbf{v}_1[4] = 1, \mathbf{v}_1[5] = 2, \mathbf{v}_1[6] = 1, \mathbf{v}_1[7] = 3$$

In this case, $\mathbf{v}_1$ is a co-occurrence vector.

2.2.1.2. *Term Frequency and Inverse Document Frequency*

Now that we can represent our documents by vectors, we present a method that allows us to put more weights on occurrences of certain words that are deemed to be a good signal that the document is of a particular type as opposed to the bag-of-words method that simply counts the occurrence of each word. The resulting bag-of-words vector representation of the documents would yield less information that might allow us to discriminate between documents. The intuition is that globally rare words that are common across a particular subset of documents might be a stronger indicator of their similarity than a word that appears frequently in all the documents. Therefore allowing us to more reliably cluster similar documents together. Now, let us define some terms.

- **Term Frequency**
  The term frequency component of TF-IDF is related to the frequency with which a word or term t_j appears in a given document d_i. There exist several ways to compute this value, but one of the simplest ways is the following:

$$tf(t_j, d_i) = \frac{\text{Number of times } t_j \text{ appears in } d_i}{\text{Total number of terms (words) in document } d_i} \tag{2.2.1}$$

  We could also simply use the raw count of the term in the document as the value of $tf(t_j,d_i)$ such that we can rewrite the above expression like

$$tf(t_j,d_i) = \#\text{ of times } t_j\text{appears in } d_i = N(t_j,d_i) \tag{2.2.2}$$

- **Inverse Document Frequency**
  As discussed above, we need some way to evaluate how much a given word is relevant to the particular document we are interested in. To do this we take the logarithm of the ratio of number of document in our corpus to the number of documents containing the term t_j.

$$idf(t_j, D) = \frac{\text{Total number of documents in our corpus}}{\text{Number of documents where the term } t_j \text{ appears}} \tag{2.2.3}$$

Finally we compute the TF-IDF weight of a term by taking the product of both values.

$$tfidf(t_j, d_i, D) = tf(t_j, d_i) \times idf(t_j, D) \tag{2.2.4}$$

2.2.2. Distributional Models

In this section, we present a different approach to representing text as vectors based on the ***distributional hypothesis*** [Harris, 1954] which states that linguistic items with similar distributions have similar meanings. In other words, terms that have the same context or same surroundings tend to have similar meaning. With the advent of neural networks, [Bengio, Ducharme, Vincent, and Jauvin, 2003] have proposed a *neural network* based statistical language model which forms the basis of most of the recent advances in learning representations of words. In this work, we are mainly interested in the `fasttext` model [Bojanowski, Grave, Joulin, and Mikolov, 2016], [Joulin, Grave, Bojanowski, and Mikolov, 2016] which is an extension of `Word2Vec` [Mikolov, Chen, Corrado, and Dean, 2013]. Distributional representations of text allow us to use dense continuous vectors instead of very sparse vectors.

In figure 2.1, we show a diagram of the two architectures proposed by [Le and Mikolov, 2014]. Both architectures are fairly simple to understand. The main idea that is common to both is that we learn a projection matrix by trying to either reconstruct a word given a sliding window of its neighbouring words (in the case of the CBOW implementation) or, in the case of the skip gram model, by trying to reconstruct the context (i.e. the surrounding words) of a given word. The learned weights that form the projection matrix are then taken to be the words embeddings of the corpus they have been trained on.

The `fasttext` model extends the `Word2Vec` model by learning embedding vectors for the character N-grams that compose each word. The word vector is then the sum of its character N-gram vectors. Thus, yielding better embeddings for rare words since they often share character N-grams with other more common words, and vectors for out of vocabulary words can be constructed from their character N-grams.

2.3. Naive Bayes

The naive Bayes model is a very simple yet very effective linear classifier. This method of learning has been extensively used in text classification and information retrieval tasks. It leverages the Bayes' rule shown in 2.3.1 along with the *naive* assumption that the features of the observed data are independently distributed, which is a phenomena that seldom happens in real life.

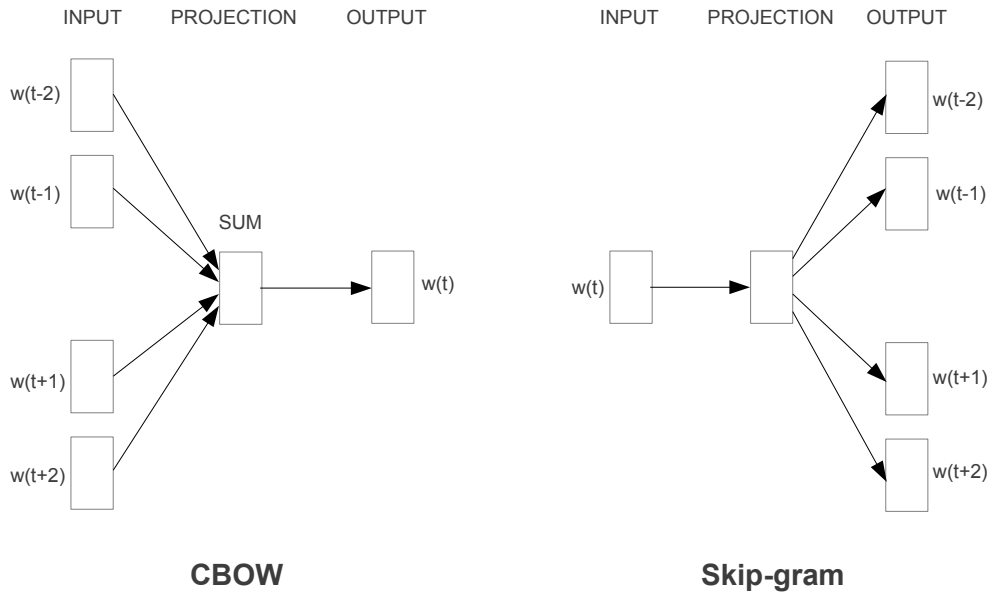


FIG. 2.1. Word2Vec model diagram taken from [Le and Mikolov, 2014]

In this section, we limit ourselves with a presentation of the theoretical framework of the models. The implementations and experimental details will be discussed in section 3.3.

$$p(c_j|x_1,\ldots,x_n) = \frac{p(x_1,\ldots,x_n|c_j)p(c_j)}{p(x_1,\ldots,x_n)} \tag{2.3.1}$$

Now, by the independence assumption on the features x_i's equation 2.3.1 can be rewritten as shown in 2.3.2 by repeatedly applying the chain rule.

$$p(c_j|x_1,\ldots,x_n) = \frac{p(c_j)\prod_{i=1}^{n} p(x_i|c_j)}{p(x_1,\ldots,x_n)} \tag{2.3.2}$$

Additionally, since the feature vector $\mathbf{x}^\top = [x_1,\ \ldots,\ x_n]$ is known, they are assumed to be observed, then the denominator term $p(x_1,\ldots,x_n)$ is a constant. Thus, the relation we are truly interested in is:

$$p(c_j|x_1,\ldots,x_n) \propto p(c_j)\prod_{i=1}^{n} p(x_i|c_j) \tag{2.3.3}$$

Where we often use the frequency of the classes observed in our training data set as the class priors $p(c_j)$ or we could assume that all the classes are equally probable, in which case, $p(c_j) = \frac{1}{\|\mathcal{C}\|},\ \forall j \in \{1, , \ldots,\ \|\mathcal{C}\|\}$

Next, we must assume a distribution over the feature in order to learn the parameters of the distribution.

2.3.1. Bernoulli Model

In the Bernoulli model the given features are represented by binary valued variables (i.e $x_i \in \{0, 1\}$). Thus, we would represent our documents as binary vectors where every dimension indicates whether the word occurs or not in the text as explained in section 2.2.

$$p(x_1, \ldots, x_n | c_j) = \prod_{i=1}^{n} P(X_i = x_i | C = c_j)^{x_i} \Big(1 - P(X_i = x_i | C = c_j)\Big)^{1-x_i} \tag{2.3.4}$$

2.3.2. Multinomial Model

In the multinomial model, the feature vectors are counts of the words that occur in a given document instead of being simply binary valued vectors indicating the occurrence of a word.

$$p(x_1, \ldots, x_n | c_j) = \frac{\Big(\sum_i x_i\Big)!}{\prod_i x_i!} \prod_i p_{ji}^{x_i} \tag{2.3.5}$$

2.3.3. Maximum a Posteriori Estimate

Finally, once the model has been completely defined, our aim is to find the class that is the most likely to have produced the observed features (the words of a document). We decide on the class of our document by finding the *maximum a posteriori* estimate for the class variable, which is equivalent to choosing the mode of the distribution. We do this by computing a vector of the same dimension as the number of classes where every component is the conditional probability of the class given the observed features.

$$\hat{y} = \arg\max_j p(c_j) \prod_{i=1}^{n} p(x_i | c_j) \tag{2.3.6}$$

2.4. N-gram Models

Inspired by language models, we also explored the N-gram prediction model to learn to predict the last position held by a candidate. Since we are trying to learn the distribution of the transitions to the next position in a career path given the job history, N-gram models lend themselves particularly well to the task at hand. thus, we provide a brief overview of N-gram language models.

In general, language models are probabilistic models that compute a probability for a sequence of words (in our case that would be a sequence of job title labels). Say we have a sentence of words (or a sequence of labels) $w_1^t = w_1, \ldots, w_t$, a language model would be able to give us an estimation of the probability of observing that sequence $P(w_1^t)$. We can compute the probability by using the chain rule shown in equation 2.4.1.

$$P(w_1, \ldots, w_t) = P(w_1)P(w_2|w_1)P(w_3|w_1,w_2)\ldots P(w_t|w_1^{t-1}) \tag{2.4.1}$$

However, what is of interest to us for the task at hand is the conditional probability of a word or a label given the history. We would like to estimate the probability of the last label given the history preceding that word. So we are trying to estimate $P(w_t|w_1^{t-1})$ which would allow us to select the label that maximizes this probability as our prediction. To do this we compute the maximum likelihood estimate (MLE). Nevertheless, using the entire sequence as the history or context isn't necessarily a good idea; different people have differing career trajectories before landing in the same position. This is where we utilize N-grams that we use as an approximation of the historical context of an individual's career. An N-gram is simply a sequence of N labels. Thus, we approximate the history by using a window of the N-1 previous labels and we compute the probability of observing the N^{th} label given that window. We explain the details of how this idea is implemented in section 3.4.

2.4.1. Kneser-Ney Smoothing

Several smoothing techniques exist to deal with the case where we need to compute a probability for an N-gram that hasn't been seen in our training set. In this work we used the Kneser-Ney smoothing [Kneser and Ney, 1995]. The maximum likelihood estimate for $P(w_i|w_{i-N+1}^{i-1})$, the probability of observing the word w_i after the sequence of words w_{i-N+1}^{i-1}, is given in the equation below.

$$P(w_i|w_{i-N+1}^{i-1}) = \frac{C(w_{i-N+1}^{i})}{\sum_{w_i} C(w_{i-N+1}^{i})} \tag{2.4.2}$$

Where $C(w_{i-N+1}^{i})$ is the count of the N-gram word sequence w_{i-N+1}^{i} as it appears in our training set and we normalize over the number of N-grams that share the same $N-1$ first words. We pad the sequences of words with tags (`<SOS>` and `<EOS>`) to mark the start and the end of the sentence, we treat the `<SOS>` tags as the words w_{2-N} to w_0. However, if an N-gram has not been seen in the training set, the model assigns zero probability to these events. For this reason we also add smoothing. We adjust the MLE by using kneser-ney smoothing as presented by Kneser and Ney [1995], which is what is used in this work.

2.5. Recurrent Neural Networks

Recurrent neural networks are a family of sequential models capable of passing some information from a time step to another. In a similar fashion as HMMs, the state at a given time step depends only on the previous state and the current input being fed to the network. Unlike traditional multi-layered neural networks, RNNs can scale to arbitrarily long sequences by making use of the idea of parameter sharing.

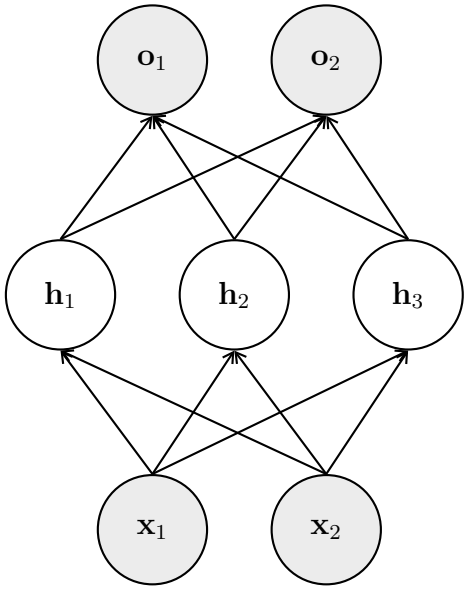


FIG. 2.2. Fully connected feed forward neural network with single 3 nodes hidden layer

2.5.1. Artificial Neuron

The main building blocks of artificial neural networks are mathematical functions commonly referred to as *artificial neurons* along with the directed edges that connect these neurons, which are normally represented by nodes when drawing the computational graph of the network. These artificial neurons can be thought of as a generalization of the perceptron which is limited to output a single binary value as opposed to a real valued output between 0 and 1. Let us define some objects.

Suppose we have an n-dimensional feature vector $\mathbf{x} = [x_1, \ldots, x_n] \in \mathbb{R}_n$. We denote by $\mathbf{h}_i$ the output value of the i^{th} neuron in a hidden layer of a standard feed forward neural network and the associated weights $\mathbf{w}^\top = [w_b, w_1, \ldots, w_n] \in \mathbb{R}^{n+1}$, note that there is an *offset* value $\mathbf{w}_0 = w_b$ that we call the ***bias***. We define the augmented vector $\tilde{\mathbf{x}}^\top = [1, \mathbf{x}] \in \mathbb{R}^{n+1}$ where we append a new value $\tilde{\mathbf{x}}_0 = 1$ that we call the ***constant feature*** to allow for easy computation using matrix operations. Thus a neuron would have $n + 1$ inputs. Equation 2.5.2 shows how we compute the i^{th} coordinate of the ***pre-activations*** vector which is the input to the activation function $f_{\mathbf{h}}$.

$$\mathbf{a}_i = \mathbf{w} \cdot \mathbf{x} = \sum_j w_j x_j \tag{2.5.1}$$

$$\mathbf{h}_i = f_{\mathbf{h}}(\mathbf{a}_i) \tag{2.5.2}$$

2.5.1.1. *Activation Functions*

Activation functions are the functions we apply on what we called the pre-activation vectors. We list here some of the activation functions that are used in this work, we omit the presentation of the *sigmoid* and the *softmax* function as we present them with more depth when discussing the output layer of the neural networks in section 2.6.4.

A popular activation function is the rectified linear unit (ReLU), we show below how it is computed.

$$relu(\mathbf{x}_i) = max(\mathbf{x}_i, 0) \tag{2.5.3}$$

We also use the hyperbolic tangent, shown in 2.5.4, mainly inside the recurrent network's LSTM cells.

$$tanh(\mathbf{x}_i) = \frac{e^{\mathbf{x}_i} - e^{-\mathbf{x}_i}}{e^{\mathbf{x}_i} + e^{-\mathbf{x}_i}} \tag{2.5.4}$$

2.5.2. Introduction to Recurrent Neural Networks

The implicit assumption made in our previous presentation on feed forward neural networks is that they are acyclical. By relaxing this constraint to allow feedback loops, which are essentially connections between the hidden layer that spans through every time step of the network, we obtain a recurrent network. We show a standard RNN that maps every input to an output in figure 2.3. Every individual time step can be seen as a fully connected feed forward network as shown in 2.2 where $\mathbf{h}^{(t)}$ is the hidden layer, $\mathbf{x}^{(t)}$ the input layer and $\mathbf{o}^{(t)}$ the output layer. The parameters are shared throughout time steps allowing the network to generalize over different sequence lengths.

