Toxic Comment NLP Analysis and Classification

Authors: Kaige Gao, Qiyuan Wang, Yijia Wei

1.- Executive Summary

The rapid development of technology has brought us access to vast amounts of information, enhancing our lives and boosting productivity. However, it has also led to an increase in toxic content, which can lead to potential harm by causing emotional distress, spreading discrimination, and damaging reputations. Our project aims to carry out an NLP analysis on the toxic comment dataset, present topic modeling analysis, and implement comprehensive models that classify toxic comments. Our project can be utilized in filtering harmful comments or posts in forums, and media comments. Additionally, it can be applied to chat AI, reducing potential harmful responses.

Obviously, our project can be applied to create a positive and friendly online environment. It would result in a better user experience, improving user stickiness and retention. Moreover, filtering harmful content can also maintain a good reputation for the brand and avoid potential legal disputes and negative incidents. From a social perspective, our project can contribute to building a more inclusive and friendly online community.

The data we use for this project is from Kaggle Toxic Comment Classification Challenge, which collected comments from Wikipedia's talk page edits. Data can be accessed from Toxic Comment Classification Challenge | Kaggle

2.- Data Preprocessing

2.1 Data Description - Summary of datasets and visualizations

The training dataset contains more than 155 thousand samples. Each instance contains 7 features after excluding the id number.

- comment_text: the original user comment on Wikipedia's talk page edits
- toxic: categorical data. 1 if the comment includes toxic words and 0 if not.
- severie_toxic: categorical data. 1 if the comment is considered severely toxic and 0 if not.
- obscene: categorical data. 1 if the comment includes obscene words and 0 if not.
- threat: categorical data. 1 if the comment includes threatening words and 0 if not.
- insult: categorical data. 1 if the comment includes insulting words and 0 if not.
- identity_hate: categorical data. 1 if the comment includes identity hate words and 0 if not.

To better understand the input data, we first visualize how much data are harmful and not harmful. The instance is harmful if any one or more categorical features have value 1. The instance is not harmful if all categorical features have a value of 0. We use a pie chart to

visualize the percentage of harmful and unharmful instances. From figure 1, instances that are not harmful capture 89.8% of the input data, while harmful instances capture 10.2% of the input data. The pie chart also indicates the input data is imbalanced.

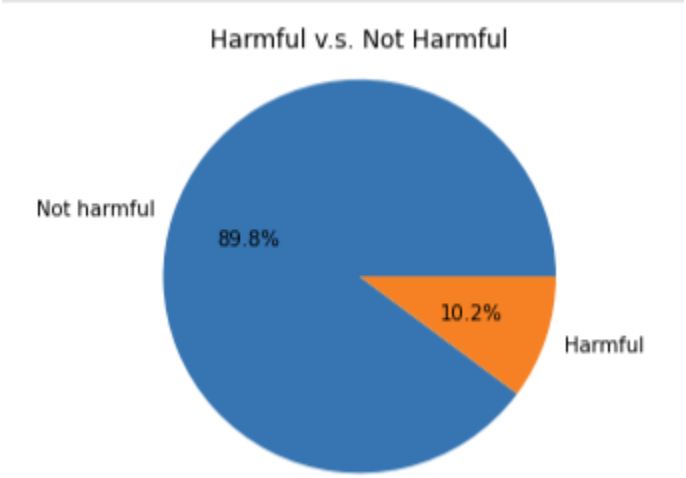


Figure 1: harmful data vs. not harmful data

For each categorical data, we created a bar chart to visualize the labels' frequency. As shown in figure 1, toxic comments have the highest occurrence, obscene comments have the second highest occurrence, and insult comments have the third highest frequency. All these three features have a much higher frequency than the rest three features, which are severely toxic, threat, and identity hate.

