Automatic Cyber-violent Language Detecting and Classifying Using Machine Learning

Author: Kevin Huang

2020-01-02

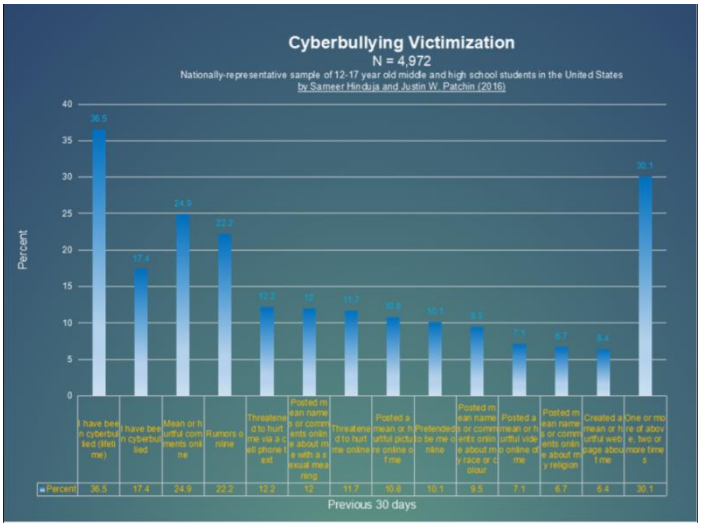
Contents

1. Brief Introduction

This project is aimed for constructing a machine learning model which can automatically classify cyber-violent comments online into 6 categories, including toxic, severe toxic, threat, insult, obscenity, and identity hate. Each comment may belong to more than one category. By doing so, system can ban comments belong to a specific category purposefully afterwards. In addition, this project can benefit website supervisor by letting them focus more on certain category. The accuracy of this machine learning model, finally, reaches above 95%.

2. Background information, Question Proposal, and Final Purpose

2.1 Background Information



As technology develops rapidly, any one may become a victim of cyber-violence nowadays. Cyber-violence, which may occur at any time and through any form, is now torturing people by using its own destructive forces. And, cyber-violent language, as a usual pattern of cyber-violence, is around every people, and negatively affect every online website user. According to a statistics published by Sameer Hinduja and Justin W. Patchin in 2016, there are 36.5% of American middle and high school students between the age of 12 and 17 who have been negatively affected by Internet violence. What's more, 30% of those students have experienced obscenity, discrimination and menace. Therefore, it can be seen that the prevalence of Internet violence in real life and its negative impact on people are very serious.

2.2 Question Proposal

After recognizing the severeness of cyber-violence, as a high school student, I think whether there is a way to solve this problem by using computer knowledge. By doing so, company can not only save the cost for direct human monitor but also

process every cyber-violent language accurately and quickly.

Therefore, the idea of using machine learning to solve cyber-violent phenomenon conjures up in my mind. Considering that Chinese character is difficult to transfer into bicameral numbers, I try to start my project with English comments in Wikipedia comment area.

2.3 Final Purpose

Detecting whether there is cyber-violent comment online. If so, the computer language would automatically classify those cyber-violent comments by their different purpose. In my project, all cyber-violent comments can be classified into 6 groups, including toxic, severe toxic, threat, insult, obscenity, and identity hate. By doing so, computer can purposefully ban *some* of the harmful comments.

3. Research Method

In order to achieve the ultimate goal of the project, which is to allow the computer to automatically categorize a comment, a series of steps are required, from the data collection in the early stage to the machine learning model after training. To begin with, I do the crawler, go to Wikipedia, and get all the comments down. As a result, I obtain lots of unlabeled data. The next step is manual labeling, transferring those unlabeled data into labeled data. I uploaded the untagged data to the specific tagging company to tag the comments, where "1" means "qualified" and "0" means "not qualified" (the specific data and their tagged results are detailed in part 3). Next step is to clean up the data, unify the format of all comment entries, and remove unnecessary parts or words (detailed in part 4).

