Learning Career Progression by Mining Social Media Profiles

Zakaria Soliman[1(✉)], Philippe Langlais[1], and Ludovic Bourg[2]

[1] Université de Montréal, Montreal, QC H3C 3J7, Canada
{solimanz,felipe}@iro.umontreal.ca
http://rali.iro.umontreal.ca/rali/en/
[2] LittleBIGJob, Montreal, QC H3B 4W5, Canada
ludovic.bourg@lorenzandhamilton.com

Abstract. With the popularity of social media, large amounts of data have given us the possibility to learn and build products to optimize certain areas of our existence. In this work, we focus on exploring methods by which we can model the career trajectory of a given candidate, with the help of data mining techniques applied to professional social media data. We first discuss our efforts to normalizing raw data in order to get good enough data for predictive models to be trained. We then report the experiments we conducted. Results show that we can predict job transitions with 67% accuracy when looking at the 10 top predictions.

Keywords: Data mining · Machine learning · Carrier path prediction · Contextual embedding

1 Introduction

Labor flow networks have been an area of interest by researchers in the social sciences and by economists [2–4]. Labour mobility between industries has been shown to have a positive effect on economic indicators [3]. For this work, we gathered a large collection of user profiles, with the motivation of being able to predict a set of plausible recommended positions that a professional can take as his next career move given his working history. However, systems to predict the next job position has received less attention. Building predictive models that are personalized for every individual professional is a hard task because it involves several hard subproblems that need to be solved such as normalizing the large number of job titles, and skill set since these are all defined by individuals. This results in job titles that seemingly have the same function but different names for the role. Recommendation systems have been proposed as well [1,10] which are based on whether or not a user applies or clicks for a recommended job.

In this article, we discuss in Sect. 2 the specificities of the data we collected, and our approach to design a dataset amendable to benchmarking. We report in Sect. 3 the predictive models we implemented, the results they obtained in Sect. 4, and conclude in Sect. 5.

M.-J. Meurs and F. Rudzicz (Eds.): Canadian AI 2019, LNAI 11489, pp. 446–452, 2019.
https://doi.org/10.1007/978-3-030-18305-9_43

2 Dataset

We gathered a very large set of over 9.5 million public user profiles from LinkedIn. We removed the numerous profiles where no job experience was reported, yielding a still substantial set of more than 7.1 million profiles where 40% of users only have one filled out job experience. Since we are primarily interested in modeling career moves, we dropped all user profiles with less than 2 job experiences reported, and focused on profiles written in English[1] to identify the language of a profile. Removing such profiles leaves us with roughly 3 millions ones.

Table 1. Main statistics of our dataset

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
Avr. job title length (# of words)	4.5
Longest job title string (# of words)	42
Avr. length of job history	5.2 (positions held)
Shortest job history	2 (positions held)
Longest job history	140 (positions held)

The main characteristics of our dataset are reported in Table 1. One striking figure of Table 1 is that there is slightly less than twice the number of unique job titles than we have user profiles. In other words, there is more job titles than profiles. Suggesting that we may face problems fitting this data into a powerful predictive model we discuss in Sect. 2.1 our approach at resolving the issue.

2.1 Normalizing Job Titles

The distribution of job title names is expectedly Zipfian, which means that most job titles in our dataset appear rarely. In fact, 98.2% of all the job titles appear less than 10 times illustrating the root of the problem. Rare job titles are actually overly specific, therefore reducing the likelihood that they match another job title.

Looking at Fig. 1, we observe that punctuations and conjunctions seem to separate long strings describing a job title into smaller sub-strings of individual entities in order to extract the most relevant information (the more general job title) out of a longer job title containing specificities that are not relevant to us. For example, we split the job title `co instructor and teaching`

[1] The `langid` [9] toolkit was used: https://github.com/saffsd/langid.py.

```
co instructor and teaching assistant, executive programs & undergraduate...
accountant, sales and marketing department
senior manager, project management methodology & governance
journalist and travel writer
vice president for marketing department of university art group
project manager, key accounts - human health therapeutics
information technology manager - operations & architecture
```

Fig. 1. Random sampling of rarely seen job title strings. Colored text is the reduced job title we would like to have.

