Creating Variables for Social Scientists using Machine Learning?

Adrian Apaza (apaza@stanford.edu) https://github.com/adrianapaza/230Project

March 22, 2018

Abstract

In this paper I apply various deep learning/machine learning techniques to generate new variables for use in social science. I try various models to classify job posts as IT related or not. Various models yield different results suggesting that these models may be of limited use to social scientists.

1 Introduction

Machine learning, including deep learning, is a relatively new field that has yet to be fully embraced by social scientists. Many see machine learning models as black boxes that are hard to understand. Currently in social science it has been used for either dimensionality reduction when there are far too many variables and more recently to create new variables based upon model predictions and changes in prediction. Oftentimes variable creation is applied to cultural issues such as political polarization (Gentzkow et al., 2017). In this project I create a new variable to measure the rate of cultural change by considering online job postings as cultural artifacts.

The cultural change variable here is the change in predictive power. If predictive power decreases over time then we can infer that the words used to describe IT jobs are changing over time. We can then take this change over time variable as a dependent one for future research, such as the impact of a recession in year t on IT job role change by comparing predictive power of a model for jobs in year t-1, and t.

1.1 Related Work

As mentioned before Gentzkow et al. have been applying machine learning to social science problems. Kamenica and Bertrand (2018) have used changes in predictive power to create a variable corresponding to cultural divergence between rich and poor Americans over time. They see if you can still accurately predict someone's income based upon consumer behavior across many years.

Recently presented at the GSB was a dissertation chapter by (Bruce, 2018). He looked at U.S. government job postings and wanted to created a new variable of skill and knowledge differences between different job roles. Though he just looked to see

if the specific nouns and verbs used to describe a job were different across postings without using something like Word2Vec.

Boucher and Renault (2017) did work most similar to mine in that they looked at job postings. For this task I would consider that to be state of the art, and my best models applied methods very similar to theirs. GloVe is also considered state of the art in Natural Language Processing (Pennington et al. 2014)

2 Dataset

The dataset consists of pre web-scraped online job postings available on kaggle[1] and the link in the footnote has more descriptions of the data. It has 19,000 observations from 2004 to 2015. Jobpostings were posted through the Armenian human resource portal CareerCenter. It is unique in that the data was scrapped over such a long period of time. It is also unique in that the dataset is Armenian, yet the data is in English. That the data is in English shouldn't be too surprising given that Armenia is a very small country and there are many international employers who use the job site to recruit talented workers.

The data includes many details of the job posting including date posted, location, company name, job description, job requirements, required qualifications, and more. However there are some errors in the data set: some of job requirements, qualifications, or description may be empty and all be in one of the other categories, and sometimes they are even in location data. To relieve this I combine job description, location, qualifications, and requirements into one combined variable. The variable on interest though is another variable on whether the job was IT or not as identified by the job title.

Here is a sample job post for a Network administrator (so it is coded as an IT job):
- Network monitoring and administration;
- Database administration (MS SQL 2000).
- Excellent knowledge of Windows 2000 Server, Linux platform, Networking
TCP/ IP technologies, MS SQL 2000 Server;
- At least 2 years of experience in the proper field;
- Good knowledge of English.

Now we apply 3 different Glove Embeddings. One is from Pennington et al. and downloaded from their webpage[2] (I use their 3oo dimensional one using wikipedia and gigaword). One is an embedding from the job posting dataset: I set a window of size ten and it looks at words both preceding and following. Another is from both job postings and a corpus of Wikipedia text [3]. Once I have vectors for each possible word, I then get the average vector of each job postings (still 300 dimensional). So if there are N words, I sum the vector for each word along every dimension then divide by N and have a one by 300 vector. So for example that job posting above has values of [0.2935887, 0.3349259, -0.1279257, 0.3492532,...] and another job posting following that one that is

[1]https://www.kaggle.com/madhab/jobposts
[2]exact link to download is http://nlp.stanford.edu/data/glove.6B.zip
[3]Download file at http://mattmahoney.net/dc/text8.zip

not IT related has values of [0.8417046, 0.0005977303, -0.7734787, 0.1654267,...]. These final average vectors are the inputs of the standard neural network used later.

