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
```

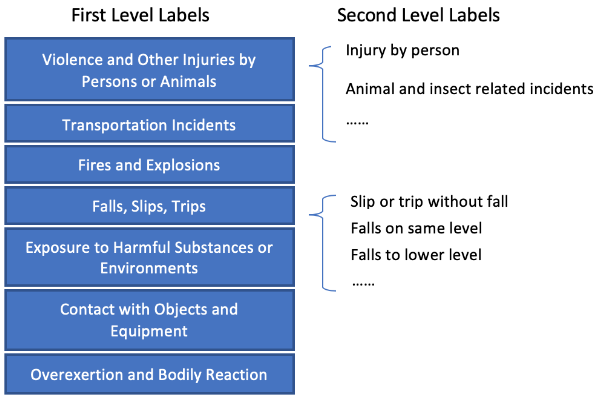
```r
dim(Train)[1]
```

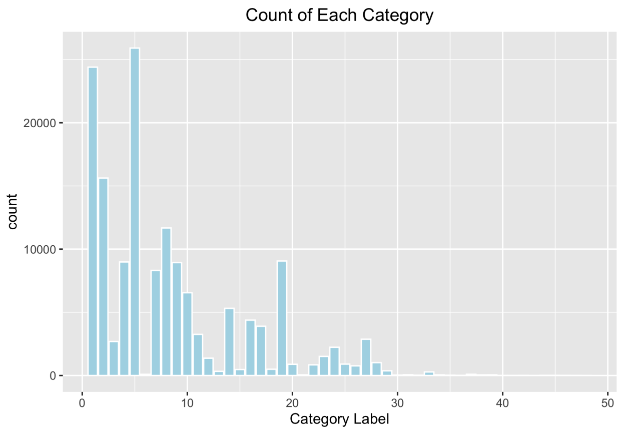
```
## [1] 153956
```

```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
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
Second Level Labels
Injury by person
Animal and insect related incidents
……
Slip or trip without fall
Falls on same level
Falls to lower level
……



3. Data Preprocessing and Prelimiary Analysis

a) Data Cleaning

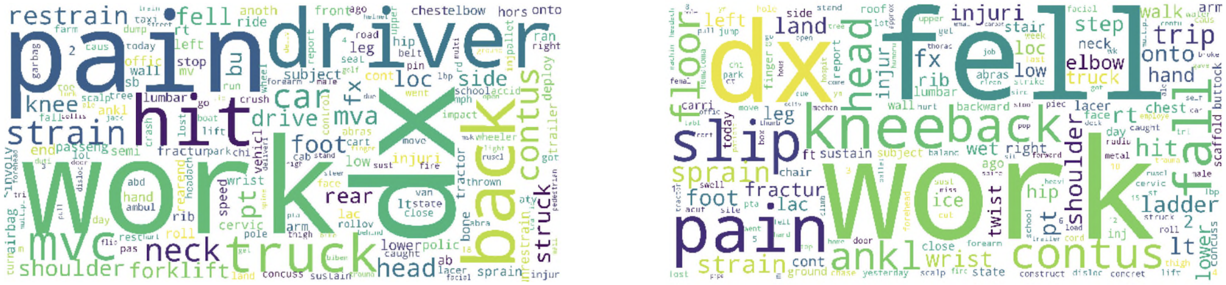
In the text data cleaning part, we applied common text-cleaning techniques including changing all letters into lower-cases, removing punctuations, word stemmming and word lemmatization. Specifically, the age and gender information were already included in the original dataset, therefore, we removed this information from "injury description" by regular expression matching.

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. Thus, we created our own stopping words list which excludes prepositions and includes single characters. Finally, we replaced those mis-spelled words with the correct ones. After cleaning, we produced an injury records corpus with 6,000 vocabulary.

b) Build Document-Term Matrix

After the cleaned-version text was generated, we built a document-term matrix, where each row is a injury record, each column represents one unique word from the vocabulary, and the elements represent the vocabulary matrix. Given that this matrix is very space-consuming, we transformed it into **sparse-matrix** form.