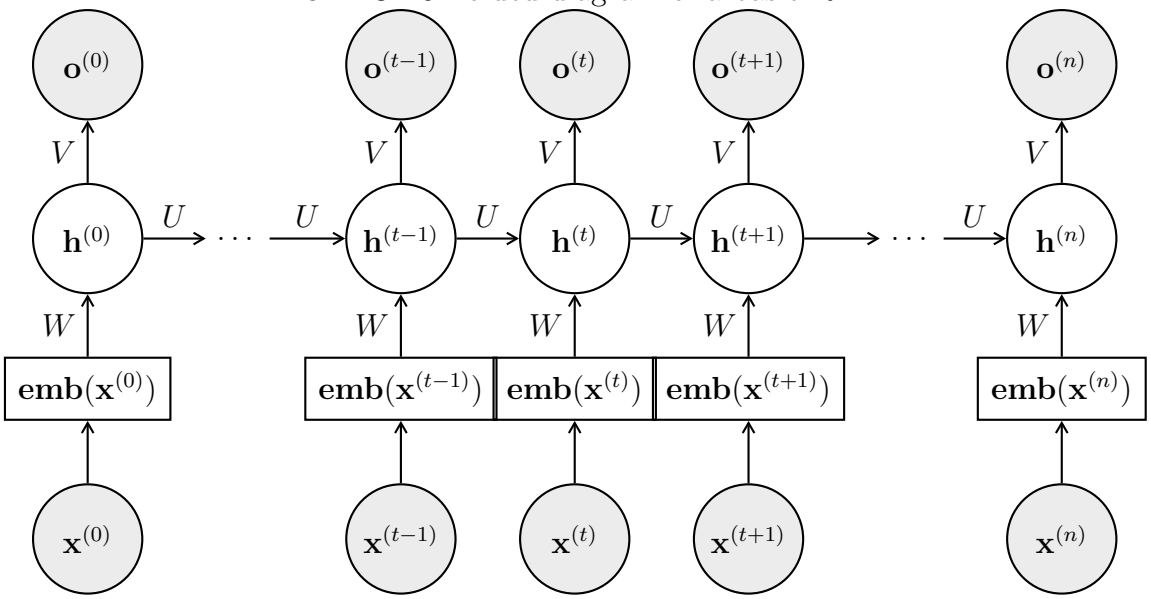
However, in practice, we encounter some challenges when trying to learn dependencies over long sequences. This is what is known as the ***vanishing gradient*** problem [Bengio, Simard, and Frasconi, 1994], [Hochreiter, Bengio, Frasconi, Schmidhuber, et al., 2001]. We sometimes might also have an ***exploding*** gradient, which is less likely, but can still hurt the optimization of the model's parameters.

2.5.3. Long Short-Term Memory

Long Short Term Memory introduced by [Hochreiter and Schmidhuber, 1997], commonly referred to as LSTM cells, is a sub network that we insert in place of the hidden layer of the standard RNN. We show a diagram of what is happening within the cell in figure 2.4.

As explained in the previous section, a recurrent neural network can simply be viewed as a series of identical (having the exact same parameters) feed forward network where each network passes its state to the neighbouring network. The LSTM cell is made up of gates that modify the state in some way before passing it along with the the value of its hidden

FIG. 2.3. Unfolded diagram of a basic RNN



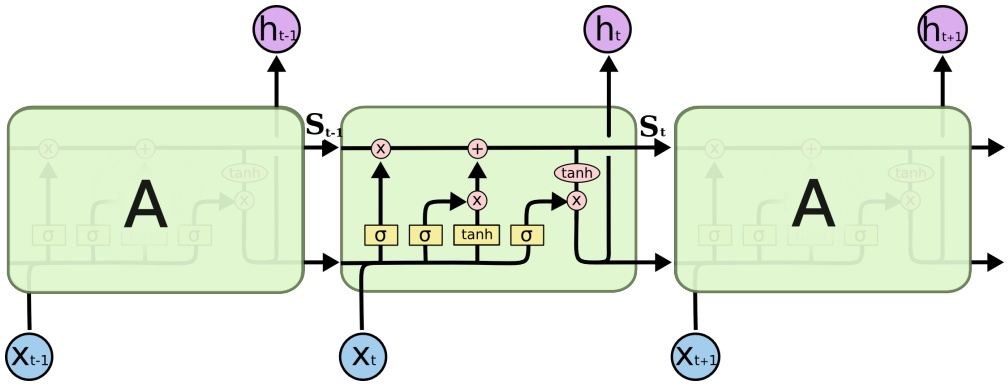


FIG. 2.4. LSTM cell diagram adapted from [Olah, 2015]

layer to the next network. The principal component we are interested in is the state vector $\mathbf{s}$ represented in figure 2.4 by the uppermost arrow going through the sequences of cells. We perform various operations on that state vector as well as the hidden layer vector $\mathbf{h}_t$.

2.5.3.1. *Forget Gate*

An LSTM cell has the ability to choose what information, from the previous states, can go through to the next network by leveraging what we define as the ***forget gate*** which is simply an element-wise multiplication of the output of the first sigmoid function, shown in figure 2.4, applied to a linear transformation on the concatenation of the previous cell's hidden layer value and the current input. The function is shown in equation 2.5.5. The output of the sigmoid function is a vector where each element has a value between 0 and 1. Thus, we can get an intuitive understanding of the first element-wise multiplication by the

forget gate as choosing what proportion of each component of the signal we want to keep and the proportion of the signal we want to remove.

$$\mathbf{f}_t = \sigma(W_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_f) \tag{2.5.5}$$

2.5.3.2. *Input Gate*

The ***input gate***, the second sigmoid layer from the left, is the gate that decides what information we will add from the new state candidate $\hat{\mathbf{s}}_t$, which is the output of the *tanh* function. The equations are shown in 2.5.6 and 2.5.7.

$$\mathbf{i}_t = \sigma(W_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_i) \tag{2.5.6}$$

$$\hat{\mathbf{s}}_t = \tanh(W_s \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_s) \tag{2.5.7}$$

After having computed the values of the input gate and the forget gate vectors, we get the new state vector to be passed to the next network (or fed back in the recurrent feedback loop) as shown in equation 2.5.8.

$$\mathbf{s}_t = \mathbf{f}_t \times \mathbf{s}_{t-1} + \mathbf{i}_t \times \hat{\mathbf{s}}_t \tag{2.5.8}$$

2.5.3.3. *Output Gate*

Finally, we compute the output of the current time step, $\mathbf{h}_t$, which is the output that we pass to either a sigmoid or a softmax layer to compute the network's prediction for this time step. The choice of the final non linearity depends on whether we are training the model for a multiclass problem or a multilabel problem (discussed later in section 2.6.3). This output is computed using the current cell state $\mathbf{s}_t$, which is itself computed form the previous cell states (the previously seen history). Thus, we apply the last gate, the ***output gate***, which decides which information form the cell state we will output, shown in equation 2.5.9. Again because of the sigmoid function, the components of the vector $\mathbf{o}$ are values between 0 and 1. So we can interpret equation 2.5.10 as weighting the components of the state, to which we apply a tanh non linearity, according to their importance for the output at the time.

$$\mathbf{o}_t = \sigma(W_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_o) \tag{2.5.9}$$

$$\mathbf{h}_t = \mathbf{o}_t \times \tanh(\mathbf{s}_t) \tag{2.5.10}$$

2.6. Convolutional Neural Networks

2.6.1. Convolution

For this work, a convolutional neural network was used as an encoder to learn abstract representation of each user's skill set. Modern CNNs were introduced by LeCun, Haffner, Bottou, and Bengio [1999], and they were mainly used for image recognition. However, CNNs have also been used for sentence classification [Kim, 2014], we base our encoder on these techniques. The easiest way to understand convolutions is by thinking of a sliding window which applies a function to a matrix. We call this sliding window the kernel. We show an example in figure 2.5.

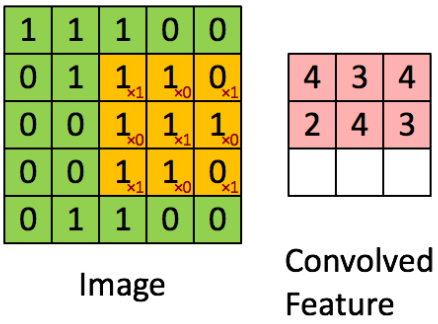


Fig. 2.5. Example of a 3 × 3 kernel applied to a 5 × 5 image. Taken from http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

In the example shown in figure 2.5, we apply a kernel K shown below in equation 2.6.1. We slide this kernel starting form the top left corner of the image left to right and form top to bottom. The $w_{i,j}$ are the models parameters which are learned during training.

$$K = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{2.6.1}$$

If we denote the input matrix by $I \in \mathbb{R}^{p \times q}$ and the kernel matrix by K we compute the output as shown in equation 2.6.2

$$O(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.6.2}$$

As we have discussed earlier, in a traditional feed forward neural network, every input neuron is connected to an output neuron on the layer above it. However, in the case of convolutional neural networks, the convolution kernel in used to compute the value of the output neuron. Thus, the neurons of the next layers are connected to the output of a region

(delimited by the kernel) of the input layer. We can have several convolutional layers, and the values of the kernels are learned through backpropagation.

For this work, we used convolution over word embeddings. In that case, the input matrix will be the matrix constructed by stacking the `fasttext` embedding vectors for each word of a document. For example, if we had 5 words, with 300 dimensional `fasttext` embeddings, we would stack them to get a 5×300 matrix, this matrix would be our input, over which we slide a convolution kernel. In computer vision applications, the kernel typically slides over patches of the image, but in our case, the kernel slides over the entire rows so we only slide it down along the height of the matrix. Figure 2.6 shows an example of that. As we see on the leftmost side of the figure, we stacked the word embeddings for skills that appear in our dataset (C++, java, python, etc). We perform a convolution with three different kernels of different sizes, these are the colored squares (red, green, and yellow). Notice that, as we said earlier, the kernels are always the same width as the input matrix, in this case the dimension of the embedding vectors. thus, we only need to slide these kernels from top to bottom and compute the value of the neuron every time we slide it downwards.

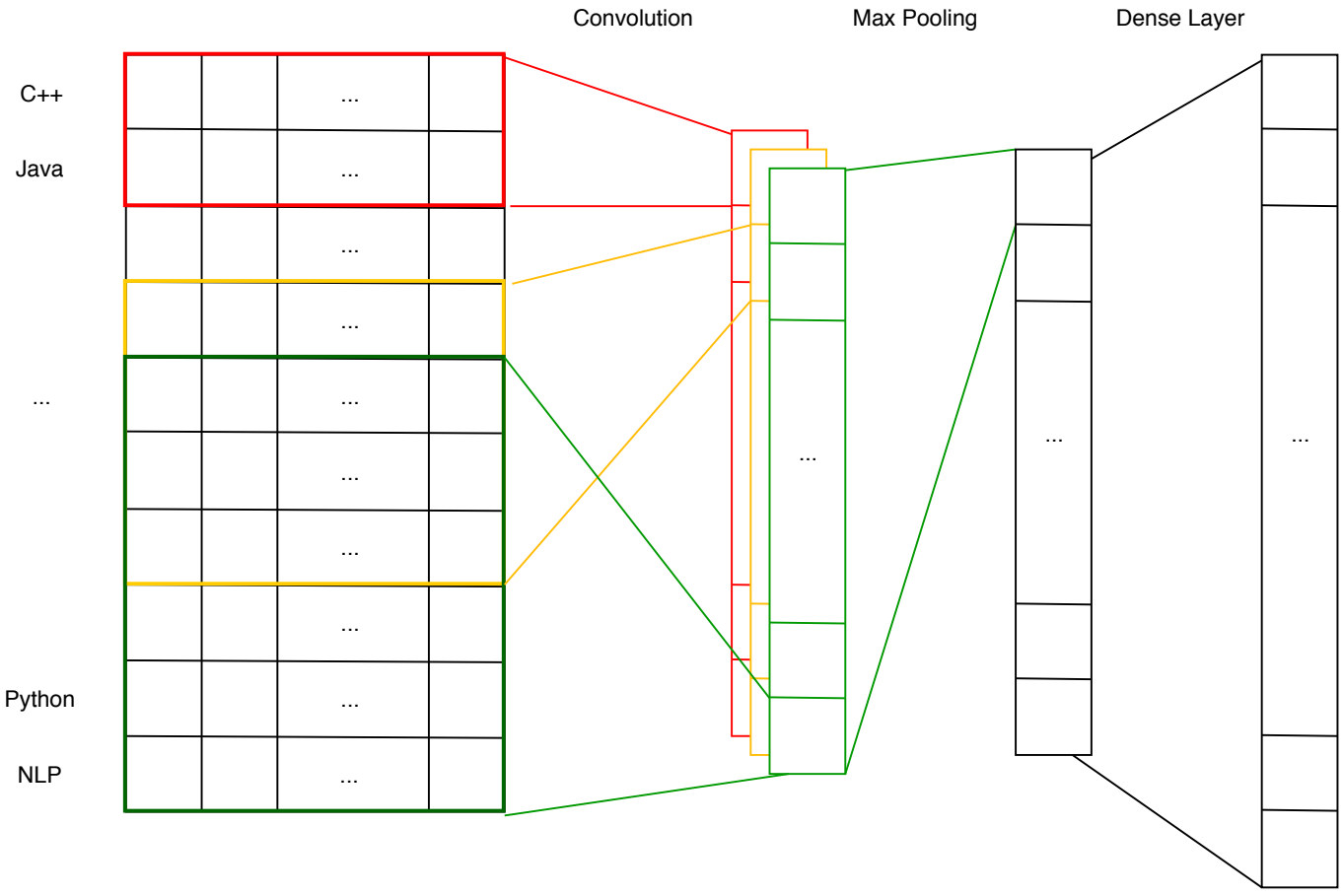


FIG. 2.6. CNN encoder that learns an abstract representation of the skill set of a user.

2.6.2. Max Pooling

After convolving the kernels with the input matrix, we get what are called ***feature maps***, shown in figure 2.6 as the second set of vectors from the left. They are colored with the color of the corresponding kernel that produces the output. Max pooling typically follows a convolution layer. Max pooling simply subsamples its input by getting the max value of the input. So in the case of the example given in figure 2.6, we apply the max pooling operation over the each feature map (the outputs of the convolutions with the kernels) and take the maximum value of each vector as the output to the next layer.

We apply this operation primarily to have a fixed sized output, no matter what the kernel dimensions are, max pooling always allows us to map the input to a fixed dimension for the output. It also reduces the size of the output all the while extracting the most important features of the N-grams. That is the intuition at the very least.

2.6.3. Multi-class v.s. Multi-label

For a *multi-class* classification problem we have one and only one class label associated with each example in the dataset, meaning that the class labels are mutually exclusive. On the other hand, modeling a problem as a *multi-label* classification problem allows for data samples to have multiple class labels; so a given data point could have multiple valid labels. More specifically, in our case, a given professional (job history and skill set) context could lead to multiple different possible future career moves. For the particular problem addressed in this work, we experiment with both modeling techniques. We ran a set of experiments by considering every job title as a class label, `senior software engineer` for example would be a class label. For a more fined-grained classification, we also approached the problem as a multi-label classification problem, thus making predictions at the word level as well as giving more flexibility to the predictions. Thus, for a given sample we could have different class labels: `senior`, `software`, `engineer`, `consultant`.

2.6.4. Output Layer

The decision of the activation function that we should use for the output layer for the neural network is directly related to the discussion above. We can choose between a softmax function (for a multi-class model) or a sigmoid function (for a multi-label classification model) we dive in the differences of these functions in the next sections.

2.6.4.1. *Softmax Function*

The softmax function *squashes* the values of a multi-dimensional vector such that every component of the vector falls in the range [0,1] and these values sum to 1. This enables us to interpret the resulting vector as a probability density over a discrete random variable.

