Injury Records Text Classification

XIE Tian, WANG Chang, WANG Sidi

12/19/2019

1. Introduction

Every day, work-related injury records are generated. **In order to alleviate the human effort expended with coding such records**, the Centers for Disease Control and Prevention (CDC) National Institute for Occupational Safety and Health (NIOSH), in close partnership with the Laboratory for Innovation Science at Harvard (LISH), is interested in **improving their NLP/ML model to automatically read injury records and classify them** according to the Occupational Injury and Illness Classification System (OIICS). Our project is inspired by this initiative.

This project represents a **text classification** problem that is expected to be solved using efficient **big dataset** handling techniques and various **classificiation algorithms**. Through exploration, we hope to achieve better accuracy and higher efficiency in injury records classification.

2. Data Inspection

A random sample of 153,956 records with the outcome event column included. The data have 4 column (text, age, sex and a response label): We have 48 unique OIICS response label in total. Given that the records were collected through document scanning, many spelling errors are included. For example, "S P" in record 3 (shown below).

```r
Train <- read.csv("./data/CDC_Text_ClassificationChallenge_TrainData.csv")
head(Train, 3)
```

```
##                                                                                                                text
## 1                    57YOM WITH CONTUSION TO FACE AFTER STRIKING IT WITH A POST POUNDER WHILE SETTING A FENCE POST
## 2                                                  A 45YOM FELL ON ARM WHILE WORKING HAD SLIPPED ON WATER FX WRIST
## 3 58YOM WITH CERVICAL STRAIN  BACK PAIN S P RESTRAINED TAXI DRIVER IN LOW SPEED REAR END MVC NO LOC NO AB DEPLOYED
##   sex age event
## 1   1  57    62
## 2   1  45    42
## 3   1  58    26
```

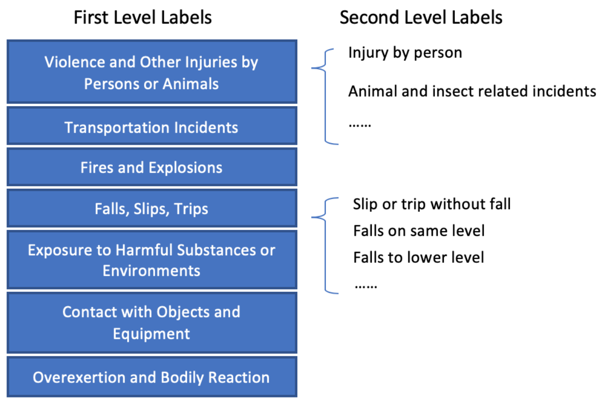
```r
dim(Train)[1]
```

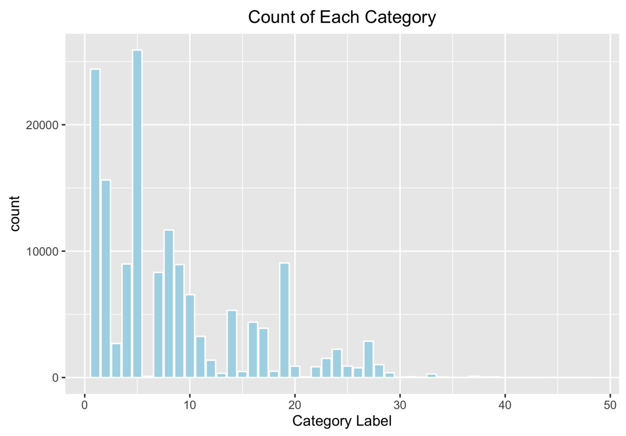
```
## [1] 153956
```

```r
summary(Train$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    0.00   27.00   37.00   38.29   48.00   96.00
```

The response variable is hierarchical (for example, if one label is 32, then it belongs to the third class "Fires and Explosions", and the second type). Figure on the left is showing the 7 first level labels and a few second level labels (48 in total). Although there are 48 possible outcomes, the frequencies is extremely unbalanced, as shown in the figure on the right. This needs to be taken into account while training different models.


First Level Labels
Second Level Labels
Violence and Other Injuries by Persons or Animals
Transportation Incidents
Fires and Explosions
Falls, Slips, Trips
Exposure to Harmful Substances or Environments
Contact with Objects and Equipment
Overexertion and Bodily Reaction
Injury by person
Animal and insect related incidents
……
Slip or trip without fall
Falls on same level
Falls to lower level
……



3. Data Processing and Prelimiary Analysis

a) Data Cleaning

In the text data cleaning part, we applied normal text-cleaning techniques including tranferring all letters into lower-cases, removing punctuations, word stemmming and word lemmatization. Specifically, the sex and gender information have been extracted in the dataset, so we remove the information by regular expression match.