Notably though, the one from Pennington et al. may not include some misspellings in the job postings (which the others may). So in this case misspelled words not found in Pennington et al. are dropped in the calculations only for their embedding.

3 Methods

I took advantage of the open source h2o package developed by the startup H2 O.ai [4]. In this package, Nesterov (1983) accelerated gradient is a suggested method which I use. Nesterov's version alters momentum. I repeat the explanation offered by Ruder (2017) below. Given an objective (cost) function J, its gradient $\Delta_\theta J(\theta)$ and parameter θ (which are weights) and momentum term $\gamma * v_t$ where $\gamma \in [0, 1]$. We update the parameters as follows : $v_t = \gamma * v_{t-1} - \alpha * \Delta_\theta J(\theta)$ where α = learning rate and $\theta = \theta - v_t$. This ends up working quite well.

3.1 Experiments/Results/Discussion

3 experiments are run for each of the different possible embedding inputs. I first split each data set into training, dev, and test sets. Since there are 19,000 observations (job postings) I make 75% of the data training data and the rest split equally between test and dev sets. For each of the three different embedding inputs I tune different hyper parameters for the neural networks. The networks are fully connected and Relu activation functions are used at each layer (so $x_{l+1} = max(0, Wx_l^{(t)} + b_l)$). Although all 3 have 3 hidden layers. Logistic loss is used as cost function: $L(\hat{(}y), y) = -(ylog\hat{y} + (1 - y)log(1 - \hat{y}))$.

Hyper parameters are tuned based upon the whole dataset (not certain years). I do not use mini batches or dropout (they did not help substantially), and the hyper-parameters of interest I tune are the number of neurons in each layer and the L2 regularization parameter. Errors for train, dev, and test sets are found in the table below. For the word vectors from embeddings only from job postings we have 300 neurons in each of the three hidden layers and L2 regularization parameter of $2 * 10^{-3}$. For the Pennington vectors again there are 300 fully connected neurons in each layer and L2 parameter of $8 * 10^{-3}$. The vectors trained on combined wikipedia and Job postings has 400 neurons in each layer and L2 parameter of $8 * 10^{-3}$.

Here is an example of an IT job from 2015 that was mislabeled as not being an IT Job when we use word embeddings trained on job postings from 2004-2009. It is for an Android Developer and the job description starts as "The successful candidate will become part of the company's growing development team. He /she will be working mainly on various parts of the company's mobile applications. - Design and develop various Android applications including and not limited to features, solutions, responsive GUIs;- Design and build reusable modules to be used throughout company android framework; - Maintain and enhance company's home grown systems..." Now notably this is an Android development job, and Android did not even exist in 2004. In fact the