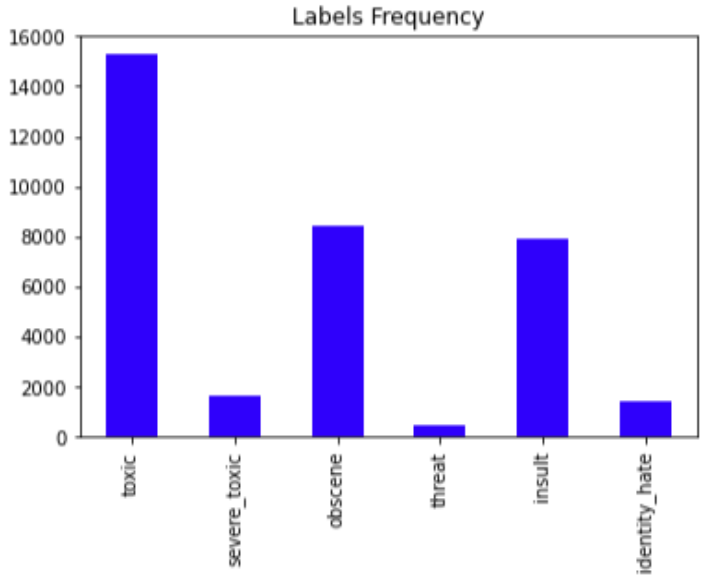


Figure 1: Labels Frequency of Each Feature

Since some comments can be identified into multiple labels, we find each combination's occurrence. Figure 3 shows some combinations of labels with the highest and lowest frequencies. The top three combinations are [toxic only], [toxic, obscene, insult], and [toxic, obscene]. The lowest three combinations are [obscene, threat], [toxic, severe toxic, threat, identity hate], and [toxic, severe toxic, threat, insult]. Since those combinations that are most frequent include toxic labels, the result in figure 3 aligns with the result in figure 2. Figure 4 is a pie chart that can help us better understand the frequency and the breakdown of different combinations of harmful comments.

Combination	Count
Toxic	5666
Toxic + Obscene + Insult	3800
Toxic + Obscene	1758
toxic + Insult	1215
Toxic + Severe Toxic + Obscene + Insult	989
Toxic + Obscene + Insult + Identity Hate	618
...	...
Obscene + Threat + Insult	2
Obscene + Threat	2
Toxic + Severe Toxic + Threat + Identity Hate	1
Toxic + Severe Toxic + Threat + Insult	1

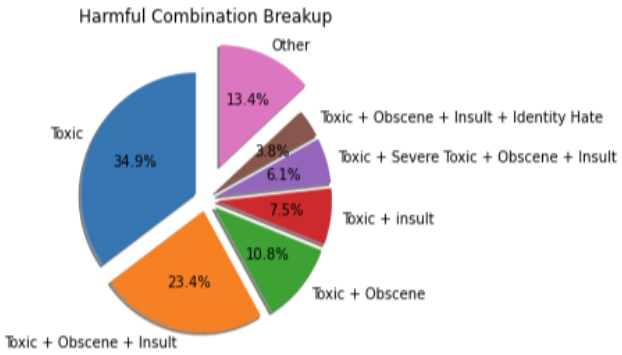


Figure 3: Count of each Combination (left)
Figure 4: Pie chart of harmful comment combination breakup(right)

2.2 Data Cleaning Metrics

In our data-cleaning process, we applied the 6 steps by using the NLTK package to the column comment_text so that the text can be understood by the machine in the future model. These 6 steps are:

- Contractions Expansion: Contraction expansion converts contractions in the text to the expanded form. For example, “don’t” is a contraction of “do not”; “can’t” is a contraction of “cannot”; and “he’s” is a contraction of “he is”. Contraction expansion can standardize the format of our text or token for accurate analysis.
- Punctuation Removal: We eliminated all the punctuation marks from the text to avoid algorithm interference since punctuation usually does not carry useful information, including periods, commas, semicolons, colons, question marks, exclamation marks, etc.
- Digit Removal: We eliminated all the digits in the text for efficiency and accuracy.
- Text Tokenization: In text tokenization, each comment or text is broken into individual words, phrases, sentences, or other meaningful elements. In our project, the tokenization is based on word level, which means we broke the text into individual words based on spaces or punctuation marks.

- Stopword Removal: Stopword removal eliminates those common words that do not carry too much significant information. Some frequent stopwords are “the”, “a”, “an”, “and”, “or”, “in”, “on”, etc. We removed stopwords to reduce the dimensionality of data to make further analysis and machine learning models more efficient and accurate.
- Stemming: Stemming is the process of reducing a word to its base form by removing prefixes and suffixes to improve efficiency and performance. Some frequent prefixes are “re-”, “dis-”, “un-”, “over-”, etc. Some frequently used suffixes are “-tion”, “-ly”, “-ing”, “-ed”, “ment”, etc. Stemming might encounter some errors when dealing with irregular words, which would lead to more advanced techniques, such as lemmatizations.
- Lemmatization: Similar to stemming, lemmatization also reduces the word to the base form. Unlike stemming, which simply remove prefix or suffix, lemmatization considers the linguistic information, such as grammar rules, for a more accurate base form. For example, “better” is sometimes stemmed to “bett” but lemmatized to “good”. “Happiness” is stemmed to “happi” but is lemmatized to “happy”.