We then visulized the high frequency words in each injury category to see whether the dominant words differ among the 7 categories. This helps us illustrate the effectiveness of our document-term matrix in subsequent model fitting process. Below are two word cloud examples for class 1: *Violence and other injuries by animals or person* and class 4: *Falls, slips, trips*



Event1: Violence and other injuries by persons or animals

Event4: Falls, slips, trips

Figure 1: Word Frequency Cloud for Different Categories

c) Dimension reduction:

For traditional methods, we will use document-word 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 kept the significant ones only, which can reduce the dimension from 6000 to 2000.
- But both methods reduced the accuracy in fact, due to the loss of information. Considering **lasso penalty** can choose the informative features automatically, we decided to use lasso penalty only rather than any other dimension reduction methods.

NLP methods do not process data as matrix, thus their corresponding "dimension reduction" is just different kinds of pre-processings.

4. Models and Results

We applied three representative methods 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.

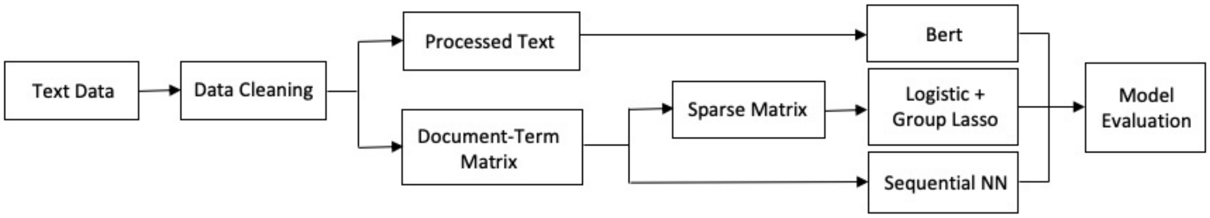


Figure 2: Pipeline for the project

a) Logistic regression

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:

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

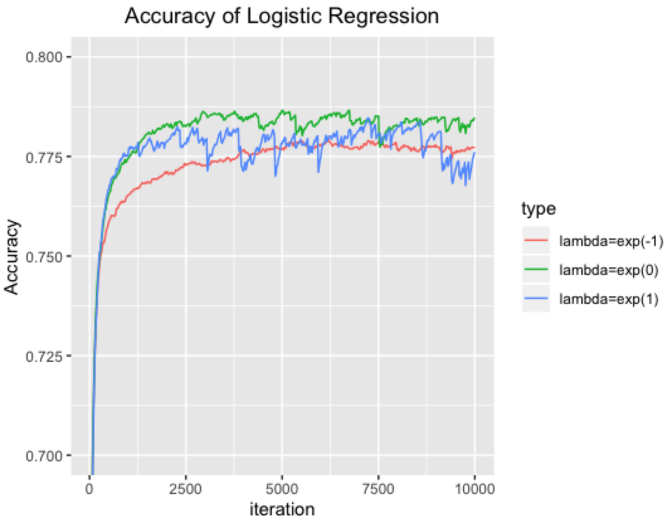
We applied lasso to automatically select features. If one word is useful to predict one class, it can also be useful to predict others (e.g. "fire" is positive to class "burn", then it is somehow negative to other class). Given that the sparse matrix still have many useless feature (i.e. noise), we apply group lasso to conquer this issue.

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 warned us with **unconvergence** (due to huge data), and failed to conduct any cross validation. (In fact, when data is big, cv.glmnet's stopping criteria seems to be arbitary and we cannot check the convergence. Its accuracy is between 0.75 - 0.79).

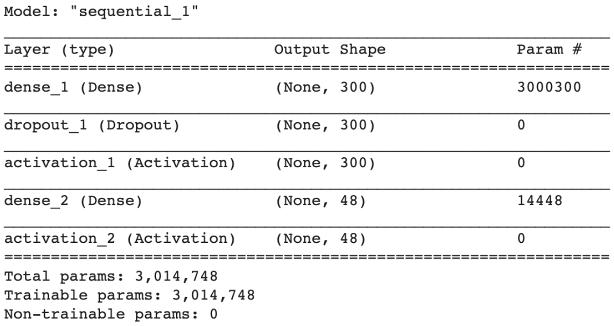
Thus we used the **biostatistic cluster** to train a logistic model with **self-written codes**, we only showed the result of one of the clustered-version logistic results here. Since we used group lasso with one additional penalty parameter λ, three results with different penalty were shown in the following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