Our text data is scanned from hand-written records, there are some wrongly-scanned words and characters, and the preposition may matter in the classfication. So we create our own stopping words list excludinhg prepositions and including single characters. Finally we replace those mis-spelled word with original ones. After cleaning, we get an injury records corpus with 6,000 vocabulary.

b) Build Document-Term Matrix

After cleaning the data, we get a cleaned-version text. AfteWe the build a document-term matrix, where each row is a injury record, each colum represents the voclabulary, and the element represent the vocabulary matrix. Since it's a huge matrix, we also store it in **sparse-matrix**.

We then visulize if the dominant words differ among the 7 category, this help us to determine if such document-term matrix will work in later model. Below are two word cloud example for class 1: *Violence and other injuries by animals or person* and class 4: *Falls, slips, trips*

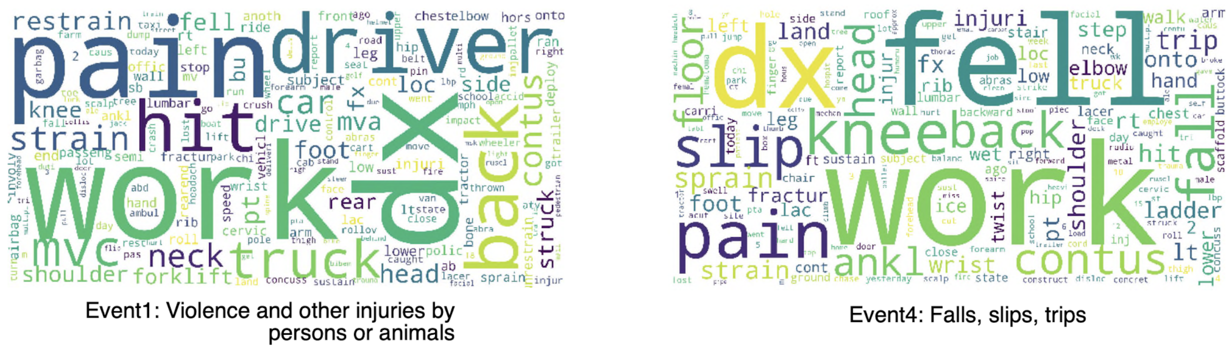


Figure 1: Word Frequency Cloud for Different Categories

c) Dimension reduction

For traditional methods, we will use document-wrod matrix as input. Although we use **sparse matrix** to store it, it is still "big". Thus we may try some dimension reduction methods:

- We tried **LDA (Linear Discriminant Analysis)** to project high-dimensional, sparse data to a low-dimensional, dense space, supervised by training data;
- We also tried **ANOVA test** on every single word and keep the significant ones only, which can reduce the dimension from 6000 to 2000.
- But both methods reduce the accuracy in fact, due to the loss of information. Considering **lasso penalty** can choose the informative feature automatically, we finally decided to use lasso penalty only rather than any other dimension reductions. For NLP methods, they don't process data as matrix, thus their corresponding "dimension reduction" is just different kinds of pre-processings.

4. Models and Results

We apply three representative method to this classification problem, where *logistic* methods are popular bacause of feasible interpretation, *Sequential NN* is a simple basic model with only 2 dense layers, and *BERT* is a complex pre-trained model. Below is the pipeline of our whole work.

a) Logistic regression

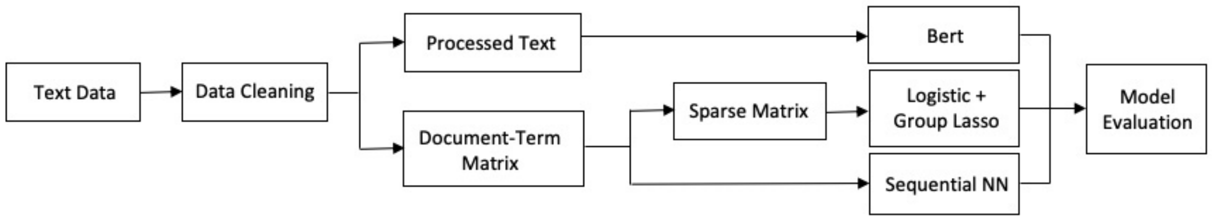


Figure 2: Pipeline for the project

Firstly, we applied logistic regression. The logistic regression assume that data have the following distributions:

$$P(Y_i = j) = \frac{exp(x_i^T \beta_j)}{\sum_{m=1}^{k} exp(x_i^T \beta_m)}$$

We want to maxmize the object function:

$$max\ Likelihood = \Pi_{i=1}^{n} \Pi_{j=1}^{k} [P(Y_i = j)]^{1_{Y_i=j}}$$

Or minimize the loss function:

$$min\ Loss = -\sum_{i=1}^{n} \sum_{j=1}^{k} 1_{Y_i=j} \times log[P(Y_i = j)]$$

Logistic regression with penalty

Group lasso:

$$min\ Loss = -\sum_{i=1}^{n} \sum_{j=1}^{k} 1_{Y_i=j} \times log[P(Y_i = j)] + \lambda \sum_{i=1}^{p} \sqrt{\sum_{j=1}^{k} \beta_{i,j}^2}$$