To better understand the process of text cleaning, we use one comment as an example.
Original Text: *“Bye! Don’t look, come or think of coming back! Tosser.”*
Contraction Expansion: *“Bye Do not look come or think of coming back tosser”*
Punctuation Removal: *“Bye Don’t look come or think of coming back tosser”*
Tokenization*: [bye, do, not, look, come, or, think, of, coming, back, tosser]*
Stopwords removal: *[bye, do, not, look, come, think, coming, tosser]*
Stemming: *[bye, do, not, look, come, think, com, toss]* or Lemmatization *[bye, do, not, look come, think, come, tosser]*

As a study based on text data, outlier detection becomes challenging because it is hard to distinguish whether the data is an outlier or the natural variation/pattern in human language. Additionally, the outlier detection methods or concepts for traditional data sometimes do not fit text data well because of the data sparseness, number of sub-groups, and distance concentration. Therefore, we have not yet applied outlier detection at this step. If time allows or is necessary for further analysis, we will learn and apply outlier detection for text data in future steps.

3.- Model Updates

1) The LDA model

Latent Dirichlet Allocation (LDA) is a statistical model used for topic modeling, which classifies documents into natural topics although these topics are not exactly what we are looking for.

The algorithm of an LDA model:

a) Randomly assign each word in each document to one of k topics.
b) For each document d, look at each word w and compute:

p(topic t|document d): the proportion of words in document d that are assigned to topic t;

p(word w|topic t): the proportion of assignments to topic t in all documents that come from this word w

c) Update the probability for the word w belonging to topic t:

p(word w, topic t) = p(topic t|document d) * p(word w|topic t)

The step will be repeated multiple times until a steady state is found.

```
[(0,
  '0.022*"equal" + 0.016*"articl" + 0.013*"would" + 0.012*"one" + '
  '0.010*"delet" + 0.009*"use" + 0.007*"time" + 0.006*"issu" + 0.006*"channel" '
  '+ 0.006*"theori"'),
 (1,
  '0.015*"articl" + 0.012*"delet" + 0.011*"v" + 0.010*"use" + 0.009*"bout" + '
  '0.009*"say" + 0.009*"equal" + 0.008*"bridg" + 0.008*"mani" + 0.008*"name"'),
 (2,
  '0.019*"would" + 0.015*"articl" + 0.015*"page" + 0.011*"imag" + 0.009*"use" '
  '+ 0.009*"delet" + 0.009*"wikipedia" + 0.008*"think" + 0.007*"like" + '
  '0.006*"polici"'),
 (3,
  '0.014*"use" + 0.010*"see" + 0.009*"would" + 0.008*"f" + 0.008*"plea" + '
  '0.007*"page" + 0.007*"one" + 0.006*"imag" + 0.006*"canada" + '
  '0.006*"wikipedia"'),
 (4,
  '0.017*"f" + 0.015*"bobov" + 0.013*"plea" + 0.010*"name" + 0.009*"even" + '
  '0.009*"rebb" + 0.008*"say" + 0.007*"believ" + 0.006*"titl" + '
  '0.006*"option"'),
 (5,
  '0.013*"edit" + 0.010*"philosophi" + 0.008*"ukrainian" + 0.008*"would" + '
  '0.007*"armi" + 0.007*"newspap" + 0.007*"mean" + 0.007*"scienc" + '
  '0.006*"plea" + 0.006*"http"'),
 (6,
  '0.016*"f" + 0.014*"delet" + 0.011*"page" + 0.011*"articl" + 0.007*"get" + '
  '0.007*"option" + 0.007*"state" + 0.006*"thing" + 0.006*"one" + '
  '0.006*"cultur"'),
 (7,
  '0.017*"talk" + 0.015*"page" + 0.012*"use" + 0.010*"articl" + 0.010*"sourc" '
  '+ 0.010*"see" + 0.008*"block" + 0.008*"someon" + 0.007*"plea" + '
  '0.007*"edit"'),
 (8,
  '0.015*"page" + 0.012*"version" + 0.011*"user" + 0.011*"attack" + '
  '0.010*"talk" + 0.010*"plea" + 0.009*"articl" + 0.008*"keep" + 0.008*"delet" '
  '+ 0.008*"ground"'),
 (9,
  '0.010*"bobov" + 0.009*"edit" + 0.009*"name" + 0.008*"recreat" + '
  '0.008*"block" + 0.007*"plea" + 0.007*"rebb" + 0.006*"page" + 0.005*"beth" + '
  '0.005*"refer"')]
```