0.1	0.7	0.05	0.049	0.1	0.001

FIG. 2.7. Example of an output vector

In our case, we interpret the resulting output vector as a density over all the possible class labels (job titles) and we pick the most probable (the index of the coordinate corresponding to the greatest value) as our prediction. We show the function in equation 2.6.3 applied on an N-dimensional vector. We show an example in figure 2.7.

$$\sigma(\mathbf{x}_i) = \frac{e^{\mathbf{x}_i}}{\sum_{n=1}^{N} e^{\mathbf{x}_n}} \tag{2.6.3}$$

To compute the value of the output vector in our network shown in figure 2.3 we apply the softmax function on a linear transformation of the hidden layer vector $\mathbf{h}_t$ which is the output of the LSTM cell as shown in equation 2.6.4.

$$\mathbf{o}^{(t)} = \sigma(V \cdot \mathbf{h}^{(t)} + b_o) \tag{2.6.4}$$

2.6.4.2. *Sigmoid Function*

If we choose for our model to allow for multiple labels for a given example from the dataset, we apply a sigmoid function on the output layer of the network. A sigmoid function outputs a real valued vector where each component is between 0 and 1, but in contrast with the softmax function, the components of the vector need not sum to one. We can therefore interpret the values as the probability of the label corresponding to the index of the vectors element. We show the function in equation 2.6.5. Notice that we don't normalize over all the input vector components as opposed to the softmax function in 2.6.3. In this case, we decide on a threshold value (normally 0.5) and pick all the indices corresponding to the coordinate for which the value is greater or equal to the threshold.

$$sigmoid(\mathbf{x}) = \frac{e^{\mathbf{x}}}{e^{\mathbf{x}} + 1} \tag{2.6.5}$$

2.6.5. Cross Entropy

We use the *cross entropy* loss to train the neural models that have been presented. We show below the cross entropy between two probability distributions. Recall that the output layer of the network provides us with a vector which we can interpret as an estimated distribution over class labels (in the case of a multi-class model) or as sequence of independent binary classifications. So for our case, the two "distributions" on which we compute the cross entropy are the ground truth vector $\mathbf{y}$ (which is a one-hot encoding of the class label for a

multi-class classification or a binary vector indicating all the labels in the case of a multi-label classification) and the output vector (the density estimation made by the model) $\hat{\mathbf{y}} = \mathbf{o}$

$$\mathcal{H}(p,q) = -\sum_{\forall x} p(x) \log(q(x)) \tag{2.6.6}$$

In the case of a **multi-class** model (class labels are mutually exclusive) we use categorical cross entropy shown in equation 2.6.7

$$\mathcal{L} = -\sum_{i} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) = -\mathbf{y} \cdot \log(\hat{\mathbf{y}}) \tag{2.6.7}$$

For the **multi-label** approach, we use the binary cross-entropy shown in equation 2.6.8 where $\hat{\mathbf{y}}$ is the output of a sigmoid activation. Notice that in this case, the result is a vector of the same dimensions as $\mathbf{y}$ and $\hat{\mathbf{y}}$ where each component is the loss value between the ground truth and the predicted label scores/probabilities. In practice, we reduce the resulting vector to the mean loss of all the components.

$$\mathcal{L} = \mathbf{y} \times -\log(\hat{\mathbf{y}}) + (1 - \mathbf{y}) \times -\log(1 - \hat{\mathbf{y}}) \tag{2.6.8}$$

We use $\times$ to denote element-wise multiplication.

Chapter 3

EXPERIMENTS

3.1. Introduction

For all the presented experiments, the various data sets were split into a training set (comprising of 80% of the dataset) and a testing set (20%) of profile IDs. The first round of experiments were performed using the `550-titles` data produced by the method introduced in section 1.3.1. A second set of experiments were also executed on the `7000-titles` data set that was produced using the method presented in section 1.3.2.

3.2. Baselines

Two baselines are used as a benchmark to the probabilistic models that are compared in this work. First, a model that simply predicts the most common job title label called *MoPro* for **Mo**st **pro**bable. Second, a model that predicts the last job title label in the job history given as input. We call this model *PreLa* for **Pre**dict the **la**st seen job title.

3.3. Naive Bayes

Two variants of the naive Bayes classifier were trained where we approach the problem as a classification task where the model is given the job history as a set of feature representations. The goal is to predict the most probable or a set of most likely next career move(s) from our universe of job titles that have been observed in our dataset. Two different approaches to represent the job history, explained in the next section, were explored:

1. We treat every *job title* as an atomic unit. More precisely, we give a numerical ID to each unique job title, and represent the candidate's job history by a vector of job title co-occurrences (example shown in section 3.3.1.1) where for every dimension there is a value representing the number of occurrences of the associated job title in the candidate's job history.
2. We treat the entire job history as a document that we tokenize (into words) and then construct our co-occurrence vectors.

For both approaches, we exclude the last job experience and use it as our target class. The class labels are represented numerically as integers. We also experimented with and without stemming the words. The results are discussed in chapter 4.
We trained two variants of the naive Bayes model. One defines the likelihood $p(t_i|t_\tau)$ as a multinomial distribution (equation 3.3.4), which is useful if we encode the job history as a word (or a job title) count vector instead of a binary vector. And the other defines it as a Bernoulli distribution (equation 3.3.1), which is used if we encode the job history as a binary vector (i.e. we are only interested in the occurrence or not of a word (or a job title) without considering how many times it occurred).

3.3.1. Representation of the Data

We will illustrate how we represented the data set by using our example from chapter 1 presented in table 1. II along with several other candidates. The candidate's job history in chronological order is shown in table 3. I

Candidate A	associate -> director, continuous improvement -> vice president mobility solutions -> senior vice president consumer marketing
Candidate B	server -> server -> human resources intern -> human resources assistant
Candidate C	management consultant-> research analyst -> partner
Candidate D	bookkeeper -> accountant -> financial analyst -> finance manager
Candidate E	guest service agent -> travel consultant -> travel consultant

TAB. 3. I. Example of job sequences represented in chronological order for several user profiles from out dataset. The blue text is the prediction target and the green text is what we condition on.

3.3.1.1. *Method #1: Job History as Vectors of Job Title Occurrences*

In this case we treat every job title string as a unit and treat our set of job titles as our vocabulary. Keeping with our example, the ID to job title mapping is shown in table 3. II.

Thus, for this example, our feature vectors would have 16 components, and for each candidate, their job history would be represented by a vector. For example, the binary feature vector representing candidate A's job history would be:

$$\mathbf{hist}^\top = [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

Notice that in this case whether we represent A's job history as a binary vector of occurrences or a vector of word counts yields the same result. If we look at candidate B, we would get a binary vector that looks like:

$$\mathbf{hist}^\top = [0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

And if we want the vector representing the count of the job title occurrences:

$$\mathbf{hist}^\top = [0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

Titles	**ID**
associate	0
director, continuous improvement	1
vice president mobility solutions	2
senior vice president consumer marketing	3
server	4
human resources intern	5
human resources assistant	6
management consultant	7
research analyst	8
partner	9
bookkeeper	10
accountant	11
financial analyst	12
finance manager	13
guest service agent	14
travel consultant	15

TAB. 3. II. Vocabulary of our dataset when we consider whole job titles (as they appear in our dataset) as tokens

3.3.1.2. *Method #2: Job History as Bag-of-Words*

This is very similar to the first method, the only difference is we tokenize the job titles into words to build our vocabulary $\mathcal{V}$ as shown in table 3. III

So in this case, the feature vectors will have $|\mathcal{V}| = 30$ components since job titles composed of multiple words are broken up and the vector represents the occurrences of every word in that new vocabulary. And again, the feature vector for candidate A would be:

$$\mathbf{hist}^\top = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

Tokens	associate	director	continuous	improvement	vice	president
IDs	0	1	2	3	4	5
Tokens	mobility	solutions	senior	consumer	marketing	server
IDs	6	7	8	9	10	11
Token	human	resources	intern	assistant	management	consultant
IDs	12	13	14	15	16	17
Tokens	research	analyst	partner	bookkeeper	accountant	financial
IDs	18	19	20	21	22	23
Token	finance	manager	guest	service	agent	travel
IDs	24	25	26	27	28	29

TAB. 3. III. Vocabulary of our dataset after breaking up individual job titles into the words that compose them.

We might immediately notice how this method might be superior because these are bag-of-words vectors. For example, two candidates in the engineering field would both have the words engineer, junior, and senior appear in their past experiences if both end up vice presidents of engineering (a typical career path), the second method could learn a relation between these words and that outcome. However, that signal is lost in the first method if we take a software engineer and an electrical engineer, the similarities won't be captured in the feature vectors. We still choose to compare them because, even though naive Bayes doesn't take into account the order of the sequences, the recurrent neural network does. This allows us to compare the behaviour of the models with the same feature vectors.

3.3.2. Learning

Once we have built our vocabulary and we have constructed our feature vectors, we are ready to learn the parameters for the models and do some inference. We start by setting some notation to explain the algorithm.

Set of job histories (document) with its associated label	$\mathcal{H} = \{(\mathbf{hist}_1, c_{k_1}), \ldots, (\mathbf{hist}_m, c_{k_m})\}$
Set of labels (classes)	$\mathcal{C} = \{c_1, \ldots, c_l\} \subset \mathbb{N}$
Vocabulary/Set of job titles	$\mathcal{V} = \{w_1, \ldots, w_{\lvert\mathcal{V}\rvert}\}$

Where $|\mathcal{V}| = n$, and $\mathbf{hist}_i^\top = [x_{i,1}, \ldots, x_{i,t} \ldots, x_{i,n}]$ where the indices $t \in \{1, \ldots, n\}$ are the word/job title IDs. We have m documents.

3.3.2.1. *Bernoulli Model*

$$\begin{aligned} p(\mathbf{hist}_i|c_j) &= \prod_t p(x_{i,t}|c_j) \text{ (By the assumption of independence)} \\ &= \prod_t p(w_t|c_j)^{x_{i,t}}\Big(1 - p(w_t|c_j)\Big)^{1-x_{i,t}} \end{aligned} \tag{3.3.1}$$

For the Bernoulli model, the parameters that we need to learn are the class conditional probability for every word/job title, $\theta_{t,j} = p(w_t|c_j)$, and the class prior, $\theta_j = p(c_j)$. The expressions to compute both are given below. In this setup, $x_{i,t} \in \{0, 1\}$ indicates whether or not the t^{th} word/job title occurs in the i^{th} job history (document) vector.

Let us define the following indicator function

$$\mathbb{1}(i,j) = \begin{cases} 1 \text{ if } i^{th} \text{ history vector has label } c_j \\ 0 \quad \text{otherwise} \end{cases}$$

Then, we define the document frequency that a word t appears in a document (history vector) of class c_j as:

$$df_{t,j} = \sum_{i=1}^{m} \mathbb{1}(i,j)x_{i,t}$$

Finally, we can compute the class conditional probability. We also apply Laplace smoothing which is simply adding a constant value (k=1 in this case) to the numerator.

$$p(w_t|c_j) = \frac{1 + df_{t,j}}{2 + \sum_{i=1}^{m} \mathbb{1}(i,j)} \tag{3.3.2}$$

For the class prior, we simply count the frequency of the observed classes in our training data set.

$$p(c_j) = \frac{\sum_i \mathbb{1}(i,j)}{|\mathcal{H}|} \tag{3.3.3}$$

3.3.2.2. *Multinomial Model*

In this model, we assume that an experience sequence is a series of independent trials where we draw words (or full job titles) from the same vocabulary $\mathcal{V}$, and every component of $\mathbf{hist}_i$, $x_{i,t} \in \mathbb{N}$, is a count of the number of times the word w_t occurs in the i^{th} candidate's job history. Thus, the vectors $\mathbf{hist}_i$ are drawn from a multinomial distribution such that $\mathbf{hist}_i \sim Multi(\sum_{j=1}^{|\mathcal{V}|} x_{i,j}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{|\mathcal{C}|})$. Where,

$$\boldsymbol{\theta}_j = (\theta_{j,1}, \ldots, \theta_{j,|\mathcal{V}|}) \text{ s.t. } \theta_{j,t} = p(w_t|c_j)$$

$$
\begin{aligned}
p(\mathbf{hist}_i|c_j) &= \prod_t p(x_{i,t}|c_j) \\
&= \frac{\sum_{t=1}^{|\mathcal{V}|} x_{i,t}}{\prod_t (x_{i,t}!)} \prod_t \theta_{j,t}^{x_{i,t}}
\end{aligned}
\tag{3.3.4}
$$

Thus, we have $|\mathcal{C}| \times |\mathcal{V}|$ parameters to estimate ($\theta_{j,t}$, $\forall j,t$). We simply use a smoothed version of the maximum likelihood estimates of these parameters. First, we define another indicator function.

$$
\mathbb{1}(t,j) = \begin{cases} 1 \text{ if } w_t \text{ appears in a job history sequence with label } c_j \\ 0 \quad \text{otherwise} \end{cases}
$$

$$
\theta_{j,t} = \frac{\sum_{i=1}^{m} \mathbb{1}(t,j) x_{i,t} + 1}{\sum_i \sum_t \mathbb{1}(t,j) x_{i,t} + |\mathcal{V}|} \tag{3.3.5}
$$

Where, $\sum_{i=1}^{m} \mathbb{1}(t,j) x_{i,t}$ is the number of times word w_t appears in all job histories of label c_j, and $\sum_i \sum_t \mathbb{1}(t,j) x_{i,t}$ is the total number of word occurrences for a class label c_j.

3.3.3. Inference

Once we have learned our parameter estimates from our training data set, we can compute the class label that maximizes $p(c_j|\mathbf{hist}_i)$. To do this, for every job history sequence in our test data, we compute a $m \times |\mathcal{C}|$ matrix of probabilities where each row is a vector of probabilities over the class labels. Note that $|\mathcal{C}| = 550$ in the case of the `550-titles` data set and $|\mathcal{C}| = 7000$ in the case of the `7000-titles` data set. After that is done, we find the row index that has the maximal value as shown in 2.3.6.

3.4. N-gram Models

Inspired by figure 4.1 and as discussed in section 2.4, we also experimented with the N-gram model that predicts the most probable (based on the sequences appearing in the training set) next job title to be held by a candidate, given a *portion* of their previous history. The model is trained by memorizing the frequencies of the observed N-grams for $N \in 2, 3, 4, 5$ where N is a hyper-parameter that we can choose beforehand. In other words, we count the frequencies of every N-gram that appears in our training dataset.

To do this, for a chosen value of N, we counted the number of N-grams in each sequence individually (i.e. we won't count the bigram composed of the last element of a sequence and the first element of the next sequence in the case of $N = 2$ for example).

After the counting process is done, we test the model by iterating over the sequences of job titles in our test dataset, hiding the last job title of course. Then, for a given test sequence, we iterate over all the job title labels and compute the probability of seeing the

last $N-1$ job titles of the current sequence along with the job title of the current iteration as the N^{th} job title. Finally we pick the title that maximized the probability of the event as our prediction.

3.5. Neural Models

We experimented with two fundamentally different neural models, the first one is a traditional recurrent neural network that takes the sequence of job title as input and outputs the predicted next job. The second model is an encoder-decoder model. The encoder is a convolutional neural network (discussed in section 3.5.5) that learns an abstract representation of the skill set for a given profile. This abstract representation (a vector) is then given to the decoder, which is the same traditional RNN, as the initial state of that RNN instead of an zero vector initial state as is usually the case.

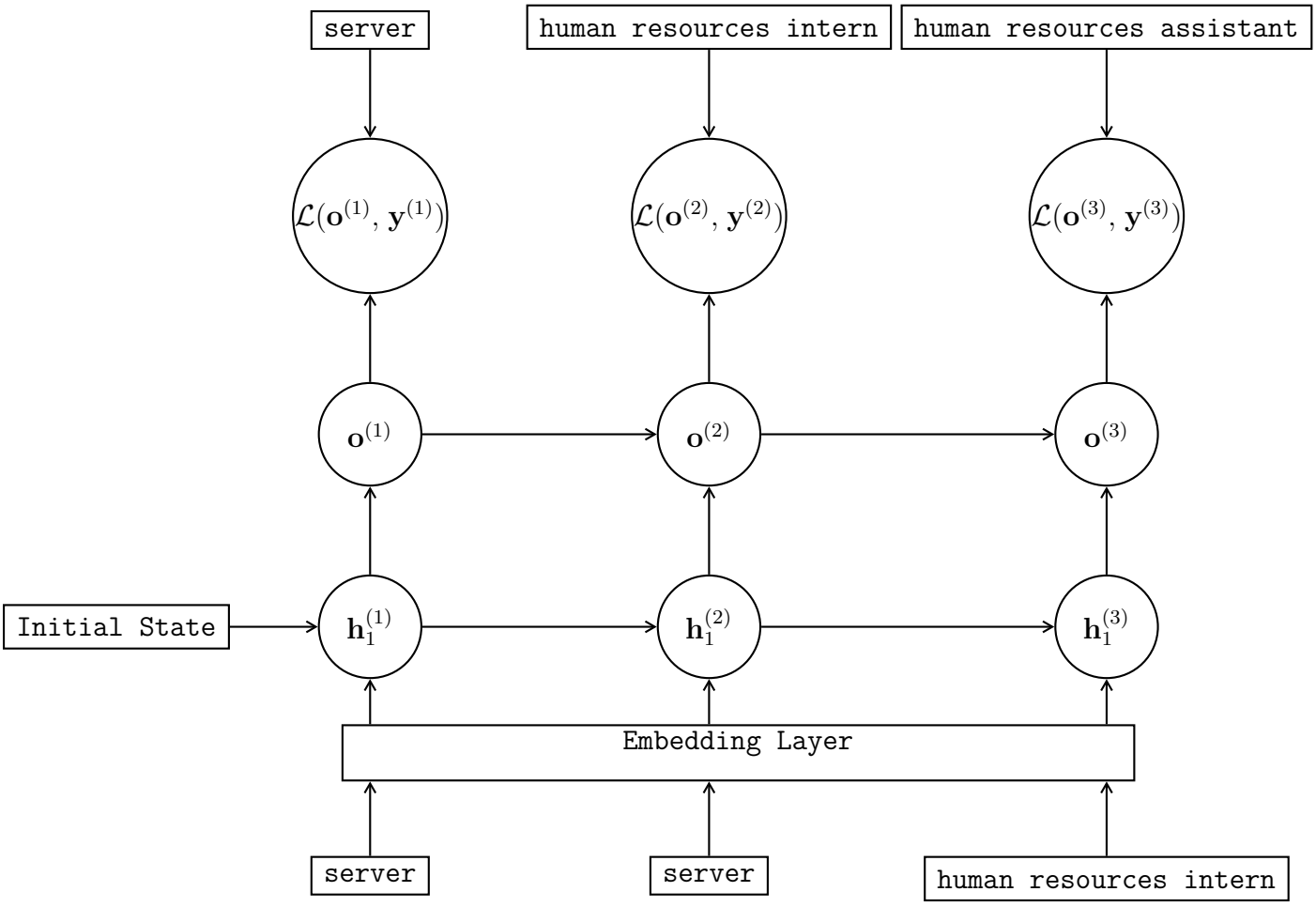


Fig. 3.1. RNN diagram showing example

3.5.1. RNN

The initial network architecture that was trained is similar to the one previously shown in figure 2.3 that we presented earlier in chapter 2. Figure 3.1 show an example of how we trained the model. This same network was also used as the decoder network for the second round of experimentation with neural models where we give the encoded representation of the skill set, provided by the CNN, as the initial state to the RNN for it to 'decode' it and use that information to make better predictions.