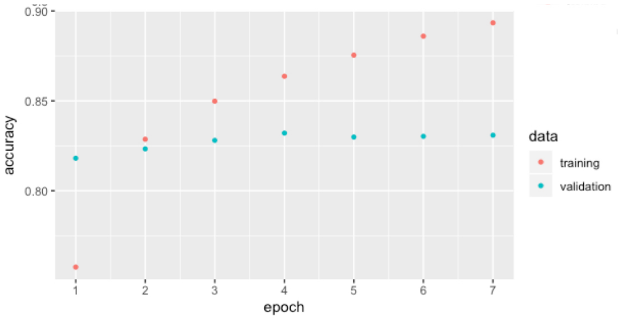
b) Neural Network: Sequential Model

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 built a 2-dense-layer network (only 1 hidden layer), the first one with dimension 300 and the last one with dimension 48.



```
Model: "sequential_1"
________________________________________________________________
Layer (type)                  Output Shape              Param #
================================================================
dense_1 (Dense)               (None, 300)               3000300
________________________________________________________________
dropout_1 (Dropout)           (None, 300)               0
________________________________________________________________
activation_1 (Activation)     (None, 300)               0
________________________________________________________________
dense_2 (Dense)               (None, 48)                14448
________________________________________________________________
activation_2 (Activation)     (None, 48)                0
================================================================
Total params: 3,014,748
Trainable params: 3,014,748
Non-trainable params: 0
```



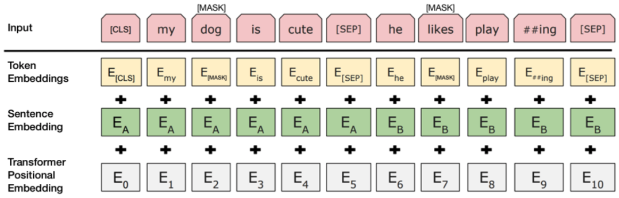
Sequential Models of different layers have been fitted to categorize injury records. After a very time-consuming tuning process, a single layer model showed the best testing accuracy. As shown in the figure above (on the right), even though training accuracy keeps increasing as epoch increases, testing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

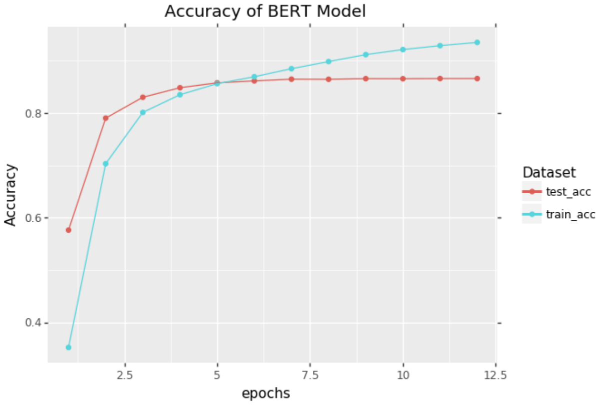
c) Neural Network: BERT Pre-trained Model

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 achieved excellent performance in lots of NLP tasks. Hence, in our particular case, when we initialized our model with the **pre-trained** parameters, the model only needed to be fine tuned, saving enormous training time.

Compared to the two models above, the input matrix representation here is rather complicated, including both word, sentence and word position information. It is a huge nueral network with 1.2G parameters. In our small sample (10,000 records) trial, the training time with batch size 60 is 90 minutes for running each epoch. To conduct it in a less time-consuming manner, we decided to 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.


Input
[CLS]
my
[MASK]
dog
is
cute
[SEP]
he
[MASK]
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}



d) Comparison among Models

In experiments, the model precision and accuracy are recorded in order to carry out a comparison of different approaches. We will use *Weighted-F1* score as the metric for 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 the best on this classification problem; since it is the most complicated model, it has the longest estimated running time on CPU, while 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	0.93	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 explored various prevailing traditional and deep learning methods for classification purpose, and eventually focused on Logistic, Sequential NN and BERT models.

For a natural language problem, explaining 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 the information in each injury records, and achieves best result; however it is 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 the activation layer. It shows an excellent improvement on both weighted-F1 socre and efficiency compared with traditional Logitistic method. Its 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 data into models, we can project the high-dimension input matrix to low-dimension space with *Word2Vec*, *Doc2Vec* or *Glove*.

Project Github: https://github.com/xietian99/DiseaseTextClassification