Find my NOC

CISC 372 – Project Report

Vedant Srinivasan Kartik Srinivasan | [17vsks@queensu.ca](mailto:17vsks@queensu.ca)

Abstract

NOCs are job classification codes that the Canadian government uses for organizing and analyzing the available jobs in the market. Job postings could be more informative if they have the corresponding NOC with them. We are finding a machine learning solution to map job postings to NOCs by taking advantage of the hierarchy style of the codes. The results show that we need to also explore methods that work better with smaller datasets or depend on other contextual information hidden in job postings.

Introduction

The NOC is a taxonomy of occupations in the Canadian labour market. Every occupation that can be found in the market has a corresponding code. Additional information is also collected about the occupation such as job duties, job requirements, and similar job titles. On one hand, this kind of classification helps in collecting and analyzing data about job market. With each renewal of the NOC, the changes reflect the addition of new job titles and modification of existing ones. It helps in supply and demand analysis and occupational forecasting by bringing structure into an otherwise unstructured type of data. On the other hand, the NOC can be used by jobseekers to find out what they can expect from their occupation and if the employer is using the job title correctly. The occupations are categorized by the duties and skill levels. The structure of the NOC is in the form of a hierarchy. At the top level, we have different sectors. As we go down the hierarchy, we have more granular differences between the occupations. At the bottom-most level, we have the actual job title.

Two real-world scenarios where NOCs are extremely important are Express Entry applications and job market analysis. The very first step in the Express Entry Immigration Program application is finding out what NOC the jobs you have completed are. The government uses this information to find out if the time spent at that occupation is eligible for the program. There are only certain skill types and NOCs that are covered under this program so it is important that the new jobseeker looking to immigrate must keep NOC in mind. Jobseekers could benefit from finding the NOCs even when they start applying for different jobs instead of later.

In terms of job market analysis, to understand which jobs are more in demand and which skills are needed in a region, we can look at the jobs offered by different employers and study the trends. The NOC makes this process streamlined. We can simply categorize job postings into their codes, and this will be useful in extracting category relevant statistics.

We can understand how a jobseeker can use NOC using an example of the job title ‘Database analysts and data administrators’ which corresponds to NOC 2172. The occupational profile for this job has example titles, main duties, employment requirements, and exclusions. The jobseeker can use this as a reference to make sure that the job offer that they have for a database analyst is similar and legitimate. They can also find out if NOC 2172 falls under the criteria for Express Entry. For data analysts, studying how many job postings corresponding to NOC 2172 will give us an idea of how trending this job is and what kind of resources we should dedicate to promote it.

Problem Statement

The project is aimed at using a machine learning solution to match job postings to job codes. The input is data obtained from external sources which are a result of scraping popular job posting sites. Some of this data is hand-labelled and this will be used as the ground-truth for training. The output would be a column which contain the correct NOC for each job posting. The classification needs to be done in a way that takes into account that the total number of classes is 500 and that there is a possibility that each NOC may not have enough data points for good training.

Broadly speaking, this is an attempt to come up with a simple and generalized method of classifying data that follow a hierarchy.

This problem is a modified subset of the work I am doing as part of an external research on supply and demand in the local job market.

Proposed Method or Solution

To start with the most straightforward approach, we could look at this problem as a classification problem. For example, we have collected 50,000 job postings and there are 500 NOCs in the hierarchy. These 500 NOCs do not all have the same number of job postings which is a result of different popularities and demand in the job market. There could even be the case where there are no job postings for a given NOC. Also, having 500 classes does not seem like a good idea since most of the classification we have seen so far are between 2 and 20 classes. Another problem is that we need to find what part of the job posting will reveal important information about the job classification.