[4] https://www.h2o.ai/

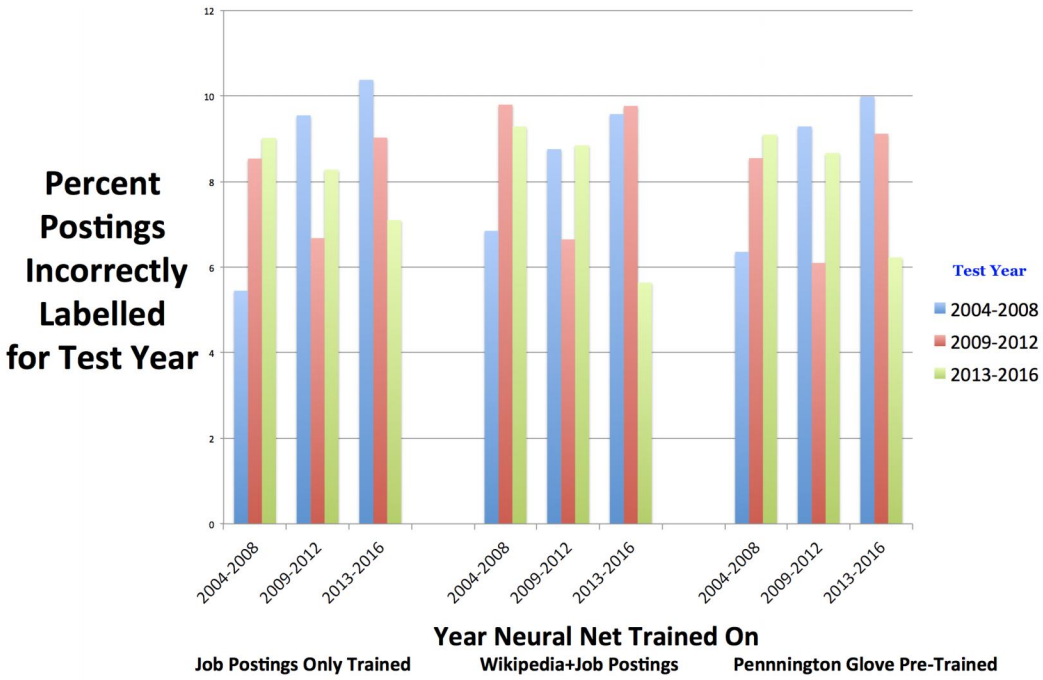
words that are closest (by cosine distance) are ios, developer, iphone, sdk, and mobile. So it is easier to see in this example how an IT job could be misclassified because new words have appeared.

Quantitatively I compare each different embedding model's neural network trained on a certain period of years (year batches chosen to be of equal sample size of roughly 6333, and within years the train set is 30% test is 70%) and see how predictive power changes over time for different test years. As we can see in the chart below for Pennington et al. embeddings and embeddings from job posts only the predictions get worse as the test years deviate more from the training years. However the same is not true for the embeddings from a combined wikipedia and job posting corpus, one can see that a neural net trained on 2004-2008 does worse on 2013-2016 test data than 2009-2012 data.

3.2 Future Work

For future work I would like to take bigrams into account. For example the word developer means something very different if it is preceded by 'real estate' as opposed to 'software'. A model that includes bigrams would consider 'estate developer' and 'software developer' as very different words.

However the heterogeneity in model performance across years is very important. That one model (word vectors from wikipedia and job posting) shows non-linear decreasing predictive power over time is important. Many social scientists, including those cited previously(e.g. Gentzkow et al.), consistently show changes in their models over time. In this case one can see that depending on the word vectors used, all of which seem like reasonable choices, one may find that predictive power steadily worsens over time or does not. However many social scientists know very little about machine learning and are not able to fully understand how simple choices of models may give different results. That they use these predictions and change in predictive power over time is dependent or independent variables in further regression models should be of concern.



	Job Posting Only Embeddings	Job Postings + Wikipedia Embeddings	Pennington et al. Embbeddings
Train Error% (N=14251)	1.64199	2.561224	2.526139
Dev Error% (N=2750)	6.231579	6.568421	6.357895
Test Error% (N=2750)	6.273684	6.989474	6.778947

4 References

Eric Boucher, Clement Renault (2017). Job Classification Based on LinkedIn Summaries. CS 224d Project

Josh Bruce (2018). Wage and Promotion Outcomes Due to Human and Social Capital Accumulation in the U.S. Civil Service, 1989-2011 . Presented At Stanford University March 12, 2018.

Matthew Gentzkow, Jesse M. Shapiro, and Matt Taddy (2017). "Measuring Polarization in High-Dimensional Data: Method and Application to Congressional Speech" NBER Working Paper 22423.

Emir Kamenica, Marianne Bertrand (2018). "Coming apart? Cultural Distances in the United States over Time". Presented at Stanford University February 26, 2018.

Yurii Nesterov (1983). A method for unconstrained convex minimization problem with the rate of convergence o(1/k2). Doklady ANSSSR (translated as Soviet.Math.Docl.), 269:543-547.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

Sebastian Ruder (2016). "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747.