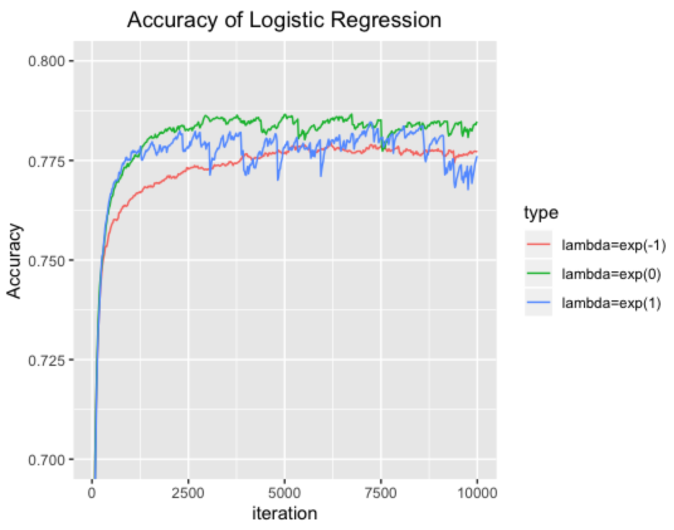
We apply lasso to automatically select and tune features. Since if one word is useful to predict one class, it is also useful to predict others (e.g. "fire" is positive to class "burn", then it is somehow negative to other class), we apply group lasso

Application of Logistic Regression

We use sparse matrix as input to fit the logistic regression.

At first we use **cv.glmnet** to perform logistic regression, but it will warn us the **unconvergence** (due to huge data), and fail to do all the cross validation. (In fact, when data is big, cv.glmnet's stopping criteria seems to be somehow arbitary and we can't check the convergence. Its accuracy is also between 0.75 - 0.79).

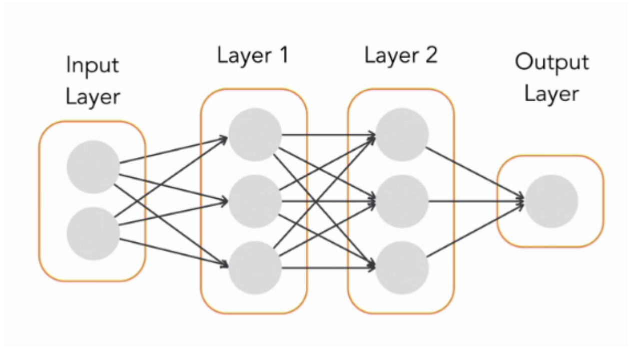
Thus we use **biostatistic cluster** to train the logistic model with **self-written codes**, we only show the result of an clustered-version logistic result here. Since we use group lasso with one additional penalty parameter λ, we show three results with different penalty in the following plot. (the training accuracy is not main concern, thus we show the test accuracy only)

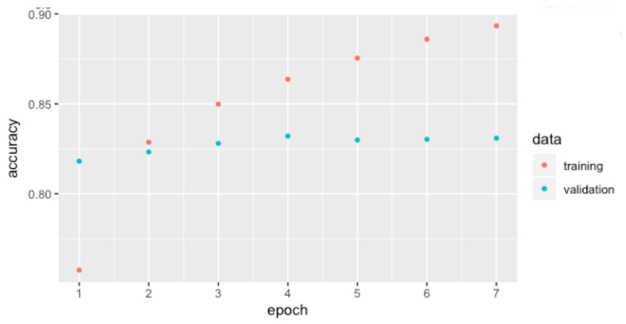


b) Neural Network: Sequential Model

Keras is a high-level tool for coding and training **neural networks**. Here we use **Keras package** in r to fit a simple sequential model to our injury data. A **Neural Network** is a machine-learning algorithm made up of individual nodes called neurons. Neurons are arranged into a series of groups called **layers**. Nodes in each layer are connected to nodes in the following layer. Data flows from the input to the output along these connections.