After being converted into numbers that the computer can recognize, data will be used for the next step, a key process in the project. Through this process, the marked data mentioned above will no longer be the data for human to see, but the reference samples for computer learning. 90% of that data will become training sets, and another 10% will become testing sets. By having the machine learn the training set over and over again with its original machine learning model, a new, immature machine learning model will be generated. This new machine learning model will then be applied to the testing set, and an accuracy rate of this machine learning model will be given manually. If the accuracy meets the requirements, the model will become the final machine learning model (already mature). If the accuracy does not meet the requirements, it will re-enter the training set for training, resulting in a more

complete machine learning model. Then, repeat the previous steps and test its accuracy. Similarly, if the accuracy meet the requirements, the machine learning model is the output. If it does not, continue repeating the steps. This is constantly repeated until the accuracy of the final machine learning model reaches its optimal accuracy.

Here is an analogy to the process described above: the processed data is like a question bank, in which 90% of the questions are assigned as exercises (i.e., the training set mentioned above) and 10% are assigned as exams (i.e., the testing set mentioned above). A student uses the knowledge he has learned by himself to do all the exercises. After finishing all the exercises, this student's amount of knowledge obtained was improved. At this time, the student can use the new and more perfect knowledge system to do the test. After the test, if his score (that is, the accuracy) is satisfactory, then the student can graduate, his knowledge system is already perfect. But if the score is not ideal, then he needs to re-do the exercises and get a more complete and perfect knowledge system. If the new knowledge can let him pass the test, then he can graduate. If it does not again, he needs to repeat previous steps until his knowledge system is ideal which means he can use it to solve any question with high accuracy.

At the end of the machine learning process, a relatively perfect machine learning model is obtained, and the purpose of the project is achieved. Using this machine learning model, the computer can automatically categorize any comment as toxic, severe toxic, threat, insult, obscenity, and identity hate.

4. Main Procedures

According to the above research methods, the research content is now stated in detail below. The first part is collecting data. After crawling all the data from Wikipedia comment area, a total of 153,164 annotated data were obtained. Then, Put all those already annotated data into test_labels.csv. The following lines of code shows the fifth, sixth, and seventh data as an example. Thus, the fifth and seventh data do not involve any cyber violence, but the sixth data does involve cyber violence and can be classified as toxic severe toxic, obscene and insult.

The next step is to processing the data. Fist is to standardize the formatting, removing unnecessary information that prevents the computer from translating text, such as case, sender ID, @, HTTP prefixes, and punctuation for pauses or connections in sentences. Here are the details of the code:

Get rid of '\\n':

```
train['comment_text'] = train['comment_text'].map(lambda x: re.sub('\\n', ' ', str(x)))
```

Get rid of the comments begin with the word 'user':

```
train['comment_text'] = train['comment_text'].map(lambda x: re.sub('\[\[User.*', ' ', str(x)))
```

Get rid of IP address and user ID:

```
train['comment_text']=train['comment_text'].map(lambdax:re.sub("\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}",",str(x)))
```

Get rid of capitalized letters and transfer them into lowercase letters:

```
train['comment_text'] = train['comment_text'].map(lambda x: str(x).lower())
```

Get rid of http link:

```
train['comment_text']=train['comment_text'].map(lambdax:re.sub("(http://.*?\s)|(http://.*)",",str(x)
```

Get rid of all the punctuation except for " ' " used as a pause or a connection

between sentences:

```
train['comment_text']=train['comment_text'].map(lambdax:re.sub('[!"#$%&\()*+,-./:;<=>?@[\\]^_
`{|}~]',",str(x)))
```

By standardizing the format of all comments, the computer can easily convert words into numbers. In this project, I convert words into numbers(tokenizing the data) by the number of occurrences of the word. The more the number of occurrences, the smaller the number. The number marked ‘1’ is the one with the most occurrences, and the number of occurrences after 20,000 is represented by ‘0’. For example, a comment "I am smart" becomes a series of numbers based on the frequency the words occurs -- [1, 1000, 0]. The specific implementation code is as follows:

```
tokenizer = Tokenizer(num_words=20000)
tokenizer.fit_on_texts(list(X_train))
X_train = tokenizer.texts_to_sequences(X_train)
X_train = sequence.pad_sequences(X_train, maxlen=200)
```