`assistant, executive programs & undergraduate programs` into the substrings `co instructor`, `teaching assistant`, `executive programs`, and `undergraduate programs`. Among those, `teaching assistant` is the most frequent one we keep. Manual inspection of the resulting job titles did not reveal any ill-formed job titles.

2.2 Selecting User Profiles

In an effort to minimize the ratio of number of unique job title strings to the total number of user profiles, we looked at what happens when we discard the least common job titles based on some lower bound on their frequency in user profiles. We ordered the job titles in $\mathcal{J}$ in decreasing order of frequency leading to $\mathbf{j} = [j_1, \ldots, j_{|\mathcal{J}|}]$ with $j_{|\mathcal{J}|}$ being the most recent job title. 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:

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

For example, we may look at how many user profiles solely use the 10 most common job titles in their job history (1 166 626), and compare it to how many users exclusively use the 110 most common ones. This leads to a gain of 211.22%. This method resulted in a dataset of 550 job titles with 120 371 user profiles. Considering more job titles yielded small gains only.

3 Experimental Setup and Models

The dataset was randomly split into a train (65%), a validation (15%) and a test (20%) sets prior to any experimentation. All hyperparameter tuning was made on the validation set, and all results presented here are those measured on the test set.

Along with standard classification accuracy, we use the mean percentile rank (MPR), a metric used for recommendation systems [6] as well as for evaluating models for similar tasks [7].

We compared 4 families of models briefly described in the sequel.

Baseline. A simple baseline consisting in always predicting the last job title in the job history. We call it *PreLa* for **Pre**dicting the **la**st seen job title. In the dataset, 34.15% of the profiles have their last experience identical to the previous one.

Naive Bayes. We experimented with the multinomial and bernoulli Naive Bayes variants along with different methods to represent the data features. The input was always a bag-of-words, the difference was in what we considered 'words' (complete job title as a token v.s. individual words as tokens).

N-gram Model. We tested an N-gram language model, which computes an estimation of the transition probability of a sequence of elements, by using the KenLM library [5]. In our case, instead of focusing on a sequence of words in a text, we look at the sequence of job titles.

RNN-based models. We used an LSTM based RNN network as the decoder of our encoder-decoder approach as well as a stand-alone network. The difference is in the initial state $\mathbf{h_0}$ that is the encoder's output in the former case and a zero vector in the latter case. The choice of a recurrent network is due to it's success in sequential modeling [11].

As previously mentioned, we tested variants making use of the job history alone, as well as some incorporating the skills of a user. To do this, we use an encoder that takes as input the skills and outputs a real-valued vector that we interpret as a representation of that user's skills. The networks inputs (skills) were represented by pre-trained, `FastText` [8] embeddings. The output is a real valued vector that is fed into the LSTM decoder.

4 Results

Table 2 shows the MPR on the prediction of the next job title. There is no way to compute this metric on the PreLa baseline since it outputs a single prediction. Figure 2 shows the accuracy of the models when considering the top K predictions, so a model is counted correct if the reference prediction is found in the top K best scored ones.

Table 2 shows that the N-gram models outperform the Naive Bayes model on this metric suggesting that keeping contextual historic information about past job transitions give the models a better understanding of the future outcome. This is also reflected in Fig. 2, where the Naive Bayes model is beaten by every other model.

Looking at Fig. 2, we notice that the LSTM network is lagging behind the other models when we look at predictive accuracy. However, as can be seen in Table 2, when evaluating the models as recommendation engines, both neural models outperform the other model families. It is difficult to clearly understand why that is the case; this would be an interesting area of inquiry for future work. We notice that the addition of the skill set of a user (a feature given to the CNN-LSTM encoder-decoder model) increases the performance of the model as is evident in Fig. 2 as well as in Table 2.

Table 2. Mean percentile rank of the correct job title we are looking for within the model's predictions. A lower value is better.