As we see in figure 3.1, we feed the RNN a sequence of job titles (server, server, human resources intern). The first step is to look up the embedding vector for the given job title at every time step. This is done by the embedding layer. The word embeddings for the job titles are tuned during training as well in order to minimize the loss $\mathcal{L}$. So the model takes the embedding vector for each job title and feeds it to the hidden layer LSTM cells ($\mathbf{h}_i^{(t)}$). We finally take the output and compute the loss $\mathcal{L}$ by comparing the true value to the predicted value. The gradient is then computed and we apply the backpropagation algorithm to update the parameters in order to minimize the loss. Note that we only use the embedding layer when we run the experiments where we want to represent the input job titles as embedding vectors. In the experiments where we used one-hot encoding or bag-of-words representation, the embedding layer is instead replaced by a lookup table where we have a one-hot or a bag-of-words vector associated to each job title.

3.5.2. Input Matrix for LSTM-RNN

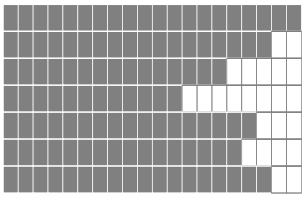


FIG. 3.2. Input data matrix. Each row is a vector representation of the sequence of job titles for a given user profile. Since the sequences aren't all of equal length, we add some padding, represented by the white cells.

For all of our recurrent models, we organize our input data as shown in figure 3.2 where the gray filled in boxes illustrate a representation of one element of the sequence (job title in our case) and each row is a candidates job history. They are obviously not of the same length, but we remedy that by padding the rest of the shorter sequences to match the length of the longest one and we keep a list of the actual lengths of each sequence. This gives our network the ability to output zero vectors as outputs and next states after having run through the entire sequence. Thus, the weights do not affect these outputs and aren't trained or affected by them.

Several representations of the sequence elements were attempted with this first model.

1. One-hot encoding of job titles
2. Each job title represented as a vector of word counts based on the words appearing in that job title.
3. A normalized average of word embeddings (equation 3.5.1) of the words appearing in a job title. The word embeddings are based on a pre-trained `fasttext` model

$$\mathbf{job}_{emb} = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathbf{v}_i}{\|\mathbf{v}_i\|} \tag{3.5.1}$$

Where we want to compute a vector for a job title of N words and $\mathbf{v}_i$ are the word embedding.

3.5.3. Multi-class

In the multiclass approach, we sequentially feed in the job titles (using various representations discussed later see section 3.5.2) and we compare the output at every time step with the ground truth $\mathbf{y}$ (which is simply the next job title) by computing the cross entropy loss function that we are trying to minimize. For all of our experiments, we used a projection layer to learn a richer representation of the input data. Embeddings from a pre-trained `fasttext` model were used for the projection layer and were adjusted during training. The model is shown in 3.1. For this model, the outputs $\mathbf{o}^{(t)}$ are $|\mathcal{J}|$ dimensional vectors where $\mathcal{J}$ is the set of job titles. Thus, $|\mathcal{J}|$ is either 7000 or 550 depending on the data set we are using. The vectors $\mathbf{o}^{(t)}$ can be interpreted as probabilities of each class label given the input sub-sequence (the inputs of time steps $1, \ldots, t-1$) at every time step.

3.5.4. Multi-label

In the multilabel approach, the only difference is that the output layer is a vector of dimension $\mathcal{V}$, which is the size of the vocabulary after having tokenized into words all the job title labels and removing stopwords. So, each word that can be part of a job title is now a label in and of itself. For this method, we apply a sigmoid non linearity as discussed in section 2.6.4.2

3.5.5. CNN Skill Set Encoder

As previously mentioned, after with the job history alone, we later incorporated the skill set of a candidate. However, not all user profiles have the same number of skills, so we have chosen to learn an abstract, high level, representation of the skill set of a user profile by using convolution over, pre-trained, `fasttext` embeddings of the words composing the skills.

Figure 3.3 illustrates how this was done, we stack the skill embedding vectors into a matrix as shown completely on the right. Then, several convolution layers are applied each with different kernel sizes. In the figure, we are applying 3 convolutions with kernel sizes 2

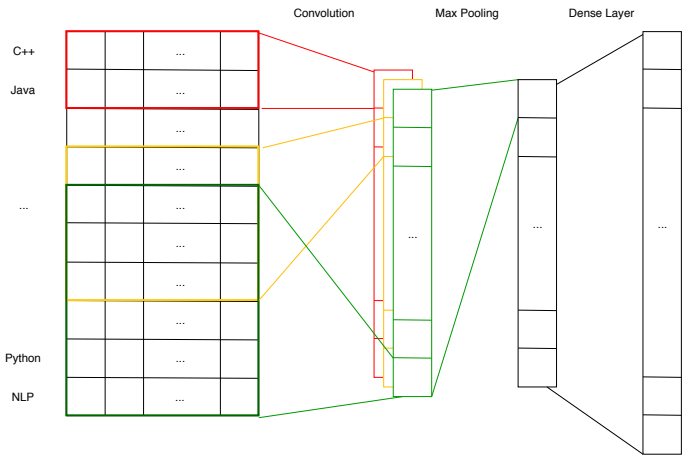


FIG. 3.3. CNN encoder that learns an abstract representation of the skill set of a user.

(red), 4 (yellow), and 6 (green). We slide these windows vertically over the skill embedding matrix for a given user profile. Each kernel convolution results in a vector on which we apply a max pooling over the entire resulting vector as for each component of the max pooling output vector (in essence constructing a new vector by taking the maximum value of each convolution output). The next step is to apply a dense layer with a relu activation function. The end result is what we interpret as the high level representation of the skill set which is fed to the recurrent neural network as the initial state at the very start of the sequence.

3.5.6. Evaluation

To help present the evaluation metrics, let us re-introduce some notation. For clarity and simplicity, we use a different notation than the one that was previously presented. Let $\mathcal{J}$ be the set of job title labels, $j \in \mathcal{J}$ a job title label, $\mathcal{V}$ the vocabulary or set of words after tokenizing the job title labels in $\mathcal{J}$ into word tokens, and $w \in \mathcal{V}$ a word from the vocabulary. We denote by N the number of data samples in the test set.

We define a job title label as the set of the words that compose it, $j_i = \{w_1^i, \ \ldots, \ w_n^i\}$ and we say that $\hat{j}_i = \{\hat{w}_1^i, \ \ldots, \ \hat{w}_n^i\}$ is the predicted label and $j_i = \{w_1^i, \ \ldots, \ w_n^i\}$ is the target (true) label for the i^{th} example from the test set.

To enable us to compare both the multi-class and multi-label approach, we opt for evaluation metrics computed at the word level. First we look at the exact label prediction *accuracy*, which means that we look at whether or not the model predicted exactly the same set of words as the ground truth. We will call this metric the *exact metric*, computed as shown in equation 3.5.2

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{\{\hat{j}_i = j_i\}} \tag{3.5.2}$$

This metric is essentially equivalent to computing the accuracy on the multi-class approach. It is equivalent to counting the proportion of test examples that have been correctly labeled.

Example 3.1. *Example of the exact metric*

Suppose that we have 3 user profiles for which we'd like to predict the next career move. Assume that we want the predict the following job titles for each profile:

Candidate 1: software engineer
Candidate 2: chief finance officer
Candidate 3: line cook

However, our model predicted the following job titles:

Candidate 1: software developer
Candidate 2: vice president finance
Candidate 3: cashier

So, using the notation defined above, we have:

The ground truth:

$j_1 = \{software,\ engineer\}$
$j_2 = \{chief,\ finance,\ officer\}$
$j_3 = \{line,\ cook\}$

And the predicted labels:

$\hat{j}_1 = \{software,\ developer\}$
$\hat{j}_2 = \{vice,\ president,\ finance\}$
$\hat{j}_3 = \{cashier\}$

$$\begin{aligned}\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}_{\{\hat{j}_i=j_i\}} &= \frac{1}{3}\sum_{i=1}^{3}\mathbb{1}_{\{\hat{j}_i=j_i\}} \\ &= \frac{1}{3}\left(\mathbb{1}_{\{\hat{j}_1=j_1\}} + \mathbb{1}_{\{\hat{j}_2=j_2\}} + \mathbb{1}_{\{\hat{j}_3=j_3\}}\right) \\ &= \frac{1}{3} * 0 = 0\end{aligned}$$

Second, we relax these constraints and we use a more fine grained approach by looking at the proportion of words that are commonly shared by the prediction and the ground truth. We call this the *strict metric* and the precision and recall are computed as shown in equations 3.5.4 and 3.5.3.

$$recall = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\ \hat{j}_i \cap j_i\ \right|}{\left|\ j_i\ \right|} \tag{3.5.3}$$

$$precision = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\ \hat{j}_i \cap j_i\ \right|}{\left|\ \hat{j}_i\ \right|} \tag{3.5.4}$$

For continuity, we stay with the same example. For brevity, let us only compute the recall.

Example 3.2. *Example of the strict metric*

Again, we have the ground truth:

$j_1 =$ *{software, engineer}*
$j_2 =$ *{chief, finance, officer}*
$j_3 =$ *{line, cook}*

And the predicted labels:

$\hat{j}_1 =$ *{software, developer}*
$\hat{j}_2 =$ *{vice, president, finance}*
$\hat{j}_3 =$ *{cashier}*

$$\begin{aligned}\frac{1}{N}\sum_{i=1}^{N}\frac{\left|\ \hat{j}_i \cap j_i\ \right|}{\left|\ j_i\ \right|} &= \frac{1}{3}\sum_{i=1}^{3}\frac{\left|\ \hat{j}_i \cap j_i\ \right|}{\left|\ j_i\ \right|} \\ &= \frac{1}{3}\left(\frac{\left|\ \{software\}\ \right|}{\left|\ \{software, engineer\}\ \right|} + \frac{\left|\ \{finance\}\ \right|}{\left|\ \{chief, finance, officer\}\ \right|} + \frac{\left|\ \emptyset\ \right|}{\left|\ \{line, cook\}\ \right|}\right) \\ &= \frac{1}{3} * (\frac{1}{2} + \frac{1}{3} + 0) = \frac{5}{18} = 0.27\end{aligned}$$

A third metric counts how many times at least one of the words in the predicted label is a word that appears in the target label for each test example. Let us define the following indicator function:

$$\mathbb{1}(\hat{j}_i, j_i) = \begin{cases} 1 & \text{if } \hat{j}_i \cap j_i \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

We can then write the equation for this metric as shown in equation 3.5.5. Later on, we will call this metric the *loose metric*.

$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{\hat{j}_i, j_i\}} \tag{3.5.5}$$

Example 3.3. *Example of the loose metric*

The ground truth:

j_1 = *{software, engineer}*
j_2 = *{chief, finance, officer}*
j_3 = *{line, cook}*

And the predicted labels:

$\hat{j}_1$ = *{software, developer}*
$\hat{j}_2$ = *{vice, president, finance}*
$\hat{j}_3$ = *{cashier}*

First, we see that the indicator function defined above gives us the following results:
$\mathbb{1}(\hat{j}_1, j_1) = 1$
$\mathbb{1}(\hat{j}_2, j_2) = 1$
$\mathbb{1}(\hat{j}_3, j_3) = 0$

thus, the metric is computed as follows:

$$\begin{aligned} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{\hat{j}_i, j_i\}} &= \frac{1}{3} \sum_{i=1}^{3} \mathbb{1}_{\{\hat{j}_i, j_i\}} \\ &= \frac{1}{3} * (1 + 1 + 0) = \frac{2}{3} = 0.66 \end{aligned}$$

We add another metric for the multi-label model. Since the model outputs a score for every word in the vocabulary, we can take the first five word labels with the highest score and compare them with the target label set. If we define the best 5 word labels predicted by the multi-label model as $\hat{j}_i^5$ we can define the following indicator function:

$$\mathbb{1}_{\{\hat{j}^5, j\}} = \begin{cases} 1 \text{ if } \hat{j}_i^5 \cap j_i \neq \emptyset \\ 0 \quad \text{otherwise} \end{cases}$$

The metric is computed as shown in equations 3.5.6 and 3.5.7 and we'll refer to this metric as the *best 5 metric*.

$$recall = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(j_i^5, j_i) \tag{3.5.6}$$

$$precision = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(\hat{j}_i^5, j_i) \tag{3.5.7}$$

This metric is exactly the same as the loose metric except for that $\hat{j}_i^5$ is the set of the 5 highest scoring labels instead of strictly the labels that have a score above a threshold (0.5 in our case).

In the case of the multi-class approach, we compute the accuracy of the models.

Chapter 4

RESULTS & DISCUSSION

4.1. Introduction

In this section, we present the results obtained form our models and provide an analysis of the results and explore various ways we can interpret them. We start by presenting an overview of the results that we obtained on each dataset. We start by looking at the prediction accuracy for the models we have trained to later select the best performing ones within each family of models to be compared in more depth. We will attempt to understand the results of the models and gain some insight about the task.

4.2. Result Presentation

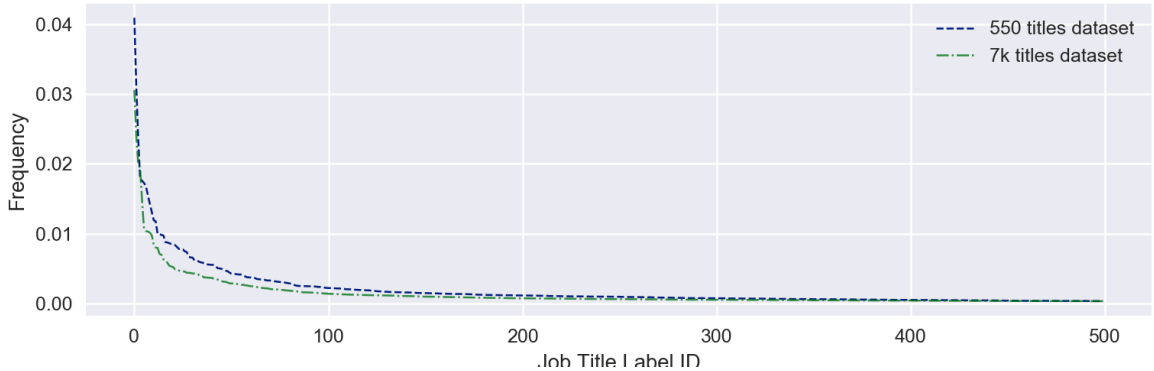


Fig. 4.1. Distribution of the job title strings we use as the prediction targets for our models. Normalized frequency of the job title strings both our datasets.

A quick overview of the performance of the models that have been trained on the `550-titles` and the `7000-titles` data sets can be provided by focusing on their accuracy to help us chose the best performing models on which we will do a more thorough analysis in order to understand the source of the prediction errors and to gain a better understanding of how the models learned.

We have discussed the distribution of job titles over the entire dataset in section 1.2.2. However, we would like to focus on the distribution of the last job titles (i.e. the ones we are interested in predicting). Figure 4.1 show the distribution of the target labels on both *training* data sets after the normalization attempts discussed in sections 1.3 we notice that the distribution has a very long tail, to this point, most of the job title labels occur infrequently.