We are given the job title, contents of the postings, date of the posting, salary, region, and a bunch of other columns that we could consider. Out of all this, the contents of the postings are the most comprehensive part of the posting. However, the problem with the contents is the large file size and heavy pre-processing that we must perform. Yet, we will go with this as the contents are more trustworthy than other things on the table. For example, the titles of jobs are not consistent amongst different companies. What could be called a software engineer in one place could be called programmer in another. In one company, we could have both software engineer and programmer titles available which have different roles and responsibilities. The content of the postings usually covers the duties of the job.

To solve the other problem of the large number of classes, we need to find a way to break down the problem into smaller subsets. Fortunately for us, we could achieve that by simply taking advantage of the hierarchical structure of the NOC. We could first classify the postings into one of the ten classes at the top level. This is a straightforward text classification problem.

We can then create ten partitions of the data. With each partition, we find classifications for the second level. Once again, we can partition the data into the different classes before find the classes for the next level. Each of the four digits of the NOC tell us which class it belongs to at that level. If we continue this method of partitioning and classifying, we will have all four digits for each of the postings.

Before all of the classification we also need to convert raw text into some sort of numerical format that the models can use. We will also need to perform some cleaning before that.

To clean the raw text, we can remove stop words, symbols, and numbers. We can also only keep words that are verbs and nouns as these will provide the chunk of the meaning. This is because we are looking for qualities and actions like ‘certified’ and ‘analyze’. We are assuming that sentence structure and context will not benefit us too much in this venture. By limiting the types of words, we are also greatly reducing the file size of the training data which was one of the roadblocks mentioned above.

We can also stem and lemmatize the text to root forms. This is done so that different forms of the same word are not considered separately. Once again, words like ‘analyze’, ‘analysis’, and ‘analyzed’, all share the same theme and for our purpose is the same concept.

Before training, we should also choose a numeric representation for the text. In this case, we will go for a standard tf-idf representation which converts the text into a matrix of numbers whose values tell us how important a word is in a particular document based on frequency.

When we are ready to train the model, we will split the data into 80-20 ratio each time we create a partition. We will use a standard accuracy measure to test our model on the 20% of the data.

We are going to use two different kinds of models here – logistic regression and linear SVM. Also, upon training, we will generate model dumps for each level and each parent class. This means that we will have a model at the second level that can distinguish between classes at the third level for that given parent class.

To solidify the method, let us consider the example of NOC 4011. Now, let us start with the top level which corresponds to the broad occupational category 4. Our top-level classifier that has 10 classes will give us this answer. Now, under 4, we have 40, 41, 42, 43, and 44. The classifier corresponding to the second level under parent class 4 should give us the answer 40. Under 40, we have 401, 402, and 403. The classifier corresponding to the third level under parent class 40, should give us the answer 401. Finally, under 401, we have 4011 and 4012. The classifier corresponding to the last level under parent class 401 should give us the answer 4011.

We can observe that at each level, the number of classes do not stay the same. The classifier for the second level under class 4 will have different features compared to the classifier for the second level under class 3. This is why we are making separate model dumps.

Experimental Results and Conclusions

Since we are using many classifiers together, we will obtain an accuracy score for each of them. The numbers are included in the project notebook.

The accuracies obtained are largely dependent on how many data points each classifier receives. This means that more popular NOCs had better classifiers since there were more job postings to find out their features correctly.

The results were poorer than I expected. The key accuracy is at the top level. If the postings are classified properly at the top level, then the performance at the bottom levels would be greater. To increase the accuracy at the bottom level we could manually annotate postings at the top level as this is just a matter of 1 of 10 classes.

The issue with gathering more data for training is that manual labelling of postings is tedious especially if all 4 digits must be annotated.

Some alternative methods that could be explored are to use a different text processing method instead of tf-idf and using a neural network architecture.

Another method could be to compare the text of the postings with the text provided in the occupational profile for each NOC. We could use text similarity algorithms that also take into account the meaning of words rather than just frequencies.

A neural network could have worked for this kind of structure since the results obtained at one level rely on the results obtained in the previous level.