Figure 5: Topics of the LDA model

The next step to our goal is to analyze the topics of the LDA model to detect harmful comments. Other models and algorithms that may be used are: XGBOOST, XLNet, KNN, fastText, SVM, Naive Bayes.

2) BERTopic Model
3) Logistic Regression
    a) TFIDF: Before we construct the logistic regression model, we need to vectorize the text into a meaningful representation of numbers to fit the machine learning

algorithm. The vectorizer algorithm we are using for this project is Term Frequency Inverse Document Frequency (TFIDF).

$$TFIDF(t) = TF(t, d) \times IDF(t)$$

$$where\ TF(t, d) = \sum_{x \epsilon d} frequency(x, t),\ and\ frequency(x, t) = 1\ if\ x = t$$

$$and\ IDF(t) = log\frac{N}{1+nt}$$

$$N\ is\ the\ number\ of\ documents\ in\ the\ corpus$$

$$nt = |\{d \epsilon D: t \epsilon d\}|$$

For convenience, we are using TFidfVectorizer from sklearn package to vectorize the comment in our project.

b) Building Logistic Regression Model

Logistic regression uses logistic function to predict binary or multinomial classes. In our project, we are classify the comment as either harmful or not harmful. It is reasonable to use logistic regression model in our study as a baseline and later compare to other models. The logistic function we usually use is called sigmoid function, which can be calculated as:

$$\sigma(a) = \frac{1}{1+e^{-a}}\ and\ \frac{d\sigma}{da} = \sigma(a)(1 - \sigma(a))$$

The logistic regression models the probability as the output based on input. Then by chossing the cutoff value, we classify the input as True or 1 if probability is greater than the cutoff value. We classify the input as False or 0 if probability is smaller than the cutoff value. The estimation process is usually done by the maximum likelihood estimation (MLE) approach.

The accuracy of our logistic regression model is 0.89. As a baseline model of our project, the logistic regression is successful.

```
Accuracy of logistic regression: 0.8923790589794566
```

4) XGBoost Model

We also create a classifier based on the Extreme Gradient Boosting algorithm. XGBoost is a scalable and distributed gradient boosted decision tree (GBDT) algorithm by providing parallel tree boosting instead of sequencial boosting in GBDT to push both computational speed and model performance to a higher level. Gradient boosting means to improve a relatively weak model by combining it with other relatively weak models to construct a stronger model that includes all of them. The process of collecting weak models and creating stronger models is using the gradient descent algorithm.

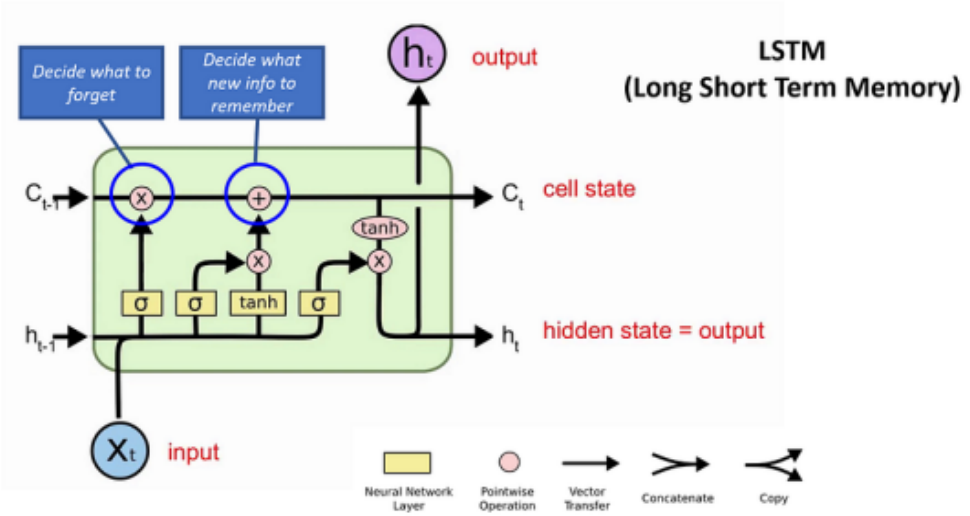
The performance of XGBoost is relatively same as logistic regression. We achieve an accuracy of 0.88.

```
Accuracy of XGBoost: 0.8803180914512923
```