4.2.1. N-gram Models

We start off by presenting the N-gram model results for the `550-titles` and the `7000-titles` data sets shown in figure 4.2. The figure shows a plot of the frequency of finding the correct target if we consider the K best predictions for different values of K (1 to 10 best predictions). Looking at the graph for the `550-titles` dataset, all the models seem to be performing more or less the same with the bigram and trigram models beating the rest albeit by a small margin. The bigram performs best when only considering the first prediction (top scoring one) but the trigram has a very slight edge when we allow for more than a single prediction. At first glance it might seem that conditioning on a larger window of the past history of job titles does not give us more predictive power, but looking at the results for the `7000-titles`, having a larger windows seems to be beneficial as the 5-gram and 4-gram models and trigram outperform the bigram model. This behavior can be explained by the fact that we see longer job history sequences in the `7000-titles` dataset than in the `550-titles` dataset. In fact, there are 120 122 job history sequences in the `7000-titles` training set that have a length greater than 5 jobs as opposed to the `550-titles` where we only find 5 646 such sequences because we have less examples as a whole. Since the N-gram models basically memorize the training dataset patterns, we find that this is a reasonable explanation.

4.2.2. Naive Bayes Models

Figure 4.3 shows the accuracy of the various naive Bayes models that we trained. Recall from section 3.3 that we have trained two variants of the naive Bayes model and we experimented by varying the representation of the job history that we give as input to the model the names we added to the legend in figure 4.3 reflect how the input data is represented. For instance `multi_nb_bow_no_stem` would be the multinomial naive bayes on the bag-of-words representation of the career history and no stemming was applied to the words.

Looking at the figure, the multinomial naive Bayes model trained on job title IDs (i.e. without tokenizing the job title sequences into words and representing the sequence as a standard bag-of-words) outperforms all other models on both datasets. As was discussed in section 3.3 we show the results with and without stemming the words composing the job titles as well.

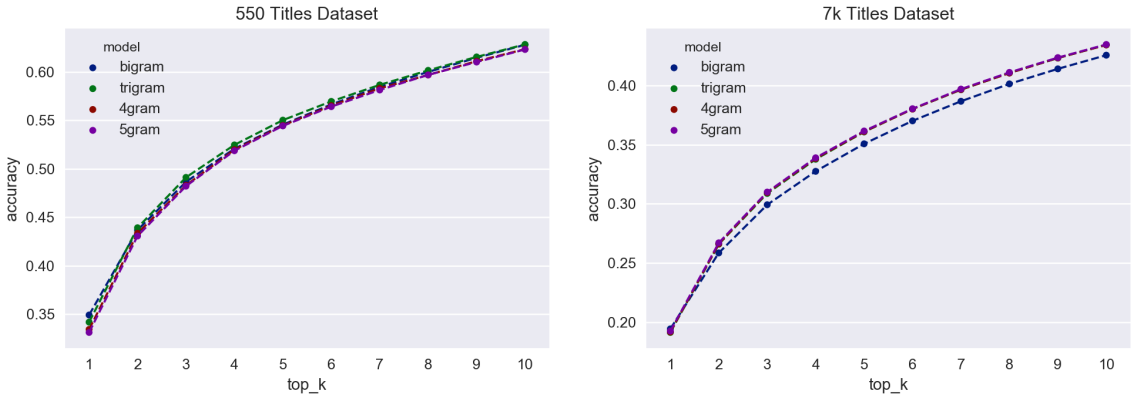


FIG. 4.2. Prediction accuracy of N-gram models by considering top k predictions. The longer N-gram seems to be more beneficial for the `7000-titles` than for the `550-titles` which is understandable seeing as we find a larger number of longer sequences in the `7000-titles` dataset

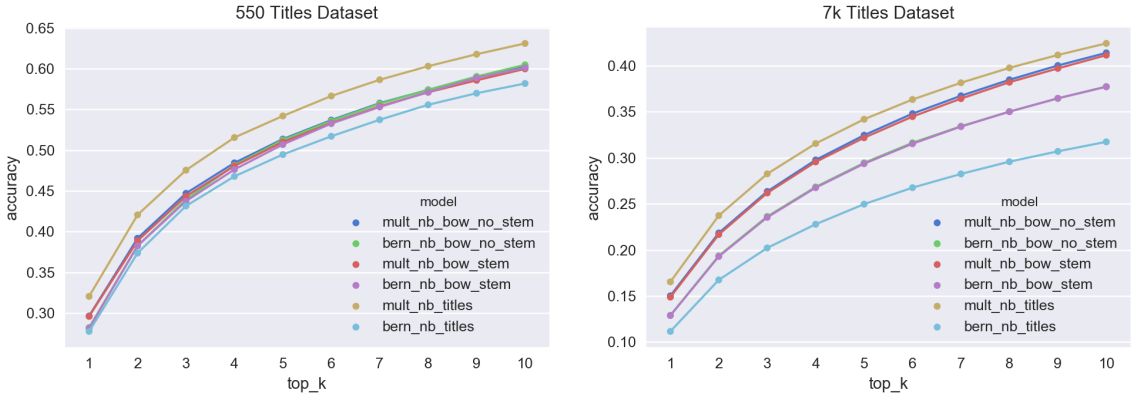


FIG. 4.3. Prediction accuracy of naive bayes models by considering top k predictions

4.2.3. Neuronal Models

Next, we show the results obtained for the LSTM recurrent neural networks using the multi-class approach (i.e. using a softmax function on the output layer) along with the model augmented with a CNN skill set encoder (`cnn-lstm`) in figure 4.4. As previously discussed in section 3.5.2, a standard recurrent LSTM network has been trained with three different methods to represent the input data, figure 4.4 shows the results obtained for each method of representation. As a naming convention, we use the data representation method discussed in section 3.5.2 for the LSTM-RNN and we call `cnn_lsmt` the CNN encoder and RNN decoder model discussed in section 3.5.5.

We learned that the data representation doesn't give us any significant advantage for this particular task. However, by representing the job titles as `fasttext` word embeddings, as explained in section 3.5.2, we get a slight edge in the performance on both datasets.

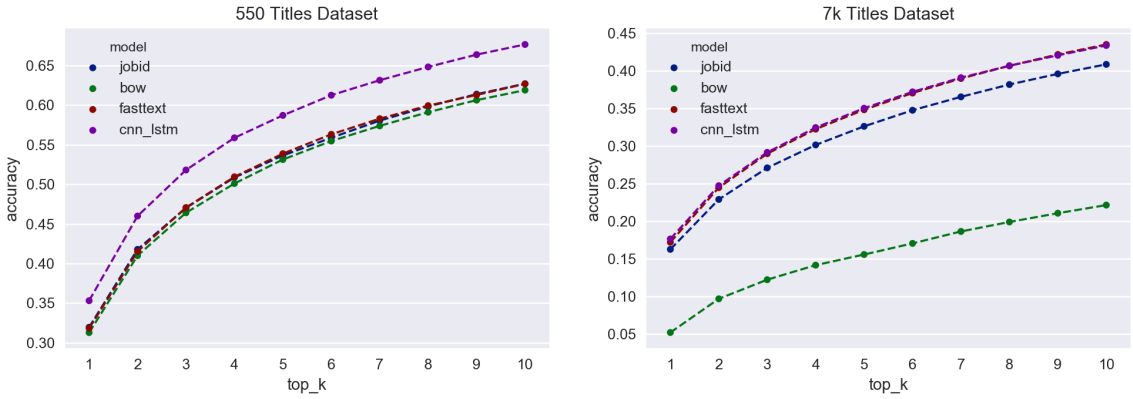


FIG. 4.4. Prediction accuracy for the LSTM RNN model

On both datasets, augmenting the network with a convolution neural network to learn a representation of the skill sets results in an increase in performance. Although, for the larger dataset (`7000-titles`), the performance is the same as without the encoder network, but with using `fasttext` word embeddings to represent the words composing the job titles.

4.2.4. Comparing the Models

Finally, we select the best performing models within each model family where we approach the task as a multi-class problem and we compare their top-K accuracy in figure 4.5. We also added the two baselines discussed in section 3.2; PreLa which predicts the last job title label in the input job history and MoPro which outputs the most frequent job title label in the training set.

The bigram models starts out with an equivalent performance as the encoder-decoder CNN-LSTM network using `fasttext` embeddings on the `550-titles`, but the CNN-LSTM network outperforms the other models as we begin to consider the accuracy at K. In the case of the larger `7000-titles` dataset, the CNN-LSTM network is a little behind the 5-gram model. However, as we increase K to compute the precision at K the neural model becomes more competitive. N-gram models perform strongly against the multinomial naive Bayes model on both datasets and the neuronal model. The PeLa baselines performs surprisingly well, but this is an artifact of the distribution of the datasets. We find that 34.15% of the job sequences have the same job label for the last two jobs in the `550-titles` and 18.87% of them for the `7000-titles`. Figure 4.6 shows how candidates change job positions less frequently the more advanced in their career they are.

4.2.4.1. *Accuracy by Length of Job History*

In figure 4.7, we look at the accuracy by length of the input sequence (length of the given job history we are conditioning on to make a prediction). The figure is overlaid with

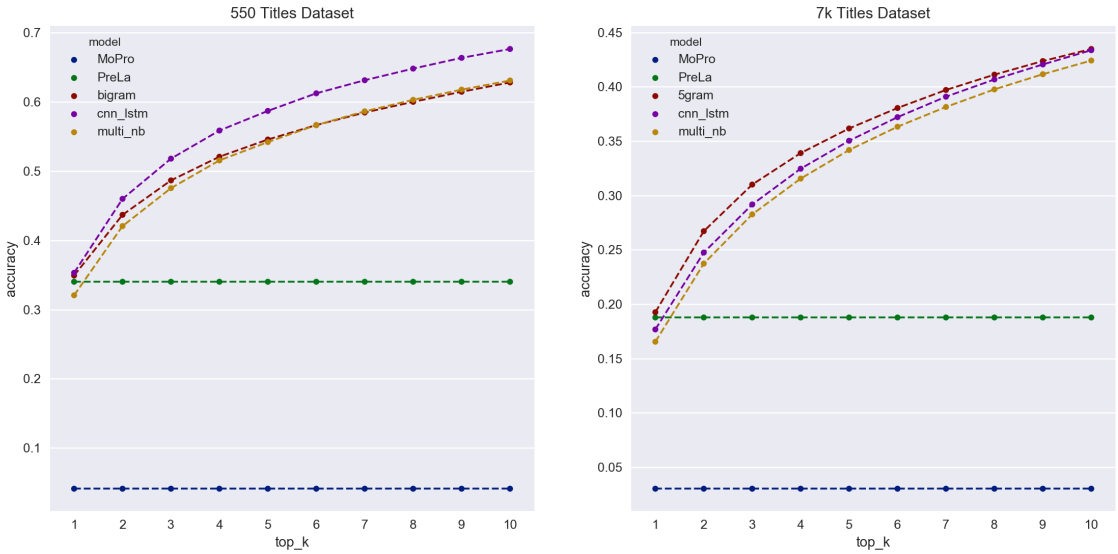


FIG. 4.5. Prediction accuracy for the LSTM RNN model

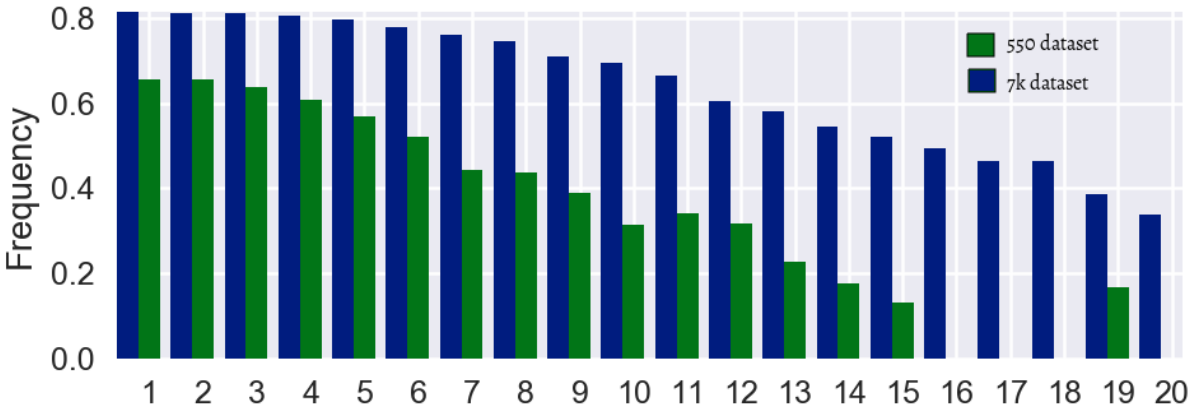


FIG. 4.6. The frequency that a candidate will change job position when at the i^{th} position in his career to the next job ($(i+1)^{th}$ position) as observed on the training dataset.

a histogram showing the number of data samples (from the test set) by length of the job history. There is an upward trend indicating at first glance that for longer sequences, the models perform better. There are multiple factors that could cause this effect.

First, by looking at the distribution of the sequence lengths in the test dataset overlaid in figure 4.7, we could argue that because there aren't enough long sequences in the test dataset to give us a statistically accurate estimate of real world performance of these models over long sequences. However, if we look at figure 4.6, we show the frequency of change, when a candidate moves from a job at a time t to his next job at time $t+1$ in the *training*

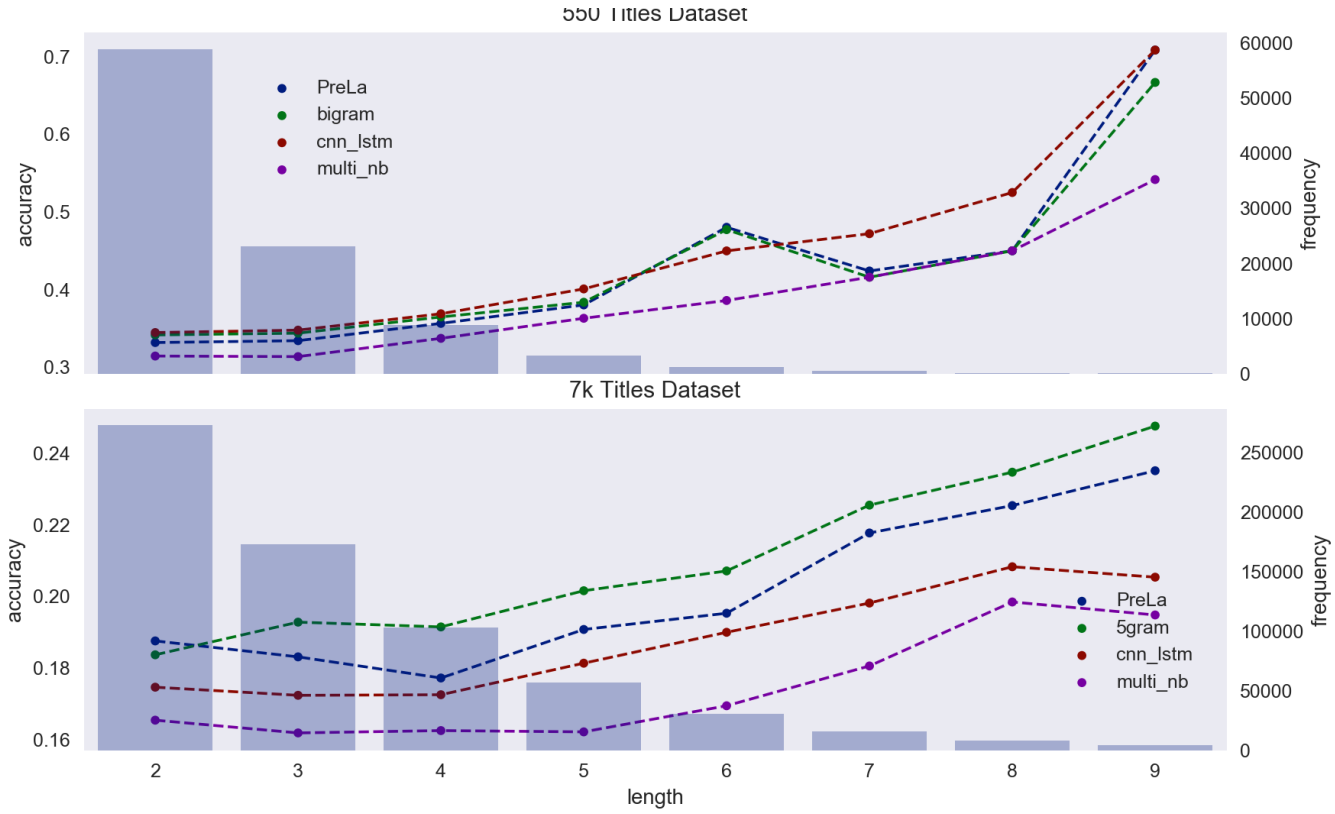


FIG. 4.7. Prediction accuracy by sequence length of the candidates job history

dataset (i.e. we look at how many candidates change job positions with respect to the length of their careers). The plot tells us that a candidate tends to stabilize or stay at the same level or position the further they are in their career.This observation is further strengthened by the fact that the PreLa baseline shows a clear performance boost as well, indicating that the chances of staying at the same position increases as candidates progress further in their career. Consequently, this indicates that what causes a better performance on longer sequences is that during training, the space of observed job transitions becomes smaller for longer sequences. Therefore, the models would have a smaller subset of job title labels to predict from when given longer sequences.