Models	MPR × 100
Bigram	0.164
Trigram	0.168
Multinomial Naive Bayes	0.213
LSTM	0.148
CNN-LSTM	**0.106**

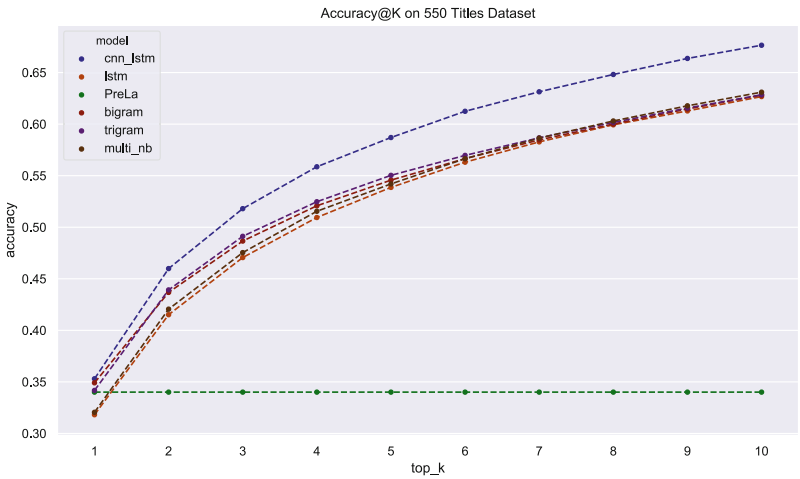

Fig. 2. Prediction accuracy@k for best performing models within each model family.

5 Conclusion

In this work, we have explored various methods to model a given candidate's career progression. We have discussed the difficulties in prepossessing and normalizing the dataset we gathered. A major difficulty for this task is the variation in job title names for the same apparent responsibilities. We compared neural models with classical language models applied to our problem and Naive Bayes methods to find that, surprisingly, the N-gram models are competitive with the neural models, since N-gram models use a smaller historical context (the value of N). This suggests that the positions held at the start of a career don't contribute as much to predict the end of a career. Thus, further motivating the use of a contextual representation of the user profile along with the sequence of experiences a user has had.

Considering that we had a large amount of different job titles to predict from (550 job titles), the models that were trained perform surprisingly well; we were able to get an accuracy at rank 1 of about 35% on the dataset with the CNN-LSTM model for exact job title matches, and we attained 67% accuracy when looking at the top 10 predictions. When looking at the mean percentile rank, the CNN-LSTM gives the correct target within the 6 best scoring prediction labels on average.

Data normalization 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 and use these clusters as prediction labels. This would allow 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.

Additionally, the heuristic presented here introduced a trade off, and we dramatically reduced the size of our dataset. Further investigations on methods to standardize the job titles is an interesting area of inquiry for future work. We also want to benefit the additional information available in user profiles that we discarded here.

References

1. Al-Otaibi, S.T., Ykhlef, M.: A survey of job recommender systems. Int. J. Phys. Sci. **7**(29), 5127–5142 (2012)
2. Bjelland, M., Fallick, B., Haltiwanger, J., McEntarfer, E.: Employer-to-employer flows in the United States: estimates using linked employer-employee data. J. Bus. Econ. Stat. **29**(4), 493–505 (2011)
3. Boschma, R., Eriksson, R.H., Lindgren, U.: Labour market externalities and regional growth in Sweden: the importance of labour mobility between skill-related industries. Reg. Stud. **48**(10), 1669–1690 (2014)
4. Guerrero, O.A., Axtell, R.L.: Employment growth through labor flow networks. PLOS ONE **8**(5), 1–12 (2013). https://doi.org/10.1371/journal.pone.0060808
5. Heafield, K.: KenLM: faster and smaller language model queries. In: Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation, Edinburgh, Scotland, United Kingdom, pp. 187–197, July 2011. https://kheafield.com/papers/avenue/kenlm.pdf
6. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Eighth IEEE International Conference on Data Mining, ICDM 2008, pp. 263–272. IEEE (2008)
7. James, C., Pappalardo, L., Sirbu, A., Simini, F.: Prediction of next career moves from scientific profiles (2018). arXiv preprint: arXiv:1802.04830
8. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification (2016). arXiv preprint: arXiv:1607.01759
9. Lui, M., Baldwin, T.: Langid.py: an off-the-shelf language identification tool. In: Proceedings of the ACL 2012 System Demonstrations, pp. 25–30. Association for Computational Linguistics (2012)

10. Paparrizos, I., Cambazoglu, B.B., Gionis, A.: Machine learned job recommendation. In: Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 325–328. ACM (2011)
11. Sundermeyer, M., Schlüter, R., Ney, H.: LSTM neural networks for language modeling. In: Thirteenth Annual Conference of the International Speech Communication Association (2012)