The easiest way to build a neural network is to use the **Sequential Model** class, which represents a linear stack of layers. Choosing the right training features is the key. A Neural Network containing two hidden layers is shown below on the left. We build a 2 dense layer network (only 1 hidden layer), the first one with dimension 300 and the last one with dimension 48.





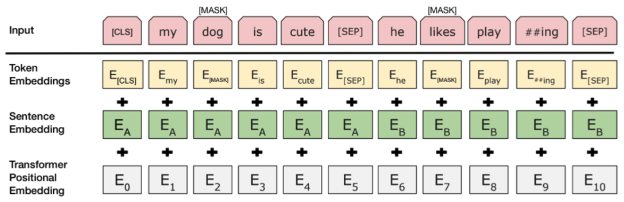
Sequential Models of different layers have been fitted to categorize injury records. After a very time-consuming tuning process, a single layer model showed the best testing accuracy. As shown in the figure above (on the right), even though training accuracy keeps increasing as epoch increases, testing accuracy stays at around **0.83** after epoch = 4 .

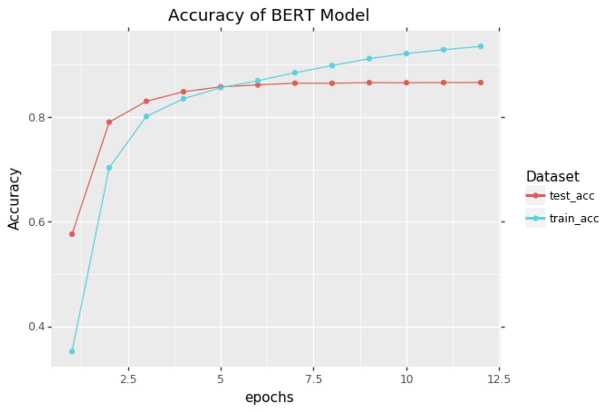
c) Neural Network: BERT Pre-trained Model

BERT (Bidirectional Encoder Representations from Transformers) is another neural network model, and applying this pre-trained model to our dataset is a kind of **Transfer Learning**. This deep and narrow nueral network model (12 layer with maximum dim 1024) has been trained by Google on large and high-quality text dataset, and it has achived excellent performance in lots of NLP tasks. So in our specific model, when we initialize our model with the **pre-trained** parameters, the model only need to be fine tuned, saving lots of training time.

Compared to the above two models, the input matrix representation here is rather complicated, including both the word, sentence and word position information. It's a huge nueral network with 1.2G parameter, in our small sample (10,000) dataset trial, at each epoch, the training time with batch size 60 is 90 minutes. So we finally train the model with **GPU** in google colab.

After 7 epochs, the model seems to converge on test dataset with an accuracy of around 0.87.





d) Comparison among Models

In experiments, the model precision and accuracy are recorded in order to make a comparison of different approaches. We'll use *Weighted-F1* score as model assessment. The input text represention of both traditional Logistic method and Sequential NN are the same, but the latter improves around 3% on both accuracy and weighted-F1 and runs 4.5 times faster than the former. Among all methods, BERT model performs best on the classification problem; since it is the most complicated model, it consumes the estimated longest time with a CPU, and runs satisfacorily with a **GPU**.

	Train_Accuracy	Test_Accuracy	Weighted_F1	Time
Logistic	0.80	0.80	0.79	3h
SequentialNN	0.89	0.83	0.82	0.67h
BERT	0.91	0.87	0.87	70h, 0.67h(GPU)

5. Conclusion and Future Work

In this injury records classification project, different models have been applied and evaluated. At the exploratory stage, we tried various prevailing traditional and deep learning methods in classification tasks, and finnaly focused on Logistic, Sequential NN and BERT models.

For a natural language problem, explainning the model may be complicated, since all vocabulary, word position and sentence as a whole may matter for the objective. BERT model makes full use of each injury records information, and achieves best result; however it's also the least efficient method.

In Logistic model, only vocabulary frequency information are used and there are 6000 predictors (vocabularies). Even with *Group Lasso* there are lots of significant words for each injury category. Compared to the other two methods, its performance are not satisfactory.

In Sequential NN we only build a basic model with 2 dense layer along with activation layer. It shows an excellent improvement on both weighted-F1 socre and efficiency compared with traditional Logitistic method. It's input matrix are also just document-term matrix, which shows that vocablary itself matters a lot in our classification problem.

We may expect improvement on both logistic and the Sequential NN model by introducing pre-trained word embeddings in data pre-processing part. For example, before feeding into models, we can project the high-dimension input matrix to low-dimension space with *Word2Vec*, *Doc2Vec* or *Glove*.