Another factor, is that for longer sequences, the prediction targets are contained within the set of most frequently occurring job titles in the training data. We see this in figure 4.8. We take the job title distribution shown in figure 4.1 and overlay it by highlighting the job titles we want to predict with a red vertical band to better visualize their frequencies. We did this for job history sequences of different lengths, shown in the figure as the subplots (see plot titles), for both datasets. We notice that for longer sequences, the target job title labels (the one we would like to predict) tend to be one of the most frequent job titles in the training set and for smaller sequences, the possible prediction targets are more evenly

distributed throughout the observed job titles. So it seems that in addition to having less movement to different jobs as a person progresses in their career they also tend to converge to common job titles.

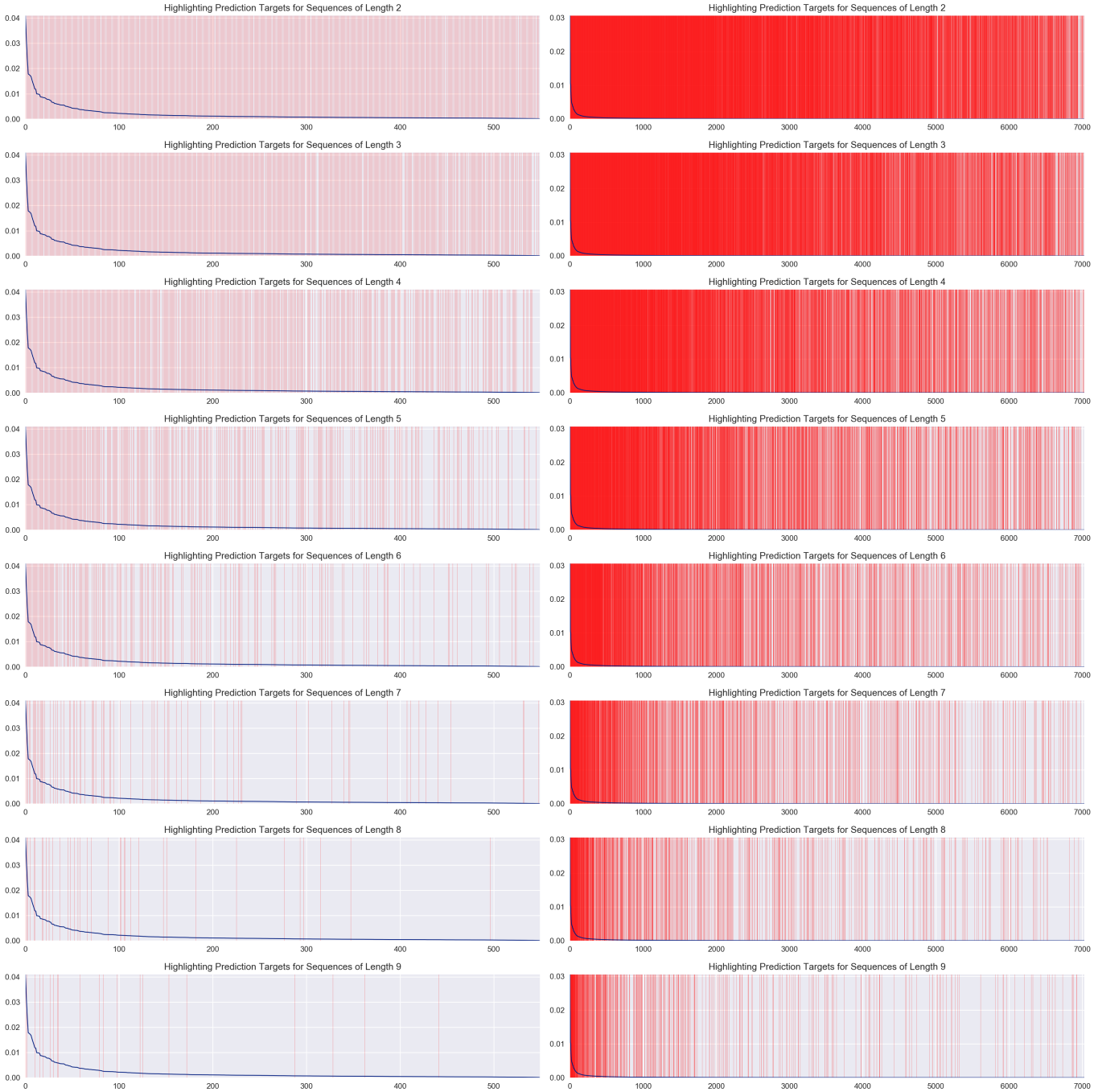


FIG. 4.8. The blue curve is the distribution of the target job title labels on our **training set**. The overlaid red highlighting are the target labels in the **test set**. This illustrates that for longer sequences (or job histories) the target label in the testing phase tends to become one of the most common ones.

4.2.5. Multi-label/Multi-class comparisons

Now, we compare both approaches by first looking at tables 4. I and 4. II for a quick glance at how the models perform. We are looking at the same metrics that were presented in section 3.5.6

TAB. 4. I. Comparing metrics for all models on the `550-titles` dataset

Metrics	Exact	Strict		Best 5		Loose
	Accuracy	Precision	Recall	Precision	Recall	Accuracy
PreLa (baseline)	34.05%	44.30%	44.02%	N/A	N/A	51.26%
bigram	34.97%	45.14%	44.32%	N/A	N/A	51.54%
Multi NB	32.10%	42.25%	41.00%	N/A	N/A	48.28%
fasttext CNN-LSTM	**35.36%**	**45.66%**	**44.68%**	N/A	N/A	**51.93%**
Multilabel CNN-LSTM	24.53%	38.19%	35.81%	21.28%	58.88%	48.46%

In the case of the `550-titles` dataset, the CNN-LSTM using the mutli-class approach seems to be performing the best on all metrics, in the next section we take a closer look at the type of errors between the models.

TAB. 4. II. Comparing metrics for all models on the `7000-titles` dataset

Metrics	Exact	Strict		Best 5		Loose
	Accuracy	Precision	Recall	Precision	Recall	Accuracy
PreLa (baseline)	**19.31%**	**30.13%**	**29.85%**	N/A	N/A	**37.87%**
5-gram	16.55%	30.00%	28.97%	N/A	N/A	36.43%
Multi NB	16.55%	28.01%	25.65%	N/A	N/A	33.49%
fasttext CNN-LSTM	17.70%	29.75%	27.44%	N/A	N/A	35.65%
Multilabel CNN-LSTM	9.80%	23.27%	19.33%	16.85%	44.91%	35.31%

For the `7000-titles` dataset however, the PreLa baseline is the best performing model followed closely by the N-gram model. It is important to keep in mind that this dataset contains 14 times more class labels adding to the complexity of the learning task.

4.2.6. Error Analysis

In an effort to understand how the different models behave when trying to make a prediction we start by looking at how much the models are biased towards very common occurrences of job titles or transitions between a job title to another. Table 4. III shows us the proportion of mistakenly predicted final transition are part of the most frequently occurring

transitions. To do that, we looked at all the transitions (or bigrams) that are observed in the training dataset, ordered them by frequency and built a mapping dictionary that maps every job title (all 550/7000 of them only once) to the most likely next step. Then, we compare the predicted final job for each example in our test dataset with the most common next step observed in the training data given the before last job title in the sequence. The table shows from the errors that a model made, what proportion of these mistakes are the model predicting the most likely next step as seen in the training set.

Models on `550-titles`	**Proportion of Errors**	**Models on `7000-titles`**	**Proportion of Errors**
PreLa baseline	89.65%	**PreLa baseline**	97.58%
bigram model	95.26%	**5gram model**	55.31%
CNN-LSTM	64.89%	**CNN-LSTM**	39.20%
Multinomial NB	49.68%	**Multinomial NB**	29.39%

TAB. 4. III. How many of the mistakenly predicted job titles were part of the most frequent job titles strings in outr training dataset.

Table 4. III, shows, unsurprisingly, that when the bigram model is mistaken it is because the model predicted the most likely next step (95.26% of the time), which is not surprising given that this is how the model is actually trained. The naive Bayes model and the CNN-LSTM models aren't as affected by the distribution of transitions in the training dataset, however, a larger proportion of errors for the CNN-LSTM are the most common next steps. This is due to the fact that a recurrent neural network is a sequential model, thus, learns the distribution of the transitions along with other information that could help the model discriminate the test examples when predicting the labels. Given these observations, the more interesting comparisons would be between the naive Bayes model and the neuronal model.

Next, we look at the accuracy of both models when trying to predict a job title that occurs more frequently or a job title that occurs rarely. To do that we look at what is the performance of the models when trying the predict the last jobs that are part of the top 10%most commonly observed in the training dataset and compute their accuracy at predicting those, then we do the same for the rest (the tail end of the distribution). Table 4. IV show us these results on the `550-titles` and table 4. V for the `7000-titles` dataset. We observe that to predict common job titles the neural models outperforms the rest when considering the accuracy on the most common labels. However, the neural networks perform poorly compared to the other models when we look at the least common labels, which is somewhat surprising.

Models	Accuracy for Most Common Labels	Accuracy for Least Common Labels
PreLa	35.34%	32.61%
bigram	37.96%	31.72%
CNN-LSTM	40.97%	29.36%
Multinomial NB	40.41%	23.30%

TAB. 4. IV. How many of the mistakenly predicted job titles were part of the most frequent job titles strings in our training dataset. These results are on the `550-titles` dataset

Models	Accuracy for Most Common Labels	Accuracy for Least Common Labels
PreLa	20.32%	14.44%
5-gram	21.96%	11.72%
CNN-LSTM	22.55%	4.11%
Multinomial NB	21.41%	3.00%

TAB. 4. V. How many of the mistakenly predicted job titles were part of the most frequent job titles strings in our training dataset. These results are on the `7000-titles` dataset

4.2.7. Predictions Sampled from LSTM RNN Trained on the `550-titles` Dataset

In the tables 4. VI, 4. VII, 4. VIII, 4. IX we show randomly selected examples of predictions made by our models along with the desired target and the input sequence. For the multi-label model, we show the best scoring labels.

TAB. 4. VI. Prediction samples from the `550-titles` dataset

Input	student =>coach =>server =>researcher
Target	**research assistant**
PreLa	researcher
Bigram	researcher
Multi NB	student
CNN-LSTM	**research assistant**
Multilabel CNN-LSTM	graduate, **research**, director, teaching, **assistant**

Tab. 4. VII. Prediction samples from the `550-titles` dataset

Input	manager =>manager =>bartender/server
Target	**bartender**
PreLa	bartender/server
Bigram	**bartender**
Multi NB	manager
CNN-LSTM	customer service representative
Multilabel CNN-LSTM	administrator, **bartender**, marketing, manager, server

Tab. 4. VIII. Prediction samples from the `550-titles` dataset

Input	intern architect =>intern architect
Target	**architect**
PreLa	intern architect
Bigram	intern architect
Multi NB	**architect**
CNN-LSTM	intern architect
Multilabel CNN-LSTM	intern, owner, project, associate, **architect**

Tab. 4. IX. Prediction samples from the `550-titles` dataset

Input	financial analyst =>senior associate
Target	**senior associate**
PreLa	**senior associate**
Bigram	manager
Multi NB	senior financial analyst
CNN-LSTM	manager
Multilabel CNN-LSTM	president, manager, analyst, **associate**, **senior**

4.2.8. Learned Job History Context

Finally, we show a projection of the final state vector of the CNN-LSTM network (before applying the fully connected layer) in figure 4.9. This can be interpreted as the professional history context that has been learned for a specific user since it's the last vector that is fed into the dense layer which does the classification. The projection was produced by the t-SNE algorithm. User that ends up at the same jobs seem to have similar professional context vectors. Here we isolated some interesting examples. Chief financial officers seem to be farther away from other types of executives. We observe the same phenomenon with software engineers and other engineers.

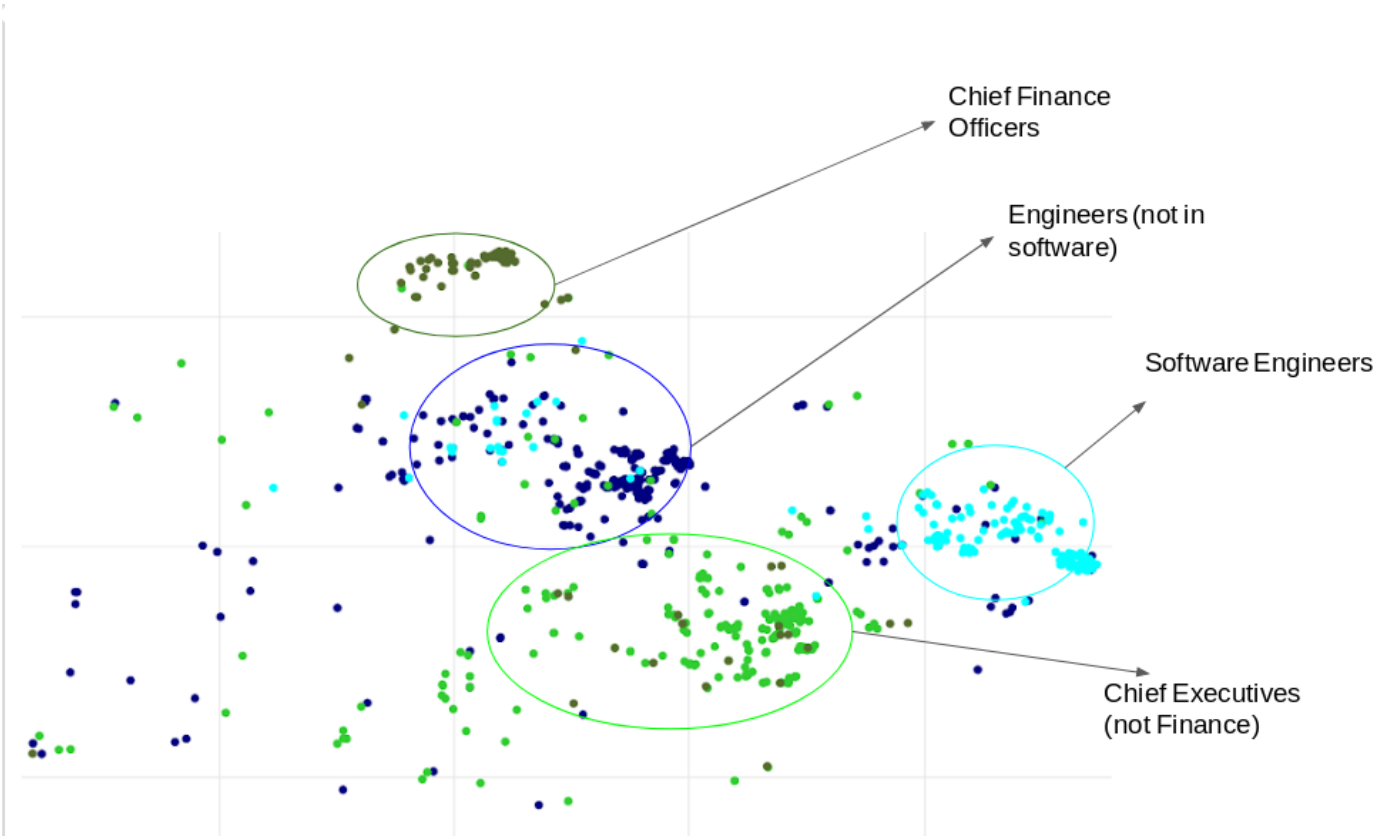


FIG. 4.9. Projection of the learned job history context

Chapter 5

CONCLUSION

In this work, we have explored various methods, inspired mainly by language models, to model a given candidate's career progression by leveraging a subset of the professional social networking site LinkedIn. We have discussed the difficulties in building a reliable predictive model which are mainly due to the distribution over the different job titles present in our dataset. We compared neural models with classical language models applied to our problem and naive bayes methods to surprisingly find that the N-gram models are competitive with the neural models.