The first line of this code, "tokenizer = Tokenizer(num_words=20000)," is intended to define a tokenizer (specifically, this line indicates: after the computer has determined the number of occurrences of all the words and ranked them according to their frequency, ONLY the words ranked as top 20,000 will be taken into consideration). The second line, "tokenizer.fit_on_texts(list(X_train))", is intended to train a tokenizer, specifically to sort. The more times the words appear, the smaller the number. The third line "X_train = tokenizer.texts_to_sequences(X_train)" is intended to use this tokenizer, which converts all text information to digital information using two lines of trained tokenizer. The last line, "X_train = sequence.pad_sequences(X_train, maxlen=200)", turns the numeric information into

the same length (in this case, max length is set to 200), with less complement (fill empty space with 0) and more truncation. By doing so, it takes the advantage of the characteristics of matrix computing, which can improve the speed of computing. Above is the whole process of data processing. The next step is for the computer to learn about the data that has already been processed, and to sort any data automatically. First, the resources in the keras package need to be referenced, including Convolutional Neural Network and Recurrent Neural Network. Here, I also import "tokenizer," "pad_sequence," "sequence," "Adam," and so on. The specific code is as follows:

```
from keras import initializers, regularizers, constraints, optimizers, layers
from keras.models import Model, Input, Sequential
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, SpatialDropout1D, Activation
from keras.layers import Conv1D, Bidirectional, GlobalMaxPool1D, MaxPooling1D, BatchNormalization
from keras.optimizers import Adam
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing import text, sequence
```

Next, start training the computer. Following the analogy used in part 3, X_train is like the problem, and y_train is like the answer. To tell a computer what questions and answers are respectively, such as the question here is "comment entry (comment_text)", and the answer here is different categories: "toxic", "severe toxic", "obscene", "threat", "insult", and "identity hate". The specific code is as follows:

```
X_train = train["comment_text"].values
```

```
y_train = train[["toxic", "severe_toxic", "obscene", "threat", "insult", "identity_hate"]].values
```

The data set is still two-dimensional, with rows and columns. Two dimensions are not enough to make the results precise. So we're going to convert this data into a vector of multiple dimensions. Here we've converted this two-dimensional data into 128-dimensional vectors. The following code shows how this is done:

```
max_features = 20000
max_len = 200
embedding_dims = 128
```

After further processing of our data set, it is an important part of machine learning. In this project, training the computer is like building blocks. The first layer is the embedding layer. At this level, the input dimension is the number of sentences * the number of words per sentence * the representation of each word. In symbols, it is n (representing the number of sentences) * 20,000 * 200 (this is the two-dimensional data mentioned above, which serves as the input). After machine learning, the output dimension becomes n*128*200 (which is 128 dimensions after conversion). The embedding layer code is as follows:

```
base_model.add(Embedding(input_dim=max_features, input_length=max_len,
output_dim=embedding_dims))
```

The second layer is the pooling layer. At this layer, input is the output of the previous layer, and the output then becomes n*128. Pooling layer code is as follows:

```
base_model.add(GlobalMaxPool1D())
```

The third layer is the dense layer. At this layer, input is the output of the previous layer, and the output becomes n*50. Dense layer code is as follows:

```
base_model.add(Dense(50, activation='relu'))
```

The fourth layer is the dropout layer. The purpose of this layer is to let the computer purposefully ‘forget’ 30% of all the data in order to prevent overfitting. At

this layer, input is the output of the previous layer, but the output is still n*50. Dropout layer code is as follows:

```
base_model.add(Dropout(rate=0.3))
```

The last layer is the dense layer. At this layer, the input is still the output of the previous layer, and the output becomes n*6. Last layer, dense layer, code is as follows:

```
base_model.add(Dense(6, activation='sigmoid'))
```

At this point, the main code for machine learning is complete. The next step is to factor in the error and optimize. Here is the code for this step. Loss is the distance between ideality and reality, metrics are the evaluation criteria for testing sets, and, to use the analogy of part 3, how you grade a test. Below is the code for this step:

```
base_model.compile(loss='binary_crossentropy',
                   optimizer=Adam(0.01), metrics=['accuracy'])
```

Here is the demonstration of the whole machine learning model：

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 200, 128)          2560000
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                6450
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 306
=================================================================
Total params: 2,566,756
Trainable params: 2,566,756
Non-trainable params: 0
```

Finally, let the computer automatically categorize the data in the testing set using new machine learning model. To use the analogy of part 3, it is like asking students to

use the already constructed knowledge structure for the test. X_train are the questions, y_train are the answers to the questions. Batch_size is the number of data in the specified training set, the amount of exercises, that is, every time the teacher assigns. Epochs is the number of times computer needs to repeat doing testing sets as well as the training set. validation_split is the proportion of training set and testing set. In this project, 90% of the entire data set is used as training set and 10% as testing set. Here is the code that the computer uses to do the testing set with the existing model:

```
base_hist = base_model.fit(X_train, y_train, batch_size=16, epochs=3, validation_split=0.1)
```

At last, the result is shown as follows:

```
Train on 143613 samples, validate on 15958 samples
Epoch 1/3
143613/143613 [==============================] – 497s 3ms/step – loss: 0.0681 – accuracy: 0.9771 – val_loss: 0.0575 – val_accuracy:
0.9806
Epoch 2/3
143613/143613 [==============================] – 502s 3ms/step – loss: 0.0603 – accuracy: 0.9775 – val_loss: 0.0603 – val_accuracy: 0.979
9
Epoch 3/3
143613/143613 [==============================] – 517s 4ms/step – loss: 0.0601 – accuracy: 0.9777 – val_loss: 0.0606 – val_accuracy: 0.9799
```

As can be seen from the result, the computer automatically classified the comments in the testing set and training set for three times. Both the training set and the test set produced results that were 95% or more accurate each time. Above are the specific procedures of this project.

5. Final Results and Analysis

5.1Final Results

Through research and exploration, a relatively complete machine learning model has been built, and the computer has used this model to automatically categorize the data in the testing set with an accuracy rate of about 97%.

5.2Analysis

This result is very satisfactory, after three training sets and automatic classification of the testing set, each time the computer classified 143,613 data in about 500 seconds. In the training set, the difference between ideal situation and actual operation is about 0.06, and the accuracy rate is about 97%. In the testing set, the difference between ideality and reality is about 0.059, and the accuracy is about 98 percent. This result indicates that the machine learning model we obtained through experiments is stable in the automatic classification of online comments. No matter how long or complicated the comments, the computer can use the project model to make a more accurate classification. It lays a good foundation for later manual censorship or systematic prohibition.

6. Flaws and Improvements

6.1 Flaws

First of all, the problem with this project is that the accuracy rate is not high enough. Even though the accuracy rate has reached 97%, there is still a chance for the accuracy rate to be even higher, even to 99.99%. That's the first problem with this project -- it's not accurate enough. The second problem is that the overall computer isn't running fast enough. So I need to look for a better machine learning model to make it run faster. Computer speed is an important factor. Faster speed can enable the computer to process more data in limited time. So the second problem is that it's not fast enough. These are the two main problems of the project so far.

6.2 Future Improvements

In the future, I need to find a more accurate model, adjust the structure of the model layers. In addition, searching for more data is also a crucial thing I need to adopt. Third, when processing data, I need to apply another method in order to achieve a faster running speed. Considering the running speed, finding a faster machine learning calculation method will be another optimal choice.

7. References：

1. 李航.统计学习方法[J].清华大学出版社，2012,3

2. Goodfellow, Bengio, Courville.深度学习[J]. 人民邮电出版社，2017,7

3. 周志华.机器学习[J].清华大学出版社，2016,1

4. 尼格尔·刘易斯.Python 深度学习[J].人民邮电出版社，2018,7