5) The LSTM Model

The Long Short Term Memory (LSTM) is a variety of Recurrent Neural Network (RNN). The aim of LSTM is to overcome the disadvantages of simple Recurrent Neural Network by allowing the network to store short term memory data and use it in the later process of the network, instead of completely changing the existing data.

The role of the LSTM model is succeeded by the cell state. With few computations, information will be removed and added to the cell state throughout the LSTM network. On the other hand, the hidden state can provide short term memory that we may need to update the cell state with the new input.



Layer (type)	Output Shape	Param #
inputs (InputLayer)	[(None, 200)]	0
embedding (Embedding)	(None, 200, 50)	100000
lstm (LSTM)	(None, 64)	29440
FC1 (Dense)	(None, 256)	16640
activation (Activation)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
out_layer (Dense)	(None, 1)	257
activation_1 (Activation)	(None, 1)	0

Total params: 146,337
Trainable params: 146,337
Non-trainable params: 0

Figure 6: LSTM Layer structure

The current accuracy for the training set is about 93%.

Our first step of training the LSTM model seems to be successful and we will seek to improve it by validation methods and model evaluation.

6) machine learning workflow

$$\mathcal{ML} = \mathcal{ML}_9 \circ \mathcal{ML}_8 \circ \mathcal{ML}_{0\text{-}7} =$$

Input Space: X^n text data, n = 155k observations

Output Space: [0, 1]

Learning Morphism: $F(\boldsymbol{x};\ \theta_{0\text{-}7}, \boldsymbol{w}, \theta_9) = F_9 \circ F_8 \circ F_{0\text{-}7}$

Parameter Prior: $P(\boldsymbol{x};\ \theta_{0\text{-}7}, \boldsymbol{w}, \theta_9) = 1$

Empirical Risk Function: $\sum 1(F(x_i;\ \theta_{0\text{-}7}, \boldsymbol{w}, \theta_9) = y_i)/\#$ of test data

$\mathcal{ML}_{0\text{-}7}$: Data Preprocessing: Contractions Expansion, ..., Lemmatization

$\mathcal{ML}_8$: Long Short Term Memory

$\mathcal{ML}_9$: Evaluation for accuracy

4.- Source Code

[QY-W/Comments_NLP_Classification (github.com)]

5.- Next Step

Our primary goal is to detect harmful comments that can be deployed to filtering system, however, we may additionally implement a multiple-label model for academic purpose to practice potentially handling potential problems. Additionally, more visualization on NLP can be carried out, the word distribution for each topic and t-SNE Clustering Chart.

At this point, as figure 6 shows, a simple LSTM model is fitted, and preprocessing is not completely done to the fitted dataset. When using TensorFlow RNN, we discovered that our previous steps are not completely compatible with the model and further adjustments will be made in further

We are to further understand the RNN structure and plan to deploy more complex models so as to improve the performance. We are considering building hybrid model with LSTM and XGBoost due to the relative high accuracy of LSTM.

Timeline:

Week 11: Explore LSTM-XGBoost hybrid model
Optimization
Add more visualization
Week 12: Final check, catch up progress
Prepare presentation
Week 13: Presentation

6.- Reference

https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data
https://builtin.com/machine-learning/nlp-machine-learning
https://analyticsindiamag.com/pseudo-labelling-a-guide-to-semi-supervised-learning/
Text Messages Classification using LSTM, Bi-LSTM, and GRU | by Nuzulul Khairu Nissa | MLearning.ai | Medium
An Improved LSTM Structure for Natural Language Processing
BERT and fastText Embeddings for Automatic Detection of Toxic Speech
https://en.wikipedia.org/wiki/Vanishing_gradient_problem