The main obstacle was the normalization of the different job title strings, we've applied some heuristic methods in an attempt to generate a more homogeneous dataset. This problem still remains to be explored more thoroughly in future work. Potential solutions could be a rule based approach or a clustering method that could create clusters of job titles that describe very similar responsibilities. This would us to have more coarsely grained target labels we could then predict on these clusters instead of the individual job titles. Pushing this idea a bit further we could also group together the job titles by type like manager or engineer for instance thus yielding a smaller pool of possible prediction targets. Other information could be used as additional features such as the schooling of the candidate, which would have required additional cleaning.

Another, albeit smaller problem we encountered, were the classification conflicts. We attempted to solve this by adding information about the skill set of each user as a feature for a decoder-encoder model, but we did not significantly increase performance. Different methods to augment the input feature vectors to allow the models to be more discriminative between similar career paths in order to make a more reliable prediction for the next step such as the education of a given user or the industry they are a part of have yet to be explored and could be interesting path to follow.

We had 7000 job title for the `7000-titles` and 550 job titles for the `550-titles` that we used as labels to train classifiers on, the dataset was very diverse. Considering that we had

a large amount of different job titles to predict from, the models that were trained perform surprisingly well; we were able to get an accuracy of about 35% on the `550-titles` dataset on the CNN-LSTM model for exact job title matches when the pool from which to choose a job title has 550 different possibilities. Additionally, when looking at the wrong predictions of our models, we have that 64.89% and 49.68% of the wrongly predicted labels by the CNN-LSTM model multinomial Naive Bayes model respectively are part of the 100 most common job titles. Which is an indication that a significant portion of errors made by these models was not due to the fact that the frequency of job titles follows a zipf distribution. Which motivates the need to explore methods to normalize the job titles as discussed above.

Another approach would have been to train a different model for each industry sector. This would maybe give us better results since it would be several very specialized model. However, this did not fit within the scope of this project as it would have required some way to categorize the sector of activities so that we can partition the dataset. We have access to which industry a given user profile belongs too, but this does not translate well into a specific sector of activity (software, manufacturing, marketing, accounting etc.) in a reliable way.

Bibliography

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016. URL http://arxiv.org/abs/1607.04606.

Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759, 2016. URL http://arxiv.org/abs/1607.01759.

Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL http://arxiv.org/abs/1408.5882.

R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184 vol.1, May 1995. doi: 10.1109/ICASSP.1995.479394.

Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014. URL http://arxiv.org/abs/1405.4053.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL http://arxiv.org/abs/1301.3781.

Christopher Olah. Understanding lstm networks, 2015. *URL https://colah. github. io/posts/2015-08-Understanding-LSTMs*, 2015.

G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975. ISSN 0001-0782. doi: 10.1145/361219.361220. URL http://doi.acm.org/10.1145/361219.361220.

Appendix A

COMPLETE JOB PROFILE IN JSON FORMAT


```
{
 '_id': ObjectId('52b31a870b045119318b456e'),
 'addedValue': 46.0,
 'administrativeAreaLevel1': 'CA-ON',
 'bbox': {'east': -79.02891052158193,
  'north': 43.9788806322227,
  'south': 43.42134694377731,
  'west': -79.80369786641808},
 'city': 'TORONTO',
 'collectDate': '2015-08-29',
 'collector': ['scrapinghub'],
 'countryCode': 'CA',
 'createdDatetime': None,
 'duplicateMaster': None,
 'educations': [{'endDate': '1997',
   'name': 'MScE',
   'schoolName': 'Stanford University',
   'schoolSupplierUrl': 'http://www.linkedin.com/edu/school?id=17926',
   'sector': 'Industrial Engineering',
   'startDate': '1996'},
  {'endDate': '1993',
   'name': 'BScE',
   'schoolName': 'Princeton University',
   'schoolSupplierUrl': 'http://www.linkedin.com/edu/school?id=18867',
   'sector': 'Civil Engineering and Operations Research',
   'startDate': '1988'}],
 'experienceLevelMonthNumber': 217,
```

```
'experiences': [{'companyName': 'TELUS',
  'company_id': '55df1c8b0b0451dc5c8bac6b',
  'function': 'Senior Vice President Consumer Marketing',
  'missions': "Support TELUS' continued growth in Consumer markets nationally across bo
  'place': 'toronto, ontario',
  'startDate': '2014-05'},
 {'companyName': 'TELUS',
  'company_id': '55df1c8b0b0451dc5c8bac6b',
  'function': 'VP Mobility Solutions',
  'missions': 'Support TELUS Consumer segment wireless overall business and marketing s
  'startDate': '2008-12'},
 {'companyName': 'Barrick Gold Corporation',
  'company_id': '55df1c4d0b0451dc5c8ba610',
  'endDate': '2003',
  'function': 'Director, Continuous Improvement',
  'missions': """Led global Operations Continuous Improvement team aimed
               at driving hundreds of millions of dollars of operational
               improvement. Pioneer in importing Lean and Six Sigma philosophies
               and tools into the Gold Mining sector.""",
  'startDate': '2000'},
 {'companyName': 'McKinsey & Company',
  'company_id': '55df1c5f0b0451dc5c8ba7f9',
  'endDate': '2003',
  'function': 'Associate',
  'startDate': '1993'}],
'id_scrapinghub': '45070289',
'indexCandidate': True,
'industriesId': [42],
'industry': 'Telecommunications',
'industryId': 42,
'language': 'en',
'lastEditedDate': '2015-08-29',
'lat': 43.70011,
'lng': -79.4163,
'personalBranding_claim': 'SVP Consumer Marketing at TELUS',
'personalBranding_pitch': """Seasoned operations and marketing executive known for
                           creating winning strategies and driving change
                           in complex organizations.""",
```

```
'privacy': 1,
'professionalData_experienceLevelId': 5,
'relationsNumber': 500,
'skills': [{'name': 'Continuous Improvement'},
 {'name': 'Six Sigma'},
 {'name': 'Lean Manufacturing'},
 {'name': 'Operational Excellence'},
 {'name': 'Mining'},
 {'name': 'Wireless'},
 {'name': 'Marketing Management'},
 {'name': 'Strategy'},
 {'name': 'Business Strategy'},
 {'name': 'Change Management'},
 {'name': 'Competitive Analysis'},
 {'name': 'Cross-functional Team...'},
 {'name': 'Leadership'},
 {'name': 'Management'},
 {'name': 'Product Marketing'},
 {'name': 'Strategic Planning'},
 {'name': 'Product Management'},
 {'name': 'Team Leadership'},
 {'name': 'Business Process...'},
 {'name': 'Telecommunications'},
 {'name': 'Management Consulting'},
 {'name': 'Program Management'},
 {'name': 'Project Management'},
 {'name': 'Operations Management'},
 {'name': 'Vendor Management'},
 {'name': 'Business Analysis'}],
'supplierParty': 'linkedin',
'updateDatetime': '2015-12-21 13:38:57'
}
```

Appendix B

APPLIED REGULAR EXPRESSIONS

```
# Remove URLs and/or emails from job titles
pattrn = re.compile(r"[\S]+\.(ca|com|org|fr|gov|net)")
df["transformed"] = df["transformed"].str.replace(pattrn, "")

# Replace .NET by dotnet to eliminate complications when removing periods
df["transformed"] = df["transformed"].str.replace(".net\b", "dotnet")

# Replace periods between word by a space
pattrn = re.compile(r"\.(?=[\w]{2})")
df["transformed"] = df["transformed"].str.replace(pattrn, " ")

# Replace periods in acronyms by empty string
df["transformed"] = df["transformed"].str.replace(".", "")

# Remove parens character
df["transformed"] = df["transformed"].str.replace(re.compile(r"\(.*\)"), "")

df["transformed"] = df["transformed"].str.replace(re.compile(r"co(?!\w)(\s)*-?(\s)*"), "
df["transformed"] = df["transformed"].str.replace(re.compile(r"\bcofounder\b"), "co foun
df["transformed"] = df["transformed"].str.replace(re.compile(r"\bcoowner\b"), "co owner"

transforms = [
    # senior/junior
    (re.compile(r'\bsr\b'), 'senior'),
    (re.compile(r'\bjr\b'), 'junior'),
    # IT
    (re.compile(r'\bit\b'), 'information technology'),
```

```
# C*O
(re.compile(r'\bceo\b'), 'chief executive officer'),
(re.compile(r'\bcoo\b'), 'chief operating officer'),
(re.compile(r'\bcto\b'), 'chief technology officer'),
(re.compile(r'\bcfo\b'), 'chief finance officer'),
(re.compile(r'\bchief financial officer\b'), 'chief finance officer'),
(re.compile(r'\bchief operations officer\b'), 'chief operating officer'),
# VP
(re.compile(r'\bvp\b'), 'vice president'),
(re.compile(r'\bvice-president\b'), 'vice president'),
#technician vs tech
(re.compile(r'\btech\b'), 'technician'),
#coop
(re.compile(r'\bco[-|\s]op\b'), 'coop'),
#addon
(re.compile(r'\badd[-|\s]on\b'), 'addon'),
# Nurses
(re.compile(r'\brn\b'), 'registered nurse'),
(re.compile(r'\brpn\b'), 'registered practical nurse'),
#T.A.
(re.compile(r"\bta\b"), "teaching assistant"),
(re.compile(r"\bteacher assistant\b"), "teaching assistant"),
(re.compile(r"\bteacher's assistant\b"), "teaching assistant"),
(re.compile(r"\bteacher's assitant\b"), "teaching assistant"),

(re.compile(r"\bra\b"), "research assistant"),
# HR
(re.compile(r'\bhr\b'), 'human resources'),
# Customer service reps
(re.compile(r'\bcsr\b'), 'customer service representative'),
# qa / qc
(re.compile(r'\bqa\b'), 'quality assurance'),
(re.compile(r'\bqc\b'), 'quality control'),
# database administrator
(re.compile(r'\bdba\b'), 'database administrator'),
(re.compile(r'\bdatabase admin\b'), 'database administrator'),
(re.compile(r'\bdb admin\b'), 'database administrator'),
# financial service representative
```

```
    (re.compile(r'\bfsr\b'), 'financial service representative'),
    # Misspellings
    (re.compile(r'\bfreelance\b'), 'freelancer'),
    (re.compile(r'\bdesiginer\b'), 'designer'),
    (re.compile(r'\bbiomed\b'), 'biomedical'),
    (re.compile(r'\bgoverenment\b'), 'government'),
    (re.compile(r'\bmachanic\b'), 'mechanic'),
    (re.compile(r'\bbusiness owner\b'), 'owner'),
    (re.compile(r'\br(\s)*&(\s)*d\b'), 'research development'),
    (re.compile(r'\br and d\b'), 'research development'),
    (re.compile(r'research/development'), 'research development'),
    (re.compile(r'\beditor-in-chief\b'), 'editor_in_chief'),
    #eit => engineer in training
    (re.compile(r'\beit\b'), 'engineer in training'),
    #ESL
    (re.compile(r'\besl\$'), 'english as a second language instructor'),
    (re.compile(r'\besl instructor\b'), 'english as a second language instructor'),
    (re.compile(r'\besl teacher\b'), 'english as a second language instructor'),
    (re.compile(r'\benglish as a second languageˆ'), 'english as a second language instr
    (re.compile(r'\benglish as a second language teacher\b'), 'english as a second langu
    (re.compile(r'\besl instructor\b'), 'english as a second language instructor'),

    (re.compile(r"\binternship\b"), "intern"),
    (re.compile(r"\bsummer intern\b"), "intern"),

    (re.compile(r"\bis\b"), "information systems"),
    (re.compile(r"\bgis\b"), "geographic information system"),
    (re.compile(r"\bpr\b"), "public relations"),
    (re.compile(r"\badmin\b"), "administrator"),
    (re.compile(r"\brep\b"), "representative")
]
```

Appendix C

LIST OF TOP 500 JOB TITLES

customer service representative	owner	sales associate
project manager	administrative assistant	research assistant
president	sales representative	manager
intern	consultant	account manager
teaching assistant	cashier	general manager
software developer	assistant manager	director
volunteer	server	receptionist
teacher	store manager	sales manager
graphic designer	operations manager	project coordinator
business analyst	supervisor	account executive
student	executive assistant	instructor
software engineer	accountant	office manager
associate	customer service	vice president
registered nurse	principal	partner
chief executive officer	financial analyst	sales
controller	project engineer	senior consultant
executive director	labourer	marketing manager
business development manager	branch manager	marketing coordinator
web developer	office administrator	senior project manager
owner/operator	founder	senior accountant

office assistant	financial advisor	summer student
staff accountant	team leader	legal assistant
research associate	co founder	production manager
analyst	bookkeeper	designer
barista	sales consultant	accounting clerk
bartender	electrician	product manager
program manager	managing director	quality assurance analyst
human resources manager	administrator	board member
general labourer	waitress	cook
member	territory manager	technician
chief finance officer	researcher	senior software engineer
realtor	senior software developer	tutor
marketing assistant	brand ambassador	editor
hostess	programmer	senior business analyst
lawyer	line cook	security guard
human resources assistant	photographer	coop student
professor	programmer analyst	co owner
director of operations	mechanical engineer	lecturer
accounting assistant	production supervisor	team lead
systems analyst	personal support worker	senior manager
information technology consultant	art director	program coordinator
buyer	developer	english as a second language instructor
english teacher	human resources coordinator	senior financial analyst
assistant professor	assistant	production assistant
marketing intern	senior account manager	process engineer
regional sales manager	district manager	carpenter
network administrator	inside sales representative	business development
secretary	creative director	information technology manager

engineer	assistant store manager	graduate teaching assistant
writer	human resources generalist	machine operator
shift supervisor	coordinator	laborer
heavy equipment operator	welder	event coordinator
producer	financial services representative	retired
operator	technical writer	service manager
system administrator	business manager	postdoctoral fellow
design engineer	customer service manager	security officer
physiotherapist	merchandiser	mechanical designer
personal trainer	technical support	recruiter
driver	truck driver	research analyst
electrical engineer	president & chief executive officer	chief operating officer
production coordinator	accounting manager	service technician
finance manager	graduate research assistant	senior associate
independent consultant	sales coordinator	freelancer writer
millwright	sales assistant	account coordinator
junior accountant	registered practical nurse	communications coordinator
accounts payable clerk	team member	foreman
computer technician	clerk	systems administrator
sales executive	regional manager	data entry clerk
director of sales	laboratory technician	marketing director
operations supervisor	senior account executive	managing partner
plant manager	property manager	general laborer
senior analyst	inside sales	reporter
front desk agent	pharmacy assistant	articling student
pharmacist	account director	treasurer
estimator	systems engineer	registered massage therapist
freelancer graphic designer	lifeguard	architect

data analyst	animator	investment advisor
director of marketing	quality assurance manager	field engineer
program assistant	senior developer	pharmacy technician
contractor	graduate student	research intern
customer service associate	educator	logistics coordinator
human resources consultant	business consultant	customer service agent
actor	law clerk	owner operator
national account manager	application developer	web designer
communications officer	superintendent	office clerk
dispatcher	retail sales associate	key holder
marketing specialist	crew member	technical support specialist
occupational therapist	technical analyst	management consultant
district sales manager	director of finance	chef
geologist	human resources advisor	interior designer
artist	associate professor	principal consultant
engineering manager	area manager	student intern
marketing consultant	project assistant	social worker
president and chief executive officer	assistant controller	customer service specialist
system analyst	coach	advisor
construction manager	engineering intern	senior engineer
auditor	key account manager	technical consultant
payroll administrator	undergraduate research assistant	field technician
database administrator	technical sales representative	camp counselor
human resources administrator	information technology specialist	senior designer
educational assistant	freelancer	quality control inspector
cashier/customer service	pipefitter	phd student
lab technician	technical support analyst	server/bartender

self employed	technical support representative	housekeeper
trainee	student nurse	programmer/analyst
department manager	warehouse manager	policy analyst
owner / operator	delivery driver	counsellor
business systems analyst	journeyman electrician	warehouse worker
executive chef	sous chef	architectural technologist
practicum student	mechanic	project leader
substitute teacher	cleaner	financial services manager
early childhood educator	administrative coordinator	phd candidate
financial controller	banquet server	sessional instructor
administration	support worker	assistant general manager
real estate agent	restaurant manager	quality assurance
research scientist	management trainee	vice president finance
trainer	intern architect	broker
medical office assistant	translator	structural engineer
communications specialist	lead hand	accounts payable
communications assistant	equipment operator	national sales manager
marketing associate	purchasing manager	electrical apprentice
mortgage agent	project administrator	peer tutor
marketing	board of directors	travel consultant
chief technology officer	program director	copywriter
field service technician	information technology analyst	student teacher
maintenance supervisor	vice president operations	shipper/receiver
network engineer	product specialist	human resources intern
technical director	flight attendant	sales clerk
service advisor	senior quality assurance analyst	operations coordinator
shift manager	account representative	waiter
private tutor	communications manager	director of business development

project lead	territory sales manager	occasional teacher
front desk receptionist	manufacturing engineer	quality assurance specialist
quality manager	dishwasher	client service representative
recruitment consultant	camp counsellor	safety advisor
production worker	painter	quality assurance tester
customer service supervisor	brand manager	senior systems analyst
credit analyst	bilingual customer service representative	guest service agent
landscaper	case manager	accounts receivable clerk
various	accounting technician	senior buyer
java developer	account supervisor	site supervisor
grocery clerk	technical specialist	human resources
senior graphic designer	caregiver	chair
maintenance manager	student-at-law	placement student
network analyst	adjunct professor	financial service representative
facilitator	sales engineer	child and youth worker
system engineer	baker	information technology technician
paralegal	warehouse supervisor	apprentice electrician
administrative support	author	research engineer
various positions	associate lawyer	team manager
financial planner	operations	laboratory assistant
legal counsel	financial consultant	planner
quality engineer	assistant director	salesperson
civil engineer	research coordinator	student ambassador
research technician	pharmacy student	data entry
outside sales representative	nursing student	pilot
marketing analyst	yoga instructor	senior programmer analyst
internal auditor	insurance broker	volunteer coordinator
quality assurance technician	independent contractor	general accountant

corporate controller	production engineer	owner/manager
commercial account manager	staff nurse	youth worker
public relations intern	engineer in training	library assistant
employee	senior auditor	hairstylist
area sales manager	communications intern	assistant project manager
sales and marketing manager	machinist	communications consultant
compositor	apprentice	software designer
visual merchandiser	senior project engineer	

