Sunshine of Your Gov: Predicting Future Salaries of Ontario Public Servants

Thomas Sattolo

January 24, 2019

1 Data Transformation

The full dataset is quite large, over a million entries, and spans 22 years. Importantly, it was never designed to be used for any sort of data science work. Consequently, data cleaning was onerous and time consuming. What we started with was a list of names and with each name is a sector, a job title, an employer and a salary.

1.1 Sectors, Employers and Job Titles

Examples of sectors are "Crown Agencies", "Hospitals" and "Universities", i.e. "Hospitals" is literally one of the sectors employees are sorted into; specific hospitals are employers. Individual universities, government departments, cites and the like are also employers. None of these—sectors, employers and universities—are standardized in any way. An Employer being 'fanshawe' in one year 'fanshawe college' in another or a sector being 'government of ontario : legislative assembly' one year 'government of ontario - legislative assembly & offices' the next is commonplace. The same sort of thing happens with job titles.

The first thing to that needed to be done was transform the sector, employer and job title columns into categorical variables that aggregate all the different names that are used for the same organization into a single category. The number of unique sectors is only twelve, all years combined, so it's practical to simply figure it out manually. Employers and job titles on the other hand number in the tens of thousands so an automated approach is required (note that the number of unique employers and Job titles was not known until after the categorization was done). To begin, we did some obvious things like removing punctuation and making every character lowercase. This is followed up by some more advanced natural language processing technique to remove function words, correct spelling mistakes and strip words to their roots. The latter makes 'taxable' become tax and 'governing' 'govern', etc. Spelling correction had mixed results. First of all, it didn't usually output something sensible when the input word was in French, which was actually quite common. The same is true of proper nouns which some pathological results like 'Landlord Flaming' for

'Sandford Flemming'. Of course, this is not great but since it's deterministic it ought not to a serious problem: 'puckering' matches 'puckering' just as obviously as 'pickering' matches 'pickering', but random spelling errors would be hard to deal with in any other way.

With this we need to define the distance between strings. We used three different measures; The idea is that is a match is close enough by any of them, a match is detected These were:

1. Jaccard Distance: the percent of words that are the same.
2. Ratcliff and Obershelp Gestalt Pattern Matching: recursively finds longest matching sequences in both strings.
3. Modified Jaccard Distance, the same as Jaccard Distance but non-matching words that represent more than 2% (an arbitrary threshold) percent of the words in the are ignored. This the goal of this modification id to exclude things like 'college' and 'city' making 'city of toronto' a match for 'toronto'.

These measure can be used to match employers and job titles from given year to those in the next year and so on for every year in the dataset. In other words, for each year every employer is compared to the all employers found in previous years. If a match is found that's at least 75% we consider the two employers to be the same. If more than one such match is found, the strongest match is used (of any of the measures). In case of a tie one measure has to take priority; we chose plain Jaccard Distance.

So far the treatment of Employers and Job Titles has been the same, but at this point they diverge slightly. For employers, we assume that distinct employer strings in a given year are truly distinct. So two employers whose strings do not match exactly cannot be the same. No such assumption is made for job titles. And that's how we got properly categorical data for job titles and employers.

1.2 Names

The next step is to track names, i.e figure out which entries are for the same person. Again, we have to define how names will be compared. If any on the following is true for a pair of names they are considered to be a match:

1. Exact match, including initials.
2. Same first name, same last name.
3. No first name, Same first initial, same last name.
4. First initial same as first letter of first name, same last name.

Otherwise, a Ratcliff Obershelp Gestalt distance of 85% also makes two name a match. From here, the process is similar to that for employers; including, naturally, the assumption that all entries in a given year are truly distinct. For each name in each we look for matches among the entries from the previous year that have the same employer and job title. If none are found, we expand the search to those with the same employer only. If multiple matches are found in either search we have to break the tie (because two entries in the same year can't be for the same person). The first match types in the above list trump those lower down, so an exact match takes priority over a match of first and last names and so on. If both conflicting matches were detected using Gestalt pattern matching, the closer match wins. If these steps fail to resolve the dilemma, we simply take the entry with the closest salary.

1.3 Gender

Our dataset is somewhat odd in that it is not anonymized. We're provided with people's full names which allows us to make a few inferences about who they are. First among these is to determine everyone's likely gender. Many, presumably most, cultures tend to give different names to little boys and little girls. This is certainly true for Westerners, who obviously form the majority of the dataset.

Fortunately this work has been done by Kensuke Muraki [**?**]. Chicksexer (a reference to baby chickens, not women) is a publicly available, fully trained model that will take a persons full name and output an estimate the probability of a person with that name being male or female. As a sanity check, we verified that the results were reasonable with some names of people with known gender (people involved with this project and members of the cabinet).

1. Thomas Sattolo: Male: 1.0
2. Justin Trudeau: Male: 1.0
3. Catherine Mckenna: Female 1.0
4. Harjit Sajjan: Male 1.0
5. Bardish Chagger Female 0.88
6. Ahmed Hussen: Male 1.0
7. Carolyn Bennett: Female 1.0

All of these match reality.

1.4 Ethnicity

Names are also informative about people's ethnicity. Once again, this a project to extract this information is available. Sood and Laohaprapanon's project Ethnicolr [**?**] takes a full name outputs an estimate of the probability that a person's ethnicity is any of the following:

1. EastAsian

2. Japanese
3. IndianSubContinent
4. Africans
5. Muslim
6. British
7. EastEuropean
8. Jewish
9. French
10. Germanic
11. Hispanic
12. Italian
13. Nordic

And using the same names as before we can check the reasonableness of the results (only categories assigned a probability greater than 10% are included)

1. Thomas Sattolo: Italian 0.55, British: 0.24
2. Justin Trudeau: French: 0.69
3. Catherine McKenna: British: 0.84
4. Harjit Sajjan: IndianSubContinent: 0.87
5. Bardish Chagger: British: 0.85
6. Ahmed Hussen: Muslim: 0.74, Africans: 0.13
7. Carolyn Bennett: British 0.72, Jewish; 0.15

Once again, all of these seem reasonable, except for one. The result for Bardish Chagger is a complete mistake.

2 Prediction Model

One of the goal of this project was to create a model to predict future salaries of employees. Each employees salary throughout the years form a time series. The earlier years of these series can be used as the input of a neural network with the later years used as labels and the network should learn the patterns in a set of input time series and hopefully this will translate into a ability to predict salary growth of other people. The demographic can also easily be fed into the network. Making use of the data we gathered on sectors, employers and job titles is less obvious because it is all categorical, with many categories. One-hot encoding works as a means of inputting categorical information into a neural network, but doing this with all employers and job titles that exist in the dataset would mean having 10,000 dimensional input. This is not feasible, so what was done was to keep only the 100 most common employers and job titles and one-hot encode those. This isn't great because it does involve a significant loss of information, but there's no other obvious way to include employers and job titles in the model.

By taking this data and using it to train a neural network using Tensorflow, we were able to consistently predict future salaries with a mean squared error of roughly 0.25 (on zero mean unit variance input). This is not significantly better than simply linearly extrapolating from past salaries on each individual to predict the future. These results were not affected significantly by changes to the neural network architecture. Whether it was a small single layer network with 30 hidden units or a larger multilayer perceptron with 100 hidden units per layer, 0.25 was the best mean squared error we could get on the validation set. That we failed to usefully predict salary growth does not prove that doing so is impossible, but does suggest that the data we used on employers, job titles and demographics has only a weak effect on a salary changes (though not necessarily on the salaries themselves). It is worth noting that the average yearly change in salary for provincial employees is surprisingly large at $8312.