Appendix D

LIST OF BOTTOM 500 JOB TITLES

glap shipper	vice president marketing - first year committee	departmental liaison, recruitment	
packaging solutions	[42]manager	reiki masterteacher	
associate - repair dispatch	summer student - entreprise risk management	senior data technician - production, operations, engineering	
volunteer coordinator and event manager	software/product verification coop	project manager, npi projects	
sales/training/marketing	intake manager,human resources coordinatior	district people manager, human resources	
custom mug designer	women's accessories team lead/floor supervisor	product marketing, document solutions	
london health sciences foundation	software developer/software manager	information technology client	dotnetwork supp canada
jursi/medical student	transportation centre research assistant	coordinator - formation training group	
chemical team co manager	customer service representative/ check-in agent	customer area sales representative	
system manager, information technology-specialist	certifier - medical	laboratory equipment	brokerage associ centre represent
registered nurse- primary care	cea homeschool teacher	manager - strategic initiatives, superstore/hard discount	

field recordist, sound editor, and library curator	athletic therapist - ubc football	ghg services manager	
manager human resources - heavy oil	public relation	community manager	contracted drive
co head counsellor: referees	senior regulatory affairs officer - civil aviation	principal technical specialist - contract	
employment counselor/recruiter	real-estate evaluator	estimator	driller c
manager, surveillance design	director of enterprise projects	senior mechanical design student	
canadian director, human resources and communications	project coordinator, customer service, trainer	medical laboratory technologist/ technician	
quality assurance mechanical / welding	research student - field worker	director editorial	strategic partn legal, business resources
sports editor; opinions editor; news editor	roulette dealer/pitboss	commercial mariner and wildlife naturalist	
director - licensing, technology contracts	trade compliance	lead game designer/project manager	janitorial and repair
business operations analyst - legal and contracts	charter/founding member	procurement	maintenance ma
sales representative, quality control and sales verification	patent of high-level technician, as regards hotels management	job capitan of construction team, senior structural designer	
interior decorating, landscape design	athelet	clinicial development nurse	
guest relationship executive	faculty -cca program	java sharepoint integration specialist - cibc wholesale portal	
associate partner, transfer pricing services	photo intern at hk magazine	java ee	gwt programmer

stylist/colourist/barber	event coordinator, public relations, host	visa officer, department of immigration and citizenship	
specialist representative cardiovascular	digital tattoo project coordinator	front and kitchen staff	
recently retired as director of administration	avp	actuary, corporate actuarial	senior web des end web develop
quality assurance coordinator - finished product examination	native post secondary education counsellor	corporate director hse	
environment and cultural heritage assistant	ptp business process expert	crossmark canada merchandiser	
renewals support administrator specialist	student- agriculture and food security	information and technology service manager	
senior category manager, national wholesale	bénévolat / parrainage pour étudiants étrangers	parts clerk/customer service representative	
independant contractor - backend developer	special projects/ purchasing manager	instructor part time evenings	
hardware quality assurance engineer, mswam test engineering	senior vendor consultant	software engineering	corporate databa
plaque program coordinator	ibm websphere instructor	intermediate developer, application services	
project coordinator, hidden gems	associate woodworker	habitat biologist and field researcher	
clerical night staff	leadership, speech	communication coach	prepress speci colour operator
cashier breaker	tree lot sales associate	coop buyer/planner for plant services canada	
local store marketing team	vice predident	information technology/web devlopment	
business analyst, counterparty credit risk systems	operation building manager	technical services representative - team lead	

a licenced automotive technician	faculty of arts	social sciences councillor	incident manage prise accounts
assembly worker and heavy machinery worker	supervisor building security	senior planner and project control	
director, acute and tertiary care	marketing supervisor / route manager	hamilton relocation engineer	
online marketing and social media consultant	nighttime custodial supervisor / project manager	store manager assistant/contract	
administrator-consulting services	program manager, epidemiology	biostatistics	project engineer resentative
master of physical therapy intermediate placement	junior assistant claims analyst	labourer, concrete finisher	
chief geologist, west africa	manufacturing/applications engineering manager, custom rollforming production services	manager of team development	photographer
electrical 2d cad support / microstation standards co ordinator	us accounting manager	marketing - higher education	
human resources - employee relations advisor	subject matter expert - immigration and ethnocultural statistics section	safety	haccp complianc tor
crisis line intervention specialist	instructor	supply	general manage america and cari
loan underwriter/ credit analyst	independent genealogy researcher	associate producer for wajd: music, politics	ecstasy
manager, public sector strategy and operations	freelancer writer, video producer-writer, communications consultant	administrator/head of retail	
copy and print operator	project manager, canadian writing research collaboratory	ibm toronto lab	
st michael‘s college orientation leader	executive team - operations director	senior manager, people and organization effectiveness	

area sales representattive	manager, sales business development	information technology architect / supervisor information technology operations	
chiropractor/clinical acupuncture/mechanical traction/orthotics/bracing/compression therapy	hse statistician	resource technician/protection assistant	
pixels	hand finisher/ draper	principal, koven	associates
senior information technology infrastructure analyst / vmware engineer	registered nurse, team leader, mood inpatient unit	judge and time keeper	
commercial account consultant, marketing coordinator, book keeper	vice president for marketing department of university art group	bartender in concessions, banquet server, server	
social worker: inpatient and outpatient mental health programs	college professor in human biology	fund raising project manager	
independent consultant - workshop facilitator	lead software engineer / architect consultant	fddfdf	
homme a tout faire men of all trade	reseaarch officer	business systems analyst, aml program	
scientific board member	marine service technition	server, office administrator and event coordinator	
patient food supervisor-	student and assistant secondary teacher	partnering committee member	
mt sheds	onsite logistics management	team leader toronto consumer call center	
casual patient services clerk - ambulatory care and mental health	marketing and sport tourism intern	electrical engineer	project engineer
chair joint occupational health and safety committee, server and devo	process improvement analyst - supply chain	receptionist/junior accounts payable	

manager - crm development and implementation	self employed welder and fabricator/ owner	customer care representative / technician dispatch	
international conference manager	vegetarian cook	manager	release lead for trade
personnel registration	bmc partner	small owner-cleaning houses	
broker, office leasing	sales	shop supervisor/cnc programmer	billing specialists
sap quality management expert	dining room server, lead hostess	- vice president, global equities, external management	
branch manager/mutual fund advisor	wayside team lead	director of maintenance engineering utilities and environment	
intern of customs control	account manager, technologies	currently looking for maintenance and reliability roles in vancouver	
registered nurse/administrative nsg coordinator	senior intergration analyst	regional client base representative	
assistant manager - e commerce	vice president communications/corporate services	clinical counseling intern	
airframe technician ch 147 chinook helicopters	project management / controls services	ceramic industry consultant	
design consultant, project coordinator	opso engineering student	site expeditor/vendor lead	
chauffer and lead coordinator	intern at the information technology department	senior manager - eastern canada transportation	
special projects	major accounts administrator	human resources/ finance and payroll, store operations and recruitment	dance captain events program act
head of cross border legal and compliance, west	supplier quality assurance administrative assistant	board operator / street team	
intern/junior law clerk	domain expert, billing	revenue managment vision project	sports fields turf

producer/director, owner	front end division manager western canada	packaging design manager, retail brands	
assistant buyer - womenswear-hr2	ici	municipal estimator, project supervisor	promotions/even
online support consultant	journeyman carpenter/maintenance	volunteer for special olympics	
seasonal work for sales during the fall	regional key account manager- eastern ontario	manager rf group	
director, ministry of energy and infrastructure	health and safety training administrative assistant	manager, education leadership canada	
international sales executive/promotional model/hair extension instructor	owner/partner - sales	marketing director	other training
completions qhsse coordinator/environmental specialist	treasury	accounting	team lead, engi mechanical desig
sales associate / kennels	business reference assistant, business information centre	information technology system administrator / project manager assistant	
brown's food service kitchen helper	director national distribution	a time-based paradigm for us airspace data	
independent translation contractor	technical research	quality associate	retail supervisor
consultant - dot2dot ventures	manager, aboriginal relations and business development	vice president impact events	
vice president, contracts	technical services	regional manager, international airports	harbour master manager marine
obstetrics and gynecology	head host/trainer	women's couselor	
customer service representative/task team	service planning dispatcher	cementing field specialist	
bim / revit leader	temporary contract sales analyst	business intelligence	data warehou tion/data archit

shop foreman / health and safety coordinator	performance evaluation specilist	psychometrist, psychological associate and psychologist	
networking event coordinator	marketing director, international domestic	marketing	buyer assistant
research assistant and personal secretary	consultant - architect - emerging technologies - nfc and emv payments and loyalty	manager of educational services	outdoor leaders director
senior alarm coordinator	probono legal counsel	research eng/ phd	
staff paralegal	national manager, nav canada, national operations centre	client service representative, hazardous waste informationdotnetwork	
ohs and dangerous goods officer	infection control speacialist	personal loan support officer	
dietary aide- part time	paramedic student intern	estimating administrative support	
director of lectures	manager, fx sales - manitoba, and saskatchewan	senior analyst, immigration	
associate producer of village global 2013 slate, actor	manager- staff	facility	assistant to dean
lot attendant/ salesman	student fashion technician	project manager, mobile business solutions	
brands finance manager	o/m assistant manager	senior marketing analyst, demand planning	
trimmings buyer/inventory	general manager - ms govern	artist in fashionality exhibition and art residency	
mla melfort	swing-manager	unix technician analyst	
commerical administrative assistant	staffing coordinator-	electric board repair	
marketer/architectural technologist	consulting and investing	bilingual account and technical representative	

consultant - financials	parts, production	warehousing	aesthetic podiat beautification
sales management	sales leadership roles	superintendent for condominium complex- 359 units	instructor/guide
gocompletions database coordinator - preservation	bc hydro contracts / accounts payable / payroll clerk	go code girl instructor, faculty of engineering	
student clinician: speech-language pathology- fluency disorders	team lead, business analysts - operational planning	food server at rendezvous	
television programme producer and events coordinator	resale real estate sales representative	french language lecturer	
read with me program volunteer	law coop student - consumer protection policy	student sports camp volunteer	
sportscaster/producer	bridal couture	on call relief rural and suburban mail carrier	
member higher education curriculum committee	assistant team leader/ fitness assessor	group brand director: neutrogena and clean	clear
field underwriter - personal lines	engineering and procurement supervisor	retired - board of trustees	
vehicle systems engineer-electrical	assistant editor: french german regional tv magazine	intern, investment assistant	
quality assurance/quality control manager-suncor firebag oil sands site	assist land surveyors locating and managing control information	special olympics coach/judge	
superviseur en flore	junior processs/equipment engineer	presenter & participant of the 35th annual residence life conference	
new movie outline of ' the first candle '	executive member of equal voice carleton chapter	trauma-informed care consultant	trainer
long term contract research assistant for dr chelsea willness and vince bruni-bossio	aircraft techician	visiting scientist at the electrical and computer engineering department	

customer service / receiver / exporter	regional manager, dealer relations, western canada	certified journeyman welder/red seal	
vice president, global risk management, latin america	3d artist	producer	consumer imp keting/launch! ambassador
co instructor and teaching assistant, executive programs	undergraduate programs	lead animator for backyardigans season 4	information m and governance
artistic director, hornist, electronics	morning show host - promotions and marketing manager ckda/cfms	director lands, resources and heritage	
lawyer at carfra lawton llp	specialist in organization and standardization of labor	sales manager, facility supplies- toronto gta and london	
asia-pacific economic cooperation , senior policy officer	women's health program assistant	level 1 child care supervisor	
director - product catalogue collection	senior manager / director enterprise professional services	vartanian research group - research assistant	
adult learner tutor	installation manager for western canada	vice president, human resources	education
thinking outside the box chief executive officer of owning your life	sales consultant - cell	lean leader - boeing advanced quality system	
designer / stylist	cf-18 hodotnet test pilot	tool makers apprentice	
superviseur informatique	territory manager retail sale	project coordinator for marketing and business development	
facility coordinator / receptionist	director of events	communication	director - meetin tor
geological technologist/ engineering technologist	manager implementation	coordination	customer suppo nance engineeri specialist ahms

teaching assistant for intro statistics	45th circuit program committee chair	dirctor of advertising	
application developer/quality assurance lead	database segment sales leader - canada	solutions director - service sales	
infastructure project - senior team lead	training development manager, lead training engineer,	senior instructional systems designer	landscaper and c
accounts payable assistant and word processor i	project manager, web integration	senior i	c consultant
marketing research on better prices of the product	conference floor supervisor	barrister - pupillage	
network planner and sales	assoc dir-senior solution prime	manager accounts, administration and commercial	
field supervisor/ driver/ mechanic	return to work / disability management coordinator	project manager psychological health	
facilities and marketing manager, events and sales coordinator	in-building systems engineer	peoplesoft support system analyst	
special event coordinator/marketer	promotional model, brand ambassador and spokesperson	medical device design intern	
teaching assistant - foundations for collaborative work environments	consultant, chef ambassador	student engineer i	
lead demand generation executive	replenishment manager - sporting goods	manager, technology workgroup	
senior analyst - fraud analytics	advertising and events coordinator	lead journeyman lineman/ safety administrator	
promoted sales	regional director, alberta environment	cashier knowledge	
senior delivery manager - infrastructure service delivery	artist, display design, floral design	program coordinator & web developer	

senior project manager - business technology management	receptionist & pre-tester	ecohawks general member	
provost district manager	accounts payable / inventory control	educational assistant/behavioural consultant	
category analyst: laptop computers, computer accs, digital media , printers and ink	downstream processing	jewellery consultant at peoples jeweller	
manager of digital marketing, tourism pei	program manager, exhibitions and outreach	server monitoring analyst	
vice president, chief architect, co founder	bruce a restart eq programme manager	manager rail assurance	
engraving jewelry	content manager for do416	assisting other staff	
accountant, sales and marketing department	remote access sales	vice president of investment research	
assistant coach, women's triple-a basketball team	team lead technician support	playwood technical services	
provincial program manager	previous coordinator/branch volunteer	quality assurance assistant/sensor tester	indstrial enginee
private tuition, assistant and customer assistance	mountainview mall florist	chief finance officer	chief informaitor
condo care	early childhood educator consultant/programmer	head of artist alley	
senior accredited classification	organizational specialist	svp, merchandising and supply chain	shipping/labour
operations mamager	social media and e commerce intern	social media/graphic design intern	
commissioning group manager	regional field